On the Design and Exploitation of Presentation Ontologies for Information Extraction

Martin Labský and Vojtěch Svátek

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

Abstract. The structure of ontologies that are considered as input to information extraction is mostly rather simple. In this paper we report on our ongoing effort of using rich ontologies with numerous constraints over the information to be extracted. Important aspects of the approach are the coupling of user-defined ontologies with other sources of knowledge such as training data and document formatting structures, and the distinction between proper domain ontologies and so-called presentation ontologies, where the latter (as 'pragmatic bridges' over the 'semantic gap') can partially be derived from the former. The extraction tool under construction builds on experience from an ongoing application in the domain of product catalogue analysis, and attempts to offer high flexibility with respect to availability of various input information sources.

1 Introduction

It is now frequently assumed that the use of ontologies can bring the required flexibility to the discipline of information extraction (IE). However, the nature of ontologies in this role may significantly vary. Let us recall that in a general overview of ontology types, van Heijst [16] 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 of formal axioms, most particularly, by the possibility to characterise the extent of a class via a logical expression over its properties (relations to other classes or values of attributes).

The range of models possibly appearing in different phases of IE[1] seems to be somewhat analogous to the general categorisation. First, at the level of plain text strings, *terminological ontologies* may come into play, being structured versions of gazetteers commonly used for named-entity recognition. Second, Stevenson & Ciravegna [14] raised the issue of ontologies 'for customer service' that do not

[1] This range is discussed and exemplified more thoroughly in [10].

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, and possibly even images and other multimedia objects. 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 objects as *presented* on the web or similar medium (e.g. computer offers encoded in HTML) rather than of real-world objects (real computers). They should thus reflect more surface-level relationships, incl. e.g. syntactic patterns and quantitative distributions of features, incl. non-textual ones.

In this paper we report on our ongoing effort of using presentation ontologies for IE, especially IE from web product catalogues. We see presentation ontologies as 'pragmatic bridges' over the 'semantic gap' between high-level domain ontologies and low-level features used as evidence in information extraction. Section 2 briefly describes the *Rainbow* project as context in which this effort started. Section 3 enumerates the types of content of presentation ontologies in our current approach to IE. Section 4 sketches a framework for semi-automatic creation of presentation ontology from a conventional domain ontology. Section 5 describes in more detail the principles and implementation status of our new tool currently under development. Finally, section 6 compares our approach with some related work, and section 7 wraps up the whole paper.

2 Background: The *Rainbow* Project

Rainbow started in 2001 as informal project of a group of researchers and students at the University of Economics, Prague. The goal was to develop and integrate (via web service interfaces) a diverse collection of tools for the analysis of content and structure of the WWW, in particular, the websites of smaller companies. In the first phase of the project (2001–2003) there were two main implemented outcomes: an unsupervised method for extraction of 'company profile' sentences via bootstrapping from web directory content and structure [7] and a graph-theoretical method for discovery of navigational structure of company websites. 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. 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 in the second phase were:

- An *information extraction* tool trained for extraction of bicycle product offers from online catalogues [11]; it consists of an annotator employing Hidden

Markov Models and a heuristic 'instance parser' already exploiting a rudimentary presentation ontology [10] (pre-cursor to those discussed in the next section)
- A collection of tools for *image analysis*, based on singular value decomposition (SVD), colour histograms and image dimensions [12]
- A *procedural application* that calls individual analysis tools, collects results and converts them to RDF
- A *result repository*[2] with simple HTML interface allowing for end-user *search and navigation* [15]
- 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*.

The research described in this paper (which can be viewed as part of the third phase of *Rainbow*) mainly focuses on the IE task and especially on the use of ontologies together with other available input to this task. A more extensive description of the project (dealing e.g. with its general knowledge modelling aspects) as well links to implemented tools can be found at `http://rainbow.vse.cz`.

3 Structure of Presentation Ontologies

Purely syntactically, the concept of presentation ontology extends that of domain ontology by coupling its *classes* and their *attributes* (or datatype properties in Semantic Web terminology) with IE "hooks", which are described in the following subsections for attributes and whole class instances.

A presentation ontology will typically only contain few classes, in fact a single class will suffice for most of our experiments. Instances of different classes (with their numerous attributes) will often be extracted separately, and their integration will be done beyond the text analysis phase. For example, from a company website we can extract contact details as well as information about its products, and instances of an 'offered-by' relation can be added at a higher level of processing.

