RDF-Based Retrieval of Information Extracted from Web Product Catalogues

Ondřej Šváb, Martin Labský, Vojtěch Svátek
University of Economics, Prague, Department of Information and Knowledge Engineering
Winston Churchill Sq. 4, 130 67 Praha 3, Prague, Czech Republic
{xsvao06,labsky,svatek}@vse.cz

ABSTRACT

Extraction of relevant data from the raw source of HTML pages poses specific requirements on their subsequent RDF storage and retrieval. We describe an application of statistical information extraction technique (Hidden Markov Models) on product catalogues, followed with conversion of extracted data to RDF format and their structured retrieval. The domain-specific query interface, built on the top of *Sesame* repository, offers a simple form of navigational retrieval. Integration of further web-analysis methods, within the *Rainbow* architecture, is forthcoming.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Languages

Keywords

RDF, Information extraction, Hidden Markov Models, Product catalogues

1. INTRODUCTION

Retrieval-oriented tasks are among the best-developed in semantic web applications, thanks to the ubiquity of web search as well as to the maturity of database (or even XML) querying. One variety of 'semantic web' retrieval aims at annotations already expressed in semantic languages such as RDF or OWL; these can be either dispersed in the *WWW space*, which calls for co-operation with keyword-based search engines [18], or collected in specialised *repositories* [2, 7]. Another 'retrieval' stream deals with extraction of information (potential annotations) from *raw web data*, and focuses on the level of *individual documents or sites* [9]. Although each of the streams has been around for a while, interactions among them are not always well investigated at the moment.

In this paper, we present an ongoing study on coupling (raw-web) information extraction with repository querying, within a selected domain of discourse, namely *bicycle sale offers*. This study is part of the *Rainbow* project[1], which aims at semantic analysis of web content and structure using a wide scope of knowledge-based methods implemented as independent web services. Our aim is to build a domain-specific semantic search engine capable of answering queries e.g. about product names and prices, and pointing the user to the original web sites. We believe that the RDF format is rather suitable as underlying representation for such search engine. On the one hand, we need a format capable of expressing structured data about diverse entities identified on the web, and flexible enough to cope with the problem of incomplete or inconsistent data, obviously arising when automatically processing raw web resources. On the other hand, we do not need the expressivity of a full ontology language such as OWL, since we will rarely be able to extract general axioms.

Section 2 is devoted to the problem of information extraction from product catalogues. We discuss the general features of this task, describe our experimental data and their acquisition process, and present three variations of statistical models used for the annotation subtask as well as the (baseline) heuristic algorithm used for the 'product-offer' composition subtask. Section 3 addresses the processes of storing and querying the results of information extraction obtained in the previous step (and possibly by other methods), in the RDF format. We show the underlying RDF Schema ontology, explain the choice of RDF repository and query language, and describe the functionality of the end-user query interface. Section 4 surveys some related work. Finally, section 5 wraps up the paper and outlines prospects for future work.

2. IE FROM PRODUCT CATALOGUES

2.1 Principles and Problems

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)

[1] `http://rainbow.vse.cz`

SIGIR '04 Semantic Web Workshop, Sheffield, United Kingdom

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 CROSSMARC[2].

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 [15, 16], 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[3] 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.

2.2 Experimental Data

As training and testing data for our extraction models, we manually annotated 100 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. 1. The six most frequent slots are enumerated in the first column of Table 1. The labelled collection is available from `http://rainbow.vse.cz`.

2.3 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 [21]. 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* [21].

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.

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

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

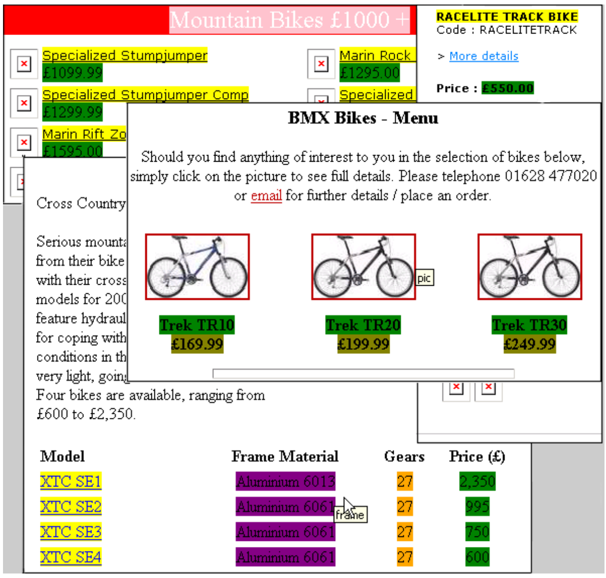

Figure 1: Samples of annotated training data

`<addtobasket>` was used in place of forms that satisfied a set of manually-defined rules.

