The *Ex* Project: Web Information Extraction using Extraction Ontologies

Martin Labský, Vojtěch Svátek, Marek Nekvasil and Dušan Rak

Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
e-mail: {labsky,svatek,nekvasim,rakdusan}@vse.cz

Abstract. Extraction ontologies represent a novel paradigm in web information extraction (as one of 'deductive' species of web mining) allowing to swiftly proceed from initial domain modelling to running a functional prototype, without the necessity of collecting and labelling large amounts of training examples. Bottlenecks in this approach are however the tedium of developing an extraction ontology adequately covering the semantic scope of web data to be processed and the difficulty of combining the ontology-based approach with inductive or wrapper-based approaches. We report on an ongoing project aiming at developing a web information extraction tool based on richly-structured extraction ontologies and with additional possibility of (1) semi-automatically constructing these from third-party domain ontologies, (2) absorbing the results of inductive learning for subtasks where pre-labelled data abound, and (3) actively exploiting formatting regularities in the wrapper style.

1 Introduction

Web information extraction (WIE) represents a specific category of web mining. It consists in the identification of typically small pieces of relevant text within web pages and their aggregation into larger structures such as data records or instances of ontology classes. As its core task is application of pre-existent patterns or models (in contrast to inductively discovering new patterns), it falls under the notion of 'deductive' web mining [10], similarly as e.g. web document classification. As such, some kind of prior knowledge is indispensable in WIE. However, the 'deductive' aspects of WIE are often complemented with inductive ones, especially in terms of learning the patterns/models (at least partly) from training data.

In the last decade, WIE was actually dominated by two paradigms. One—*wrapper*-based—consists in systematically exploiting the surface structure of HTML code, assuming the presence of regular structures that can be used as anchors for the extraction. This approach is now widely adopted in industry, however, its dependence on formatting regularity limits its use for diverse categories of web pages. The other—*inductive*—paradigm assumes the presence of training data: either web pages containing pre-annotated tokens or stand-alone examples of data instances. It is linked to exploration of various computational learning paradigms, e.g. Hidden-Markov Models, Maximum Entropy Models, Conditional Random Fields [7] or symbolic approaches such as rule learning [1]. Again, however, the presence of (sufficient amounts of) annotated training data is a pre-condition that is rarely fulfilled in real-world settings, and manual labelling of training data is often unfeasible; statistical bootstrapping alleviates this problem to some degree but at the same time burdens the whole process with 'heavy computational machinery', whose requirements and side-effects are not transparent to a casual user of a WIE tool. In addition, both approaches usually deliver extracted information as rather weakly semantically structured; if WIE is to be used to fuel semantic web repositories, secondary mapping to ontologies is typically needed, which makes the process complicated and possibly error-prone.

There were recently proposals for pushing ontologies towards the actual extraction process as immediate prior knowledge. *Extraction ontologies* [3] define the concepts, the instances of which are to be extracted, in the sense of various attributes, their allowed values as well as higher level (e.g. cardinality or mutual dependency) constraints. Extraction ontologies are assumed to be hand-crafted based on observation of a sample of resources; however, due to their clean and rich conceptual structure (allowing partial intra-domain reuse and providing immediate semantics to extracted data), they are superior to ad-hoc hand-crafted patterns used in early times of WIE. At the same time, they allow for rapid start of the actual extraction process, as even a very simple extraction ontology (designed by a competent person) is likely to cover a sensible part of target data and generate meaningful feedback for its own redesign; several iterations are of course needed to obtain results in sufficient quality. It seems that for web domains that consist of a high number of relatively tiny and evolving resources (such as web product catalogs), information extraction ontologies are the first choice. However, to make maximal use of available data and knowledge and avoid overfitting to a few data resources examined by the designer, the whole process must not neglect available labelled data, formatting regularities and even pre-existing domain ontologies.

In this paper we report on an ongoing effort in building a WIE tool named *Ex*, which would synergistically exploit all the mentioned resources, with central role of extraction ontologies. Section 2 explains the structure of extraction ontologies used in *Ex*. Section 3 describes the steps of the information extraction process. Section 4 briefly reports on experiments in two different domains. Finally, section 5 surveys related research, and section 6 outlines future work.