3.1 Attribute extraction knowledge

For IE purposes, attributes are equipped with the following metadata that relate to extraction of their values[3]:

- *data type* which can be either `string`, `text`[4], `int`, `float`, `boolean` or `image`,

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

[3] Throughout the paper, we will often use the term 'attribute' to denote the attribute *value* to be extracted, for brevity.

[4] Both `string` and `text` datatypes can have textual values and may contain other embedded attributes. The `text` datatype captures longer free-form texts (such as textual product descriptions) and allows for inclusion of intermediate text and formatting.

– *context patterns* define the attribute's typical surrounding tokens. For the case of structured documents, the notion of context can be extended to the attribute's position in the document. For HTML documents, this could be a generalised DOM path to the attribute.

Additionally, for `string` and text `datatypes`:

– *child attributes* specify whether the attribute allows certain other attributes to be embedded as part of its values,
– *content patterns* define the attribute's typical values and their formatting, possibly with information regarding content length,
– *content examples* together with a choice of a configurable *string similarity metric*.

Additionally, for numeric datatypes:

– *minimum* and *maximum* values,
– choice of a *probability distribution* over the attribute's values,
– *units* in which numeric values may be given.

And finally, for images:

– *image examples* from which classifiers can be trained based on features like image size, colour histogram and a content similarity metric, as shown in [12].

Additionally, we can associate probability estimates with some of the above attribute metadata. With the context and content patterns, we can associate estimates of $P(Pat|A)$ and $P(A|Pat)$, where A is a specific attribute of an instance to be extracted, and Pat is a pattern that – depending on whether it holds for a particular phrase in an analysed document or not – indicates if that phrase is a good candidate for A's value. We will call the first estimate *pattern recall* and the second *pattern precision*. These statistics can be obtained either from expert users, or using a proper form of training data. Pattern recall can be straightforwardly estimated from samples of attribute values. To compute precision, however, labelled training documents are needed so as to identify the number of occurrences of the pattern that do not pertain to the attribute.

For the values of *numeric attributes*, expert users may choose and parametrise a probability distribution (a probability density or mass function for continous vs. discrete variables) $Pd(N|A)$, where A is an attribute and N is an observed number. This is analogous to pattern recall of content patterns. Alternatively, such distribution can be estimated from training attribute values. Then, if labelled training documents are available, we can map[5] this distribution to an estimate of a conditional distribution $P(A|N)$ which is analogous to pattern precision of content patterns.

[5] A simple mapping can be done e.g. by sorting all occurences of numbers N in training documents according to their decreasing $Pd(N|A)$, and then estimating a probability function of the $Pd(N|A)$ values based on the portion of positive examples in the closest numbers.

For `string` attributes, users can supply a list of example values (e.g. a list of names of products of a certain type). Then a *string similarity metric*[6] can be used to assess the likelihood with which an observed phrase belongs to such list. As in the case of numeric probability distributions, values of the similarity metric for labelled phrases taken from training documents can be used to estimate a conditional distribution $P(A|phrase)$, which is analogous to pattern precision of content patterns. Classification scores for `images` can be used in a similar manner.

3.2 Instance extraction knowledge

A number of assertions can be stated about instances of an extractable class. These assertions will typically pertain to:

- *cardinality* of a certain attribute,
- *relations among attribute values*, e.g. price with tax is greater than the price without tax,
- *relations among attribute positions*, e.g. some attribute is often mentioned first,
- *general relations among attributes* (beyond their content), e.g. the text in a product picture's `alt` description is similar to the product name.
- *span of an attribute* over multiple instances, e.g. the heading with the name of a category of products can be assigned to multiple instances of product offer.

For most of these assertions, probabilities in the form $P(assertion|class)$ can be estimated. This can be again done by an expert guess or based on counts in sample training instances.

3.3 Instance data

The presentation ontology as authored by users will typically contain a limited amount of patterns, and probability estimates given by users will be very rough. As discussed above, users may supply more (and hopefully more precise) extraction knowledge by providing either

- Labelled training documents
- Examples of extracted instances (or values of certain attributes); note that in many practical scenarios, we can only get unlabelled documents but can match them with such sample instances.

Both types of training data can be used to automatically discover new patterns[7] that can be used to identify attributes.

[6] We experimented with similarities derived from character and word edit distances and with similarities based on common substrings.