Experiments were carried out using three different HMM architectures. In all cases, we experimented with a *trigram*[4] HMM instead of the commonly used bigram, seeking to capture farther-reaching dependencies between slots. The *smoothing* method applied on lexical probabilities was absolute discounting similar to [3], while for transition probabilities, linear interpolation was used.

The three architectures were as follows:

1. In the *naive approach* inspired by [19], we represented each semantic slot with a single *target* state. Additionally, we defined a *prefix* and a *suffix* state for each slot, responsible for modelling typical left and right contexts. Finally we used a single (shared) *background* state producing uninteresting data. The model topology is shown at Fig. 2[5]. In contrary to [19], where independent models were built for each slot, we created a single model containing all slots in hope of capturing the characteristic inter-slot positioning (e.g. price typically following name).

   We trained the naive model directly using counts from labelled training data, as there always was a single state sequence visible in each labelled document. Since the prefix and suffix states for each slot were not directly labelled in the data, we treated k preceding and k following tokens as being emitted by these states (in our experiments $k = 2$).

2. In the *word N-gram model*, we incorporated knowledge of internal structeres of a slot, namely, substituted the unigram lexical distributions of chosen states

[4] I.e., with transition probabilities conditioned by *two* previous labels.

[5] In the diagram, we omitted the edges for transitions between the shared background state and the states P', T' and S', and did not include further P-T-S state triples at all, for better readability.

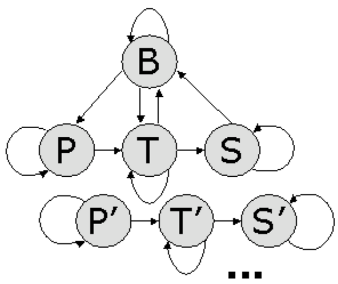

Figure 2: Basic HMM architecture used

Table 1: 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

with *n-gram* lexical distributions. The state structure and transition distributions remained unchanged compared to the naive model. In our experiments we use *word trigram*[6] models for selected slots, trained from the particular slot's training data and smoothed via linear interpolation with weights obtained using the EM algorithm described in [12][7]. To obtain the best state sequence $s(1), ...s(n)$ for observed tokens $w_1, ..., w_n$ within this model, a simple modification to the abovementioned Viterbi algorithm was designed.

3. The third approach we experimented with was previously used e.g. in [19]: we learnt an HMM *submodel* for each semantic slot having significant internal structure. HMM submodels were learnt using the unsupervised Baum-Welch algorithm [21] from the corresponding slot's data, with the desired number of states determined experimentally. Compared to the naive model, the global trigram model structure was the same, however, the learnt submodels were used in place of the original singleton target states. In Figure 3 we show an example of a 3-state submodel trained for the *bike name* slot. Only transitions with probability higher than 0.05 and the most frequent emitted words are shown. It is worth noting that the model typically learnt bike company names in state 1, bike model names in state 2, and generic properties such as colours, sizes or brakes in state 3.

The results presented in Table 1 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

[6]I.e., with lexical probabilities conditioned by *two previous words*, provided the two previous states are the same as the current state.
[7]Note that we did not need to use the EM algorithm in the naive approach, since the weights could be obtained directly from training data.

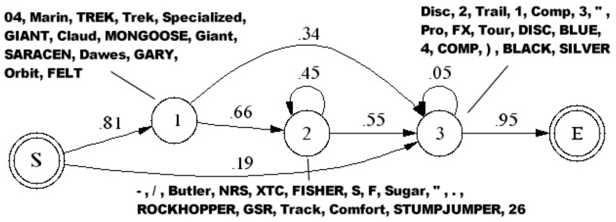

Figure 3: 3-state submodel trained for bike name

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. Some more details about the different models used can be found in the working paper [17].

2.4 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[8]. 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. [9, 10]) and could be reused.