2 Ex(traction) ontology content

Extraction ontologies in *Ex* are designed so as to extract occurrences of *attributes* (such as 'age' or 'surname'), i.e. standalone named entities or values, and occurrences of whole *instances* of *classes* (such as 'person'), as groups of attributes that 'belong together', from HTML pages (or texts in general) in a domain of interest.

2.1 Attribute-related information

Mandatory information to be specified for each attribute is: name, data type (string, long text, integer, float) and dimensionality (e.g. 2 for screen resolution like 800x600). In order to automatically extract an attribute, additional knowledge is typically needed.

Extraction knowledge about the attribute *content* includes (1) textual value patterns; (2) for integer and float types: min/max values, a numeric value distribution and possibly units of measure; (3) value length in tokens: min/max length constraints or a length distribution; (4) axioms expressing more complex constraints on the value and (5) coreference resolution knowledge. Attribute *context* knowledge includes (1) textual context patterns and (2) formatting constraints.

Textual patterns in *Ex* (for both the value and the context of an attribute) are regular patterns primarily defined at the level of words (tokens). They may be inlined in the extraction ontology or as (possibly large) external files, and may include the following:

- specific tokens, e.g. 'employed by'
- token *wildcards*, which require one or more token properties to have certain values (e.g. any capital or uppercase token, any token whose lemma is 'employ')
- *character-level* regular expressions for individual tokens
- *references* to other matched *attribute candidates*: a *value* pattern containing a reference to another attribute means that it can be nested inside this attribute's value; for *context* patterns, attribute references help encode how attributes follow each other
- *references* to other *matched patterns*; this allows for construction of complex grammars where rules can be structured and reused
- *references* to *named entities* provided by other systems: these could include part-of-speech tags, parsed chunks or output from other IE/NER systems[1]

For *numeric* types, default value patterns for integer/float numbers are provided. Tabular, uniform, normal and mixture distributions are available to model attribute values. Linking a numeric attribute to unit definitions (e.g. to various currency units) will automatically create value patterns containing the numeric value surrounded by the units. In case there are multiple convertible units the extraction knowledge is reused.

For both attribute and class definitions, *axioms* can be specified that impose constraints on attribute value(s). For a single attribute, the axiom checks the to-be-extracted value and is either satisfied or not (which may boost or suppress the attribute candidate's score). For a class, each axiom may refer to all attribute values present in the partially or fully parsed instance. For example, a price with tax must be greater than the price without tax. Axioms can be authored using the JavaScript[2] scripting language. We chose JavaScript since it allows arbitrarily complex axioms to be constructed and also because the web community is used to it.

In addition, *formatting constraints* may be provided for each attribute. Currently, four types of formatting constraints are supported: (1) the whole attribute value is contained in a single parent, i.e. it does not include other tags or their boundaries; (2) the value fits into the parent; (3) the value does not cross any inline formatting elements; (4) it does not cross any block elements. We investigate how custom constraints could easily be added by users. By default, all four constraints are in effect and influence the likelihood of attribute candidates being extracted.

[1] So far we experimented with lemmatizers and POS taggers.
[2] http://www.mozilla.org/rhino

2.2 Class-related information

Each *class definition* enumerates the attributes which may belong to it, and for each attribute it defines a *cardinality* range. Extraction knowledge may address both content and context of the class. *Class content patterns* are analogous to the attribute value patterns, however, they may match *parts* of an instance and must contain at least one *reference* to a member attribute. Class content patterns may be used e.g. to describe common wordings used between attributes or just to specify attribute ordering. *Axioms* are used to constrain or boost instances based on whether their attributes satisfy the axiom. For each attribute, an *engagedness* parameter may be specified to estimate the apriori probability of the attribute joining a class instance (as opposed to standalone occurrence). Regarding class context, analogous *class context patterns* and similar *formatting constraints* as for attributes are in effect also for classes. An excerpt from an extraction ontology about computer monitor descriptions is shown in Fig. 1.