[7] In this direction we experimented with a set-covering grammar induction algorithm that generalised from positive examples.

3.4 Local extraction knowledge

Until now we described *global* extraction knowledge, which serves for extraction from arbitrary documents from a given domain. In addition, *local* extraction knowledge, limited to a specific document collection, becomes useful especially when extracting from multiple documents from a single source with common formatting structure (e.g. HTML documents from a single web site). We rely on inducing local knowledge in an unsupervised manner during the extraction process, as described in Section 5.2.

3.5 Example

A sample taken from a presentation ontology designed to extract computer monitor descriptions from arbitrary websites is shown in Figure 1. It depicts a single `Monitor` class with an attribute `name`, which may embed another attribute `manufacturer` (in 90% of cases). A pattern is given for `name` content, which could match e.g. 'LCD Acer AL922'. Probability estimates state that 30% of monitor names match this pattern and that if this pattern appears in a document, it denotes a monitor name with 95% certainty.

```
<class id="Monitor">
 <attribute id="name" type="string" card="1">
  <name> name </name>
  <value>
    <contains>
     <att ref="manufacturer" card="1" recall="0.9" />
    </contains>
    <pattern recall="0.3" prec="0.95">
     LCD <att ref="manufacturer"/> <token type="ALPHANUM"/>+
    </pattern>
  </value>
 </attribute> ...
```

Fig. 1. Presentation ontology sample

4 From Domain Ontologies to Presentation Ontologies

Authoring a presentation ontology can be a complex and tedious task. While the inclusion of extraction patterns is specific for the IE setting, the abstract conceptual structure is analogous to that of domain ontologies. As the number of domain ontologies available on the semantic web increases, their reuse seems to be an obvious remedy for the ontology acquisition bottleneck in our approach. However, due to slightly different modelling principles we adopted (compared to 'pure' conceptual models), a *transformation* process is needed.

In order to transform a domain ontology expressed in the standard semantic web ontology language OWL[8] into one or multiple presentation ontologi/es, several steps need be carried out. Namely, for each presentation ontology to arise, we should (possibly repeatably, for a multi-class presentation ontology):

1. choose the core class C and add it to the presentation ontology,
2. create its attributes in the presentation ontology from various structures of the domain ontology (see below),
3. formulate ontological constraints (data type, cardinality) over attributes: based on constraints over properties from the domain ontology or based on known instances,
4. formulate IE "hooks" for each attribute: in addition to simple datatype restrictions over attributes, more extraction knowledge can be added based on the content or context of known instances from the domain ontology.

For step 2, several non-deterministic transformation rules can be formulated, for example:

- A *datatype property* D of C may directly yield an attribute.
- A datatype property D of some C_1, together with a *chain of object properties*[9] $(O_1, O_2, ..., O_n)$, where O_1 is object property of C, O_n is object property of C_1, and for every k, $1 \leq k \leq n-1$, there is a class having both O_k and O_{k+1} as its properties, may yield an attribute. A simple example of such chain with connecting classes, for C=`Computer`, C_1=`Disk` and D=`diskType` (with values 'CD', 'DVD'), is (`hasDiskDrive`, `writesTo`) with an intermediate class `DiskDrive`; this would yield an attribute such as 'diskDriveType'.
- A *set of mutually disjoint subclasses* of C may yield an attribute. For example, for C=`Computer`, the disjoint subclasses may be {`Desktop`, `Notebook`, `Palmtop`}, and would yield an attribute such as 'Portability' (with no property counterpart in the domain ontology).
- A set of mutually disjoint subclasses of some C_1 such that exists a chain of object properties between C and C_1 (in the same sense as in the second rule), may yield an attribute; this is a combination of previous two cases.

5 Towards a Versatile Ontology-Based IE Tool

5.1 Core Principles

Our tool under development takes advantage of several different sources of extraction knowledge. Namely the knowledge (1) entered by users, (2) learnt from training data, and (3) learnt from the structure of analysed documents. We think that using all three sources of extraction knowledge will contribute to at least two important features of an IE system.

[8] `http://www.w3.org/2004/OWL/`

[9] A typical representative of such properties is the 'part-of' property and its subproperties.

First, the system will become applicable to a broader range of tasks. In some applications, experienced users are willing to manually provide complex patterns indicating extraction targets that would be hard to induce automatically. In other cases, users are equipped with training data either in the form of examples of extracted instances or directly with labelled training documents. In some scenarios, groups of analysed documents exhibit regular formatting structures that can be exploited for extraction purposes. We envisage a system applicable to all these scenarios, in each case exploiting all available knowledge sources.