3. STORING AND QUERYING THE RDF

3.1 RDF Schema for Bicycle Sale Domain

The HMM-based extractors discussed above are currently (in the best case) able to yield instances of *retail offers*[9],

[8]The 'presentation' ontology is correlated but neither identical nor subsumed by the RDF Schema mentioned below ('domain ontology').
[9]We use this term instead of 'bike offer', so as to cover

typically consisting of *name* of a bike, its *price*, 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* [4] ontology. In addition to information produced by the HMM, the schema also covers some information about the *company* that offers the bicycle; this information is or will soon be extracted other modules developed within the above-mentioned *Rainbow* system [22], 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 and 5 (decomposed for easier readability). 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[10]. 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 [5]. 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.

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 (especially, optional path expressions, see below).

The inference-centric character of current RDF recommendations is reflected by an inferencer in Sesame. By default, the basic set of RDFS inference rules is supported, as defined in RDF Model-Theoretic Semantics (see `http://www.w3.org/RDF`). Basic rules can be insufficient for some applications (e.g. dealing with transitive properties). For this purpose, it is possible to define custom inference rules and axiomatic triples in an external file.

separately-sold bike components.

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

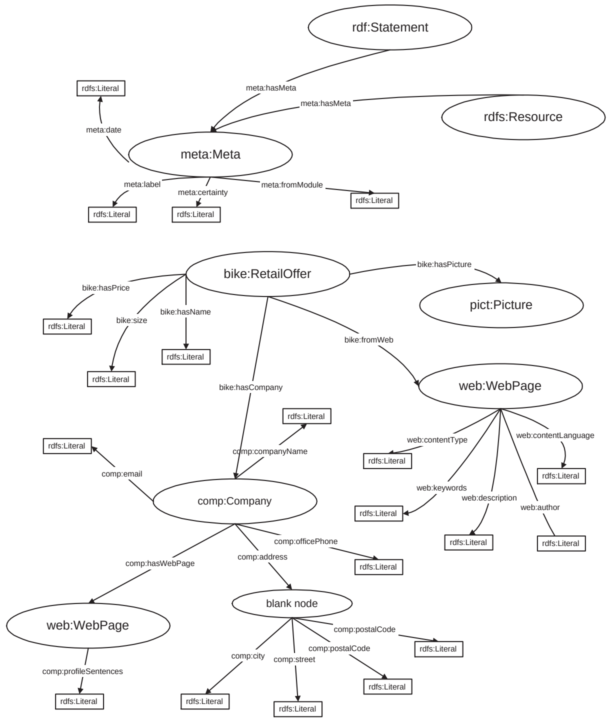

Figure 4: RDF schema of bicycle domain 1/2

Sesame also already proved scalable to larger quantities of data [14]. 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.

SeRQL [6] ("*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'[11] 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}];
      <bike:hasBikeProduct> {y},
{y} <bike:name> {name},
{x} <bike:hasCompany> {} <rdf:type> {<comp:Company>};
                              <comp:companyName> {company};
```

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

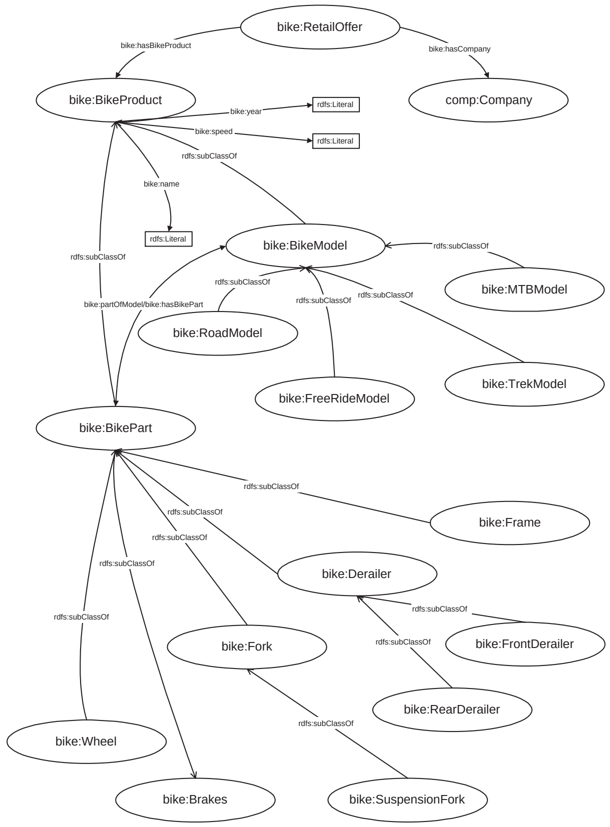

Figure 5: RDF schema of bicycle domain 2/2