2.3 Extraction evidence parameters

All types of extraction knowledge mentioned above, i.e. value and context patterns, axioms, formatting constraints and ranges or distributions for numeric attribute values and for attribute content lengths, are essentially pieces of evidence indicating the presence (or absence) of a certain attribute or class instance. In *Ex*, every piece of evidence may be equipped with two probability estimates: precision and recall. The *precision* of evidence states how probable it is for the predicted attribute or class instance to occur given the evidence holds, disregarding the truth values of other evidence. For example, the precision of a left context pattern “person name: $” (where $ denotes the predicted attribute value) may be estimated as 0.8; i.e. in 80% of cases we expect a person name to follow in text after a match of the “person name:” string. The *recall* of evidence states how abundant the evidence is among the predicted objects, disregarding whether other evidence holds. For example, the ”person name: $” pattern could have a low recall since there are many other contexts in which a person name could occur.

Pattern precision and recall can be estimated in two ways. First, annotated documents can be used to estimate both parameters using simple ratios of counts observed in text. In this case, it is necessary to smooth the parameters using an appropriate method. For a number of domains it is possible to find existing annotated data, e.g. web portals often make available online catalogs of manually populated product descriptions linking to the original sellers’ web pages. When no training data is available or if the evidence seems easy to estimate, the user can specify both parameters manually. For the experimental results reported below we estimated parameters manually.

3 The extraction process

The inputs to the extraction process are the extraction ontology and a set of documents. Extraction consists of six stages depicted in Fig. 2.

```
<class id="Monitor">
 <pattern id="name_price_order" cover="0.8">
   $name (<tok/>{0,20} $price){1,4}
 </pattern>
 <axiom cover="1"> $price_with_tax &gt; $price_wo_tax </axiom>
 <attribute id="name" type="name" card="1" eng="0.70">
  <value>
   <pattern cover="0.5" p="0.8" ignore="case">
    (LCD (monitor|panel)?)? <pattern src="manuf.txt" ign="case"/>
    (<tok type="ALPHANUM|ALPHA|INT"/>|<tok case="UC"/>){1,2}
   </pattern>
   <length> <distribution min="1" max="7"/> </length>
   <pattern cover="0.5" type="fmt"> fits_in_parent </pattern>
   <pattern cover="1.0" type="fmt"> no_cross_blocks </pattern>
  </value>
 </attribute>
```

Fig. 1. Sample extraction ontology for computer monitors

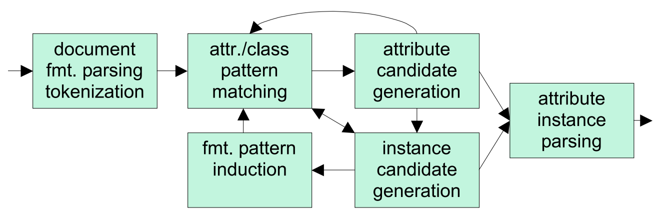

Fig. 2. Extraction process schema

3.1 Document preprocessing

First, the analysed document is loaded and its formatting structure is read into a simplified DOM tree of formatting objects. To robustly read web pages containing invalid HTML we employ the CyberNeko HTML parser[3]. Any text found in the formatting elements and their attributes is tokenized using a configurable tokenizer. A flat array of tokens is created for the document with each token linking to its parent formatting element. As part of tokenization, new words are registered in a common vocabulary, lemmatized and linked to their lemmas (if available), and classified by token type (e.g. alphanumeric) and case (e.g. capital).

3.2 Attribute candidate generation

After loading the document, all attribute value and attribute context patterns of the ontology are matched against the document's tokens. Where a value pattern matches, the

[3] `http://people.apache.org/~andyc/neko/doc/html/`

system attempts to create a new candidate for the associated attribute (attribute candidate – AC). If more value patterns match at the same place, or if there are context pattern matches for this attribute in neighbouring areas, then the corresponding evidence is turned on as well for the AC. Evidence corresponding to all other non-matched patterns is kept off for the AC. Also, during the creation of the AC, all other evidence types (axioms, formatting constraints, content length and numeric value ranges) are evaluated and set. The set of all evidence Φ_A known for the attribute A is used to compute a *conditional probability estimate* P_{AC} of how likely the AC is given all the observed evidence values:

$$P_{AC} = P(A|E \in \Phi_A) \tag{1}$$

The full formula is described and derived in [6]. We assume conditional independence of evidence given that the attribute holds or not. The AC is created only if P_{AC} exceeds a pruning threshold defined by the extraction ontology.