Second, if the IE system enables its users to systematically input their own extraction knowledge, it will be easier for them to debug cases where the extraction fails. The extraction process will thus become more transparent and it will be easier for users to understand why errors occur and how to prevent them.

To practically investigate the above aspects, we are building an IE system based on presentation ontologies. One scenario we currently experiment with is extraction of instances of typically a single class from multiple websites. Examples of such classes include detailed product descriptions or weather forecasts. Another application we aim at is IE from semi-structured medical reports and medicine leaflets.

5.2 Attribute Candidate Identification and Scoring

Based on the extraction "hooks" described in Section 3, phrases that are likely to represent attribute values are considered attribute candidates and assigned scores. As of now, we are still experimenting with different ways of computing these scores based on probability estimates discussed above. Currently we do not have a final method to present that would optimally combine the probability estimates associated with the multiple (often highly interdependent) evidences available in the presentation ontology.

This phase is intended to be executed on a collection of documents (e.g. a single website). After the first run through all documents, high-scoring attribute candidates can be selected for pattern induction. In this way, new local knowledge can be discovered that is specific to the current document collection, with its probability estimates being only based on that collection. In the case of websites, the induced knowledge will typically exploit formatting regularities, including generalised DOM paths to the extracted attributes. With an updated set of patterns, we can run attribute identification and scoring again[10] and hope to get previously uncovered (but still correct) attribute candidates. This step may be repeated several times. In this way, we can effectively induce simple *wrappers* for analysed websites in an unsupervised manner.

5.3 Instance Parsing

After identifying and scoring attribute candidates in the whole set of documents to be analysed, we may proceed with instance parsing. This step consists in se-

[10] For effectiveness, only attribute candidates whose score changes should be recomputed.

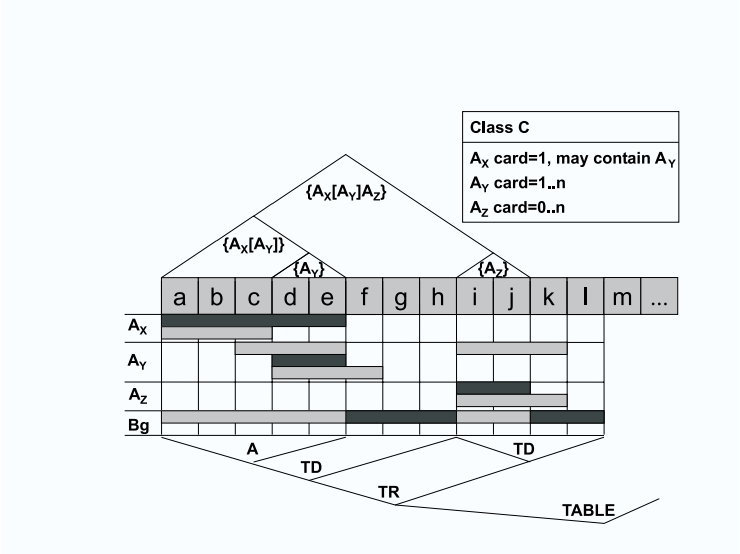

Fig. 2. Parsed instance example

lecting groups of attribute candidates that belong together and form instances of the extracted class(es). The resulting instances must adhere to the constraints imposed by the ontology, and should correspond to the most probable partitioning of the set of attribute candidates into instances.

For this purpose, we plan to use a bottom-up parsing algorithm capable of producing parses such as illustrated in Figure 2. A parse tree is shown for a single instance of the depicted class `C`, which consists of attributes `X`, `Y` and `Z`. The letters `a...k` denote document tokens while document structure is represented by the DOM tree below (for an HTML document). In the central table, attribute candidates are marked and a separate row is reserved for background (uninteresting) tokens. As in the case of attribute identification, it may be useful for the parser to infer local extraction knowledge when processing multiple structurally similar documents. The induced local knowledge should apply to whole instances, and may include a prevailing ordering of attributes in the current document collection or a regular positioning of all attributes in specific parts of document structure.