```
                                 <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. 6. In the 'from' part of sample query, all its constituent triples are listed, taking advantage of *SeRQL* shortcut notation: incomplete triples following the semicolon 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* (see [24] for in-the-context comparison): there is obviously a strong need for optional items when dealing with incomplete data extracted from HTML pages.

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.

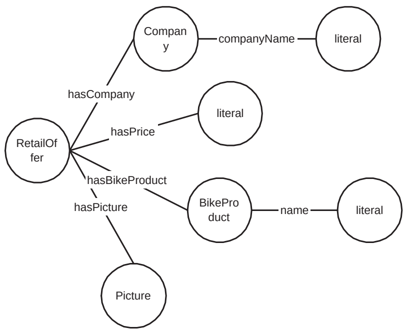

Figure 6: RDF graph for example path expression

At Fig. 7 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 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/`. Since the interface was primarily developed for query demonstration purposes, the underlying repository is currently filled with results obtained by applying automatic instance composition (section 2.4) on IE training data rather than on the direct output of HMM-based annotation. In this way, we obtained a reasonably large and consistent fact base (currently, 838 instances from 88 pages). Version with fully automatically obtained data (which are obviously sparser and less reliable) will be made available soon.

4. RELATED WORK

As mentioned above, an advanced project dealing with product information extraction is *CROSSMARC* [20]. It focuses on multi-linguality, and hence is more NLP-oriented than our current study, which only addresses English-language websites. Recently reported IE tools for semantic web are *S-CREAM* [11] and *MnM* [25]. They pay significant attention to efficient coupling of training data mark-up and subsequent automated extraction of new data. *Armadillo* [9] is probably the most advanced information extraction tool explicitly addressing the semantic web standards such as RDF (using the AKT triple store [1]). Its strong point is bootstrapping, which minimises the human annotation effort.

Since our project is more-or-less at its beginning, we cannot claim to overcome (or even match) the mentioned projects in terms of performance of IE tools involved. Rather, we attempt to bring new views on pipelining IE to subsequent *end-user retrieval* of extracted results. We also focus on