In places where a context pattern matches and there are no value pattern matches in neighbourhood, the system tries to create ACs of various length (in tokens) in the area pointed to by the context pattern. For patterns which include other attributes, we run the above process until no new ACs are generated.

The set of (possibly overlapping) ACs created during this phase is represented as an *AC lattice* going through the document, where each AC is scored by $score(AC) = log(P_{AC})$. Apart from the ACs which may span multiple tokens, the lattice also includes one 'background' state for each token that takes part in some AC. A background state BG_w for token w is scored as follows:

$$score(BG_w) = \min_{AC, w \in AC} log(\frac{1 - P(AC)}{|AC|}) \tag{2}$$

where $|AC|$ is the length of the AC in tokens. The extraction process can terminate here if no instance generation or formatting pattern induction is done, in which case all ACs on the best path through the lattice are extracted.

3.3 Instance candidate generation

At the beginning of the instance candidate (IC) generation phase, each AC is used to create a simple IC consisting just of that single AC. Then, a bottom-up IC generation algorithm is employed to generate increasingly complex ICs from the working set of ICs. At each step, the highest scoring (seed) IC is chosen and its neighbourhood is searched for ACs that could be added to it without breaking ontological constraints for the IC class. Only a *subset* of the constraints is taken into account at this time as e.g. some minimum cardinality constraints or axioms could never get satisfied initially. Each added AC is also examined to see whether it may corefer with some AC that is already present in the IC; if yes, it is only added as a reference and it does not affect the resulting IC score. To detect coreferences, the extraction ontology author may specify for each attribute a binary comparison function that compares two attribute values to determine whether they corefer (by default the values must equal to corefer).

After adding ACs to the chosen seed IC, that IC is removed from the working set and the newly created larger ICs are added to it. The seed IC is added to a *valid IC set* if it satisfies *all* ontological constraints. As more complex ICs are created by combining simpler ICs with surrounding ACs, a limited number of ACs is allowed to be skipped (AC_{skip}) between the combined components, leading to a penalization of the created IC. The IC scores are computed based on their AC content and based on the observed values of evidence E known for the IC class C:

$$sc_1(IC) = exp(\frac{\sum_{AC \in IC} log(P_{AC}) + \sum_{AC_{skip} \in IC}(1 - log(P_{AC_{skip}}))}{|IC|}) \quad (3)$$

$$sc_2(IC) = P(C|E \in \Omega_C) \quad (4)$$

where $|IC|$ is the number of member ACs and Ω_C is the set of evidence known for class C; the conditional probability is estimated as in Eq. 1. By experiment we chose the Prospector [2] pseudo-bayesian method to combine the above into the final IC score:

$$score(IC) = \frac{sc_1(IC)sc_2(IC)}{sc_1(IC)sc_2(IC) + (1 - sc_1(IC))(1 - sc_2(IC))} \quad (5)$$

The IC generation algorithm picks the best IC to expand using the highest $score(IC)$. The generation phase ends when the working set of ICs becomes empty or on some terminating condition such as after a certain number of iterations or after a time limit has elapsed. The output of this phase is the set of valid ICs.

3.4 Formatting pattern induction

During the IC generation process, it may happen that a significant part of the created valid ICs satisfies some (apriori unknown) *formatting pattern*. For example, a contact page may consist of 6 paragraphs where each paragraph starts with a bold person name together with scientific degrees. A more obvious example would be a table with the first two columns listing staff first names and surnames. Then, if e.g. 90 person names are identified in such table columns and the table has 100 rows, the induced patterns make the remaining 10 entries more likely to get extracted as well.

Based on the lattice of valid ICs, the following *pattern induction* procedure is performed. First, the best scoring sequence of non-overlapping ICs is found through the lattice. Only the ICs on the best path take part in pattern induction. For each IC, we find its nearest containing formatting block element. We then create a subtree of formatting (incl. inline) elements between the containing block element (inclusive) and the attributes comprising the IC. This subtree contains the names of the formatting elements (e.g. paragraph or bold text) and their order within parent (e.g. the first or second cell in table row). Relative frequencies of these subtrees are calculated over the examined IC set (separately for each class if there are more). If the relative and absolute frequencies of a certain subtree exceed respective configurable thresholds, a new formatting pattern is induced and the subtree is transformed into a new *context pattern* indicating the presence of the corresponding class. This induced formatting context pattern is an example of 'local' evidence only useful within the currently analysed document (or a set of documents coming from the same source). The precision and recall of the induced context patterns are based on the relative frequencies with which the patterns hold in the document (or document set) with respect to the observed ICs.