5.4 State of Implementation

Major components of the described extraction system have already been implemented and are being debugged, except for the instance parsing algorithm, which is still under development. Experiments on large data sets are yet to be done; our current application is the extraction of descriptions of products sold on the Internet. The chosen platform is Java. Extraction services will also be

offered to remote clients using web services. We expect to have an alpha version of the system by summer 2006.

6 Related Work

The number of existing IE tools is quite high. Recently reported tools include *S-CREAM* [6] and *MnM* [17]. They attempt to integrate the processes of training data labeling and subsequent automated extraction from new data. *Armadillo* [3] and *Pankow* [2], in turn, rely on bootstrapping training data from existing resources, which minimises human annotation effort. We intend to use a simple variant of bootstrapping in the form of gradual induction of local extraction knowledge, as described in sections 5.2 and 5.3. In comparison, our project focuses on applicability to a wide range of scenarios with varying amounts of training data, and allows expert users to input their extraction knowledge in the form of presentation ontologies. A similar approach is taken by [4], which introduces 'extraction ontologies' that contain attributes equipped with regular expression patterns, cardinalities and other domain knowledge. We however put more emphasis on using training data and on providing probability estimates. As in the case of wrapper induction systems [9, 8, 5], we try to exploit common structure exhibited by collections of analysed documents. The GATE NLP system [1] implements an extraction language called JAPE that serves a similar purpose as our extraction ontologies. We share our application area with the *CROSSMARC* project [13], which emphasises multi-lingual extraction mostly from web sites offering products.

7 Conclusion and Future Work

We presented a work-in-progress approach to IE that attempts to combine three sources of extraction knowledge – from expert users, from training data, and from common document structure present in an analysed document collection. The knowledge from expert and from training data is structured into a presentation ontology and includes probability estimates. Furthermore, the backbone of the presentation ontology can be, in principle, derived from a common domain ontology.

The most critical steps in developing a usable tool now consist in choosing the best way of combining the multiple evidences available for identifying and scoring attribute candidates and in implementing the instance parsing algorithm sketched in Section 5.3. Possible solutions will be tested in several application scenarios and different domains. In longer term, we plan to experiment with bootstrapping training examples supplied by users using web search engines as described in [2].

Acknowledgement

The research leading to this paper was supported by the European Commission under contract FP6-027026, Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content - K-Space.

References

1. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: ACL 2002.
2. Cimiano, P., Staab, S.: Learning by Googling. In: SIGKDD Explorations 2004.
3. Ciravegna, F., Chapman, S., Dingli, A., Wilks, Y.: Learning to Harvest Information for the Semantic Web. In: ESWS-04, Heraklion, Springer LNCS 2004.
4. Embley, D.W., Tao, C., Liddle, S.W.: Automatically extracting ontologically specified data from HTML tables with unknown structure. In: ER2002, Tampere 2002.
5. Gottlob, G., Koch, C.: Logic-based Web Information Extraction. In: SIGMOD 2004.
6. Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM – Semi-automatic CREAtion of Metadata. In: Proc. EKAW 2002.
7. 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.
8. Knoblock, C.A., Minton, S., Ambite, J.L., Ashish, N., Modi, P.J., Muslea, I., Philpot, A.G., Tejada, S.: Modeling Web Sources for Information Integration. In: Proc. AAAI WI 1998.
9. Kushmerick, N., Weld, D. S., Doorenbos, R.: Wrapper Induction for Information Extraction. In: Proc. Intl. Joint Conference on Artificial Intelligence 1997.
10. Labský, M., Svátek, V., Šváb, O.: Types and Roles of Ontologies in Web Information Extraction. In: ECML/PKDD04 Workshop on Knowledge Discovery and Ontologies, Pisa 2004.
11. Labský, 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 Int'l Conf. Web Intelligence (WI'05), IEEE 2005.
12. 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`.
13. Pazienza, M.T., Stellato, A., Vindigni, M.: Combining ontological knowledge and wrapper induction techniques into an e-retail system. In: Workshop on Adaptive Text Extraction and Mining (ATEM03) held with ECML/PKDD 2003, Cavtat 2003.
14. 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.
15. Šváb, O., Labský, M., Svátek, V.: RDF-Based Retrieval of Information Extracted from Web Product Catalogues. In: SIGIR'04 Semantic Web Workshop, Sheffield, 2004.
16. van Heijst, G., Schreiber, G., Wielinga, B.: Using Explicit Ontologies in KBS development, *Int. J. Human-Computer Studies*, Volume 46, 1997, 183-292.
17. 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.