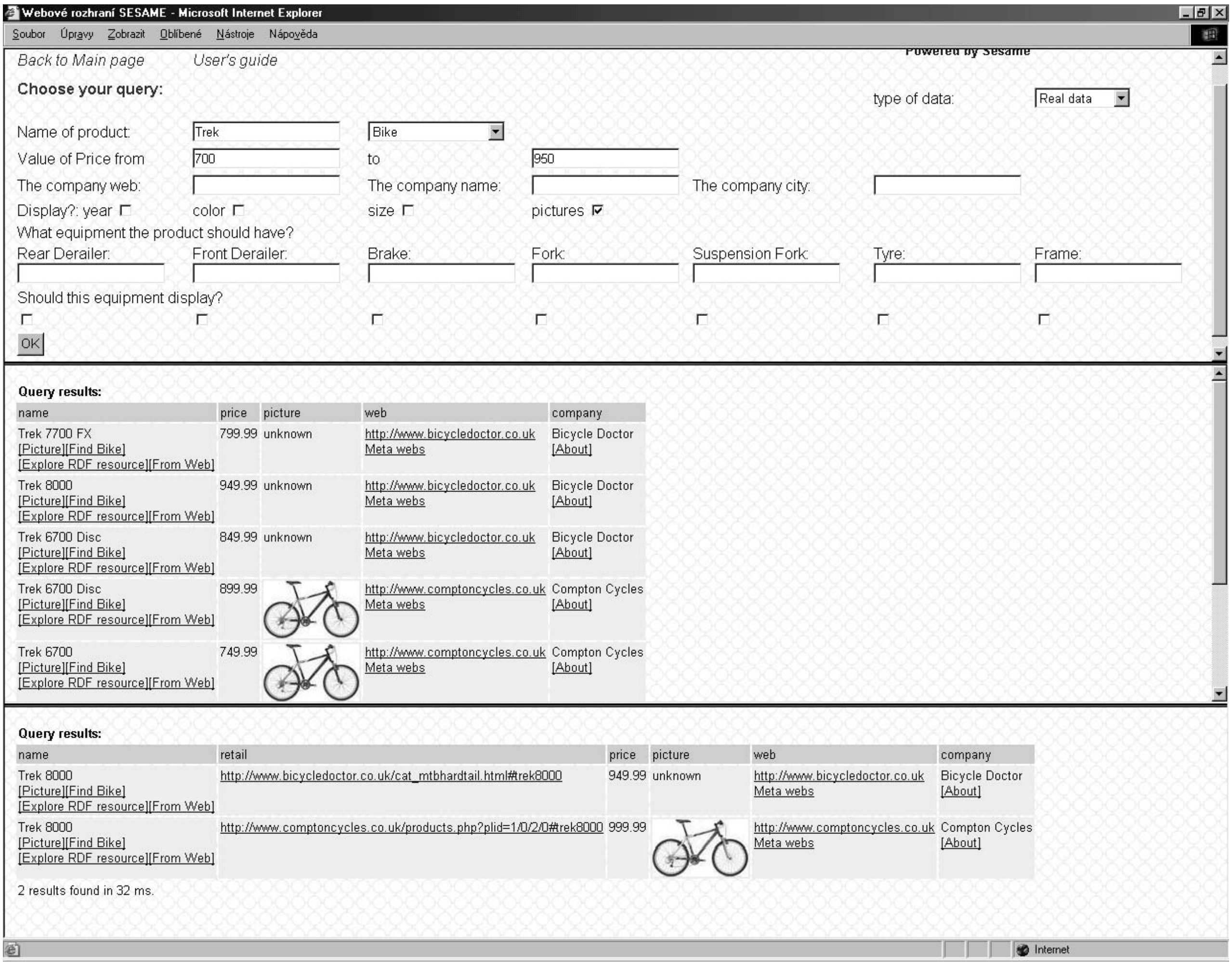

Figure 7: HTML interface to bike-offer RDF repository

company websites, which are not frequently targetted by academic IE research; 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 semantic-web IE applications. CROSSMARC is a rare exception, it however does not seem to pay particular attention to presentation of extracted results in semantic web format. While most other semantic-web IE approaches focus on rule-based methods, we attempt to fine-tune the 'alternative' *HMM paradigm* (previously proven succesful for many IE and speech recognition settings) to fit to the problem of product catalogue extraction. Finally, although *IE bootstrapping* (topical for Armadillo) is not mentioned in the current paper, it was addressed to some extent in a collateral project [13] (for general business websites), the results of which are now also being integrated with the current work.

5. CONCLUSIONS AND FUTURE WORK

We presented an application of information extraction (IE) from web product catalogues, followed with storage and retrieval of extracted results in RDF format. The IE engine currently used is based on multiple variants of statistical (Hidden Markov) models, and on a simple composition algorithm. The query interface has an underlying *Sesame* repository and comprises domain-specific query templates in *SeRQL* language, allowing for navigational querying.

The most urgent further work from the IE viewpoint regards enhancement of the *instance composition* method. We are also going to *combine* the results of product-catalogue IE with results obtained by other web-analysis methods developed in the *Rainbow* project: a control application is currently being design, which calls different tools (as web services) and integrates their results into the same RDF repository. In the more distant future, we would like to proceed to on-the-fly *knowledge-based integration*, which would take full advantage of the flexibility of RDF. We would also like to compare the performance of statistical IE methods with *rule-based* ones (such as LP^2 [8]) on the product catalogue domain. Another problem is that of *portability* to another (retail-sale) domain: apart of re-training the extraction model, the upper level of IE has to be modified as well, ideally, with maximal involvement of *domain ontologies* (provided they exist). The problem of (possibly semi-automatic) transformation between domain ontologies and *presentation ontologies* is worth investigating. Finally, we plan to provide support for on-the-fly *application construc-*

tion from available web services and its *user-controlled execution* (similar to that of Armadillo [9]), taking as starting point the *conceptual framework* for web analysis introduced in [23].