The newly created context patterns are then fed back to the pattern matching phase, where they are matched and applied. This extra iteration rescores existing ACs and ICs and may as well yield new ACs and ICs which would not have been created otherwise. With our current implementation we have so far only experimented with pattern induction for ICs composed of a single attribute. Using this feature typically increases recall but may have adverse impact on precision. One approach to avoid degradation of precision is to provide attribute evidence which will prevent unwanted attributes from being extracted.

3.5 Attribute and instance parsing

The purpose of this final phase is to output the most probable sequence of instances and standalone attributes through the analysed document. The valid ICs are merged into the AC lattice so that each IC can be avoided by taking a path through standalone ACs or through background states. In the lattice, each IC is scored as $score(IC)|IC|$. This lattice is searched for *n* best sequences of non-overlapping extractable objects and these sequences are finally output. Consequently, the best path through the document may contain both instances and standalone attributes.

3.6 Incorporating third party tools

In practical WIE tasks it often happens that some of the attributes of interest are relatively easy to extract using manually specified evidence, some require *machine learning* algorithms such as CRFs [7] in order to achieve good extraction results, and some may benefit from a combination of both. To support all three cases, *Ex* allows named entity candidates identified by other engines to be included in all types of textual patterns described above. For example, suppose our task is to extract instances of a Person class composed of a person name and a scientific degree. Let's also suppose we have training data for person names but no data for degrees. A viable approach would then be to train e.g. a CRF classifier to identify person names in text and to specify evidence for degrees manually. To incorporate the CRF classifier's suggestions into the extraction ontology, a simple *attribute value pattern* like "\${crf:personname}" can be added to the person name attribute. Here, \${} denotes a reference to an external named entity, *crf* is the source component name and *personname* is the identifier output by the CRF classifier. The precision for this value pattern can either be derived from the CRF classifier confidence score, or we can use the expected precision of the classifier for this attribute. To estimate the recall of the pattern, we can use the expected recall achieved by the classifier. Additionally to this pattern, the user may specify more patterns to correct (limit or extend) the classifier's suggestions.

4 Experimental Results

4.1 Contact Information on Medical Pages

In the EU (DG SANCO) MedIEQ project[4] we experiment with several dozens of medical website quality criteria, most of which are to be evaluated with the assistance of IE

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

tools. One of them is the presence and richness of contact information. Table 1 shows early results for contact IE. The data set consists of 109 HTML documents, which were all manually classified as contact pages (each coming from a different website); in total there are 146 HTML files as some documents include frames or iframes. The documents contain 6930 annotated named entities of 10 types. The contact extraction ontology was written based on seeing the first 30 documents of the total data; it also refers to gazetteers such as lists of city names, common first names and surnames. The ontology contains about 100 textual patterns for the context and content of attributes and of the single extracted 'contact' class, attribute length distributions and several axioms. The effort spent on developing and tuning the ontology was about 2-3 person-weeks. In the strict mode of evaluation, only exact matches are considered to be successfully extracted. In the loose mode, partial credit is given to incomplete or overflown matches; e.g. extracting 'John Newman' where 'John Newman Jr.' was supposed to be extracted will count as a 66% match (based on overlapping word counts). The performance is probably underestimated since the reliability of manual annotation was very low: the inter-annotator agreement between the 3 human annotators was only 73.2% on average, and e.g. for person names it only reached 68.7%. We are working to fix these inconsistencies. Fig. 3 shows sample automatically annotated data.

Table 1. Contact IE results

	strict mode			loose mode		
attribute	prec	recall	F	prec	recall	F
title	0.71	0.82	0.76	0.78	0.86	0.82
name	0.66	0.51	0.58	0.74	0.56	0.64
street	0.62	0.52	0.56	0.85	0.67	0.75
city	0.47	0.73	0.57	0.48	0.76	0.59
zip	0.59	0.78	0.67	0.67	0.85	0.75
country	0.58	0.89	0.70	0.59	0.89	0.71
phone	0.97	0.84	0.90	0.99	0.87	0.93
email	1.00	0.99	1.00	1.00	0.99	1.00
company	0.57	0.37	0.44	0.81	0.51	0.63
dept.	0.51	0.31	0.38	0.85	0.45	0.59
overall	0.70	0.62	0.66	0.78	0.68	0.72

4.2 Weather Forecasts

Finally, we experimented with the domain of weather forecasts. Here our goal was to investigate the possibility to assist the ontology engineer in reusing existing *domain ontologies* in order to develop the extraction one/s. An advantage of this domain was the fact that several OWL ontologies were available for it. We analysed three of them by means of applying generic rules of two kinds:

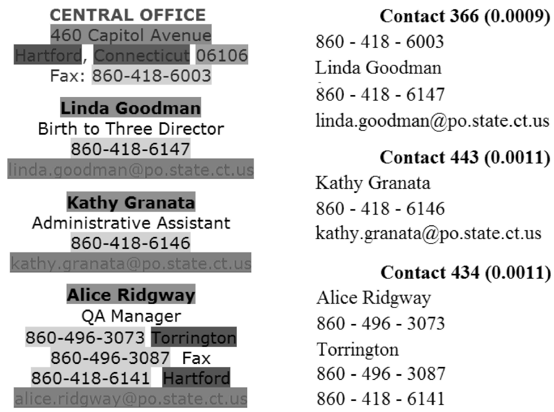

Fig. 3. Sample automatically annotated data; extracted instances on the right.

1. Rules suggesting the *core class/es* for the extraction ontology. As the extraction ontology for extraction from HTML-formatted text[5] is typically more class-centric and hierarchical than a properly-designed domain ontology, only few classes from the domain ontology are likely to become classes in the extraction ontology, while others become attributes that are dependent on the core class/es. For example, 'Day' is typically an attribute of a 'Forecast' class in an extraction ontology, while in the domain ontology they could easily be two classes connected by a relationship. One of such core class selection rules is, in verbal form, e.g. "Classes that appear more often in the domain than in the range of object properties are candidates for core class/es.".
2. Rules performing the actual *transformation*. Examples of such rules are e.g. "A data type property D of class C may directly yield an attribute of C." or "A set of mutually disjoint subclasses of class C may yield an attribute, whose values are these subclasses."

Most such independently formulated selection and transformation rules appeared as performing well in the initial experiment in the weather domain; details are in [5]. Transformation rules seemed, by first judgement, to suggest a sensible and inspiring, though by far not complete, skeleton of an extraction ontology. Testing this ontology on real weather forecast records is however needed for proper assessment.

In general, although the first experiments look promising, extensive usage of domain ontologies as starting point for extraction ontologies seems to be hindered by unavailability of high-quality domain ontologies for most domains, e.g. in relation to different categories of products or services, judging by the results of Swoogle-based[6] retrieval. This obstacle is likely to disappear in the not-so-distant future, as the semantic web technology becomes more widespread.

[5] This is not the case for extraction from free text, which is more relation-centric.

[6] `http://swoogle.umbc.edu`

5 Related Work

Most state-of-the-art WIE approaches focus on identifying structured collections of items (records), typically using inductively learnt models. Ontologies are often considered but rather as additional structures to which the extracted data are to be adapted after they have been acquired from the source documents, for the sake of a follow-up application [4]. There is no provision for directly using the rich structure of a domain-specific ontology in order to guide the extraction process. The approach to WIE that is inherently similar to ours (and from which we actually got inspiration in the early phase of our research) is that developed by Embley and colleagues at BYU [3]. The main distinctive features of our approach are: (1) the possibility to provide the extraction patterns with probability estimates (plus other quantitative info such as value distributions), allowing to calculate the weight for every attribute candidate as well as instance candidate; (2) the effort to combine hand-crafted extraction ontologies with other sources of information—HTML formatting and/or known data instances (3) the pragmatic distinction between extraction ontologies and domain ontologies proper: extraction ontologies can be arbitrarily adapted to the way domain data are typically *presented* on the web while domain ontologies address the domain as it is (but can be used as starting point for designing extraction ontologies). For similarly pragmatic reasons (easy authoring), we also used a proprietary XML syntax for extraction ontologies. An objective comparison between both approaches would require detailed experiments on a shared reference collection.