6. ACKNOWLEDGEMENTS

The authors thank Jeen Broekstra and Martin Kavalec for assistance in setting up the *Sesame* repository. The research is partially supported by grant no.201/03/1318 of the Czech Science Foundation, "Intelligent analysis of the WWW content and structure."

7. REFERENCES

[1] AKT Triplestore, `http://triplestore.aktors.org`

[2] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. 2^{nd}International Workshop on the Semantic Web, in conjunction with WWW10, Hongkong, 2001.

[3] V. Borkar, K. Deshmukh, S. Sarawagi. Automatic segmentation of text into structured records. *SIGMOD Conference*, 2001.

[4] D. Brickley, R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, World-Wide Web Consortium, Feb. 2004

[5] J. Broekstra, M. Ehrig, P. Haase, F. van Harmelen, A. Kampman, M. Sabou, R. Siebes, S. Staab, H. Stuckenschmidt, C. Tempich. 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.

[6] J. Broekstra, A. Kampman. User Guide for Sesame. `http://sesame.aidministrator.nl/publications/users/`

[7] J. Broekstra, A. Kampman, F. van Harmelen. Sesame: An Architecture for Storing and Querying RDF and RDF Schema. In: Proceedings of the First International Semantic Web Conference (ISWC 2002), Sardinia, Italy, June 9-12 2002, 54-68. Springer-Verlag Lecture Notes in Computer Science (LNCS) no. 2342.

[8] F. Ciravegna. LP^2 – an Adaptive Algorithm for Information Extraction from Web-related Texts. Proc. of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Conference on Artificial Intelligence (IJCAI-01), August, 2001.

[9] F. Ciravegna, S. Chapman, A. Dingli, Y. Wilks: Learning to Harvest Information for the Semantic Web. In: Proceedings of the 1st European Semantic Web Symposium (ESWS-04), Heraklion, Greece, 2004.

[10] D. W. Embley, C. Tao, S.W. Liddle. Automatically Extracting Ontologically Specified Data from HTML Tables of Unknown Structure. ER 2002: 322-337.

[11] S. Handschuh, S. Staab, F. Ciravegna. S-CREAM – Semi-automatic CREAtion of Metadata. In: Proceedings EKAW-02, Springer Verlag 2002.

[12] F. Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, Cambridge, Massachusetts, 1997.

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

[14] A. Kiryakov, B. Popov, D. Manov. Semantic Indexing and Retrieval. Proceedings of the SIGIR 2003 Semantic Web Workshop, Toronto, Canada, 2003.

[15] C.A. Knoblock, S. Minton, J.L. Ambite, N. Ashish, P.J. Modi, I. Muslea, A.G. Philpot, S.Tejada. *Modeling Web Sources for Information Integration.* In: Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98), Madison, WI, 1998.

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

[17] M. Labský, V. Svátek. Information Extraction from Web Product Catalogues. Working paper, 2004, online at `http://rainbow.vse.cz/hmm-working.pdf`.

[18] J. Mayfield, T. Finin. Information Retrieval on the Semantic Web: Integrating inference and retrieval. Proceedings of the SIGIR 2003 Semantic Web Workshop, Toronto, Canada, 2003.

[19] A. McCallum, D. Freitag. Information extraction with HMM structures learned by stochastic optimization. *Proceedings of the 17-th National Conference on Artificial Intelligence*, 2000.

[20] M.T. Pazienza, A. Stellato, M. Vindigni. Combining ontological knowledge and wrapper induction techniques into an e-retail system. Proceedings of the International Workshop on Adaptive Text Extraction and Mining held in conjunction with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22, 2003.

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

[22] V. Svátek, J. Kosek, M. Labský, J. Bráza, M. Kavalec, M. Vacura, V. Vávra, V. Snášel. Rainbow - Multiway Semantic Semantic Analysis of Websites. In: 2^{nd} DEXA Int'l Workshop on Web Semantics, Prague, IEEE Computer Society Press 2003.

[23] V. Svátek, M. Labský, M. Vacura. Knowledge Modelling for Deductive Web Mining. Accepted for EKAW 2004.

[24] O. Šváb, V. Svátek, M. Kavalec, M. Labský. 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`.

[25] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. The 13th International Conference on Knowledge Engineering and Management (EKAW 2002), ed Gomez-Perez, A., Springer Verlag, 2002.