An approach to automatically discover new extractable attributes from large amounts of documents using statistical and NLP methods is described in [8]. On the other hand, formatting information is heavily exploited for IE from tables in [11]. Our system has a slightly different target; it should allow for fast IE prototyping even in domains where there are few documents available and the content is semi-structured. While our system relies on the author to supply coreference resolution knowledge for attribute values, advanced automatic methods are described e.g. in [13]. A system described in [12] uses statistical methods to estimate the mutual affinity of attribute values.

Our ideas and experiments on domain ontology selection and transformation to extraction ontology are related to the generic research in ontology selection [9] and content evaluation[7], especially with respect to the notion of intra-ontology concept centrality; this relationship deserves further study.

6 Conclusions

The *Ex* system attempts to unify the often separate phases of WIE and ontology population. Multiple sources of extraction knowledge can be combined: manually encoded knowledge, knowledge acquired from annotated data, and knowledge induced from common formatting patterns by the means of wrapper induction. An alpha version of *Ex* (incl. extraction ontology samples) is publicly available[8].

[7] http://km.aifb.uni-karlsruhe.de/ws/eon2006/
[8] http://eso.vse.cz/~labsky/ex

Future work will concentrate on the integration with trainable machine learning algorithms, on improving the extraction results and on covering more domains. We currently experiment with IE from online product descriptions, where we develop an extraction ontology for each type of examined product. Typically extracted attributes include product name, price, picture and multiple product-specific attributes. In order to obtain annotated data, we cooperate with one of the largest Czech web portals. Both instance parsing and formatting pattern induction algorithms need improvement in accuracy and speed. We also plan to investigate how text mining over the extraction results could help us identify 'gaps' in the ontology, e.g. non-labelled tokens frequently appearing inside a 'cloud' of annotations are likely to be unrecognised important values. Finally, we intend to provide support for semi-automated transformation of domain ontologies to extraction ones.

The research was partially supported by the EC under contract FP6-027026, Knowledge Space of Semantic Inference for Automatic Annotation and Retrieval of Multimedia Content - K-Space. The medical website application is carried out in the context of the EC-funded (DG-SANCO) project MedIEQ.

References

1. Ciravegna, F.: LP2 – an adaptive algorithm for information extraction from web-related texts. In: Proc IJCAI-2001.
2. Duda, R.O., Gasching, J., and Hart, P.E. Model design in the Prospector consultant system for mineral exploration. In: Readings in Artificial Intelligence, pp. 334–348, 1981.
3. Embley, D. W., Tao, C., Liddle, D. W.: Automatically extracting ontologically specified data from HTML tables of unknown structure. In *Proc. ER '02*, pp. 322–337, London, UK, 2002.
4. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. In: *J. Web Sem.*, volume 2, pp. 49–79, 2004.
5. Labsky, M., Nekvasil, M., Svatek, V.: Towards Web Information Extraction using Extraction Ontologies, Domain Ontologies, and Inductive Learning. Accepted as poster paper for Proc. of K-CAP 2007, Whistler, Canada, ACM 2007.
6. Labsky, M., Svatek, V: Information extraction with presentation ontologies. Technical report, KEG UEP, `http://eso.vse.cz/~labsky/ex/ex.pdf`.
7. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. 18th International Conf. on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco, CA, 2001.
8. Popescu, A., Etzioni, O.: Extracting Product Features and Opinions from Reviews. In: Proc. EMNLP 2005.
9. Sabou, M., Lopez, V., Motta, E.: Ontology selection for the real semantic web: How to cover the queen's birthday dinner? In: Proc. EKAW 2006. Springer LNCS, 2006.
10. Svatek, V., Labsky, M., Vacura, M.: Knowledge Modelling for Deductive Web Mining. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
11. Wei, X., Croft, B., McCallum, A.: Table Extraction for Answer Retrieval. In: Information Retrieval Journal, vol. 9, issue 5, pp. 589-611, 2006.
12. Wick, M., Culotta, A., McCallum, A.: Learning Field Compatibilities to Extract Database Records from Unstructured Text. In: Proc. EMNLP, 2006.
13. Yates, A., Etzioni, O.: Unsupervised Resolution of Objects and Relations on the Web. In: Proc. HLT 2007.