# Information Extraction with Presentation Ontologies

**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
{labsky,svatek}@vse.cz

## Abstract

We describe an approach to information extraction that attempts to integrate diverse sources of extraction knowledge. The aim of our IE system under construction is to perform reasonably well under a broad scale of scenarios, with large differences in amounts of training data, manually specified patterns and document formatting structure. In our approach, the user initially creates a presentation ontology which describes the to-be-extracted objects both from a domain and a presentation point of view. Training data can then be used to improve extraction performance. The possibility to create presentation ontologies from domain ontologies is also investigated.

## 1 Introduction

In this text we discuss an approach to information extraction (IE) that 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.

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 labeled 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". Our work is inspired by an approach described by (Embley et al., 2002), which introduces "extraction ontologies" that contain attributes equipped with regular expression patterns, cardinalities and other domain knowledge. We aim to extend this work by providing a more complete framework for presentation ontology authoring, by adding support for automatic acquisition of extraction knowledge from training data, and by utilizing a statistically inspired parser with extracted instances on output.

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. A precursor IE system to the currently developed version was based on Hidden Markov Models and a heuristic instance parser described in (Labsky et al., 2005).

In Section 2, we describe the proposed presentation ontologies, show their possible content, discuss their authoring and show how training data can be used to extend the manually authored

knowledge. In Section 3 we sketch a possible approach to semi-automatically creating a presentation ontology from a conventional domain ontology. The next two sections are only work-in-progress reports and are devoted to parsing: Section 4 describes the process of identification and scoring of attribute candidates, and Section 5 sketches a bottom-up parsing algorithm aiming to extract instances. Both of these steps will process documents from the same source in parallel in order to exploit their common structure (if any). Finally, we conclude with a summary of related and future work in Sections 6 and 7.

## 2 Presentation Ontology

The concept of a presentation ontology[1] extends a 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.

A sample taken from a presentation ontology designed to extract 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.

### 2.1 Attribute extraction knowledge

For IE purposes, attributes are equipped with the following metadata that relate to extraction:

- *data type* which can be either `string`, `text`[2], `int`, `float`, `boolean` or `image`,
- *context patterns* define the attribute's typical surrounding tokens. In 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 (Labsky et al., 2005).

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

Figure 1: Presentation ontology sample

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 $P(A|Pat)$,

[1] A sample presentation ontology is available at `http://rainbow.vse.cz/preso`.

[2] The `text` data type can have a long content with other attributes and intermediate text embedded. This is motivated by capturing e.g. textual product descriptions.

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. For pattern recall, samples of attribute values suffice, while for precision, labeled training documents are needed. In case only unlabeled documents are available, a tool can be developed to help the user create and test new patterns on the unlabeled document collection, and to estimate their precision.

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 labeled training documents are available, we can map[3] this distribution to an estimate of a conditional distribution $P(A|N)$ which is analogous to pattern precision of content patterns.

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*[4] 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 labeled 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.

### 2.2 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.

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.

### 2.3 Identifying patterns in training 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 (1) examples of extracted instances (or only of values of certain attributes), or (2) labeled training documents[5]. Both types of training data can be used to automatically discover new patterns[6] that can be used to identify attributes.

### 2.4 Scope of 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 4.

## 3 Presentation vs. Domain Ontologies

Authoring a presentation ontology is a complex and tedious task. While the specification of extraction patterns is specific for the IE setting, the abstract conceptual structure is analogous to that of

[3] 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.

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

[5] In many practical scenarios, we can only get unlabeled documents paired with instances to be extracted.

[6] In this direction we experimented with a grammar induction algorithm that generalised from positive examples.

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[7] into one or multiple presentation ontologi/es, several steps need be carried out. Namely, for each presentation ontology to arise, we should:

1. choose the core class $C$,
2. create attributes of 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. formulation of 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.

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*[8] $(O_1, O_2, ..., O_n)$, 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.

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

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

## 4 Attribute Candidate Identification and Scoring

Based on the extraction "hooks" described in Section 2, 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. We 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[9] 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.

### 4.1 Combining Extraction Knowledge

Probability estimates of multiple observed evidences will be combined as follows. Let $\Phi_A$ be

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

the set of evidences that indicate presence (or absence) of attribute $A$. For each such evidence $E_i \in \Phi_A$, we have estimates of its precision $P(A|E_i)$ and its recall $P(E_i|A)$.

In addition, we need an estimate of $P(A)$ for each attribute. This will be set to a small constant (identical for all attributes), as this can hardly be estimated by users. If labeled training documents are available, we will estimate $P(A)$ using counts.

We will call a *positive evidence* any evidence where $P(A|E_i) > P(A)$, otherwise we have a *negative evidence*. On the other hand, recall is especially significant for cases where $E_i$ is missing: if a high recall is set, not observing $E_i$ will decrease the likelihood of $A$ much more than in case of a small recall.

We define an *attribute candidate* as every analysed phrase pointed to by at least 1 positive evidence. Then, if $|\Phi_A| > 1$, we need to combine multiple evidences into a single score for the attribute candidate. For each candidate for attribute $A$, let $Phi_A$ be split into $\Phi_A^+$, the set of evidences $E_i \in \Phi_A$ that hold for that candidate, and $\Phi_A^-$, the set of missing evidences $E_i \in \Phi_A$. As a score, we will use the combined probability $P(A|E \in \Phi_A)$. For example, $P(A|E_1, \neg E_2)$ represents $A$'s probability conditioned on one evidence that holds and on another that is missing. We will compute the combined probability under a kind of naive bayesian independence assumptions:

$$\forall_{E,F \in \Phi_A, F \neq E} : \; F \perp E|A, \; F \perp E|\neg A. \quad (1)$$

That is, evidences are assumed to be mutually independent within positive examples of $A$, and outside of $A$. We hypothesize this could be a reasonable assumption for a majority of patterns used in most applications. If a group of patterns significantly violates these independence assumptions, then those patterns can be combined into a single pattern.

Eq. 2 shows the computation of $P(A|E \in \Phi_A)$ under the assumptions in Eq. 1:

$$P(A|E \in \Phi_A) = \frac{P(A, E \in \Phi_A)}{P(E \in \Phi_A)} = \frac{1}{1 + \frac{P(A)}{P(\neg A)}^{|\Phi_A|-1} \Pi_{E \in \Phi_A^+} \frac{P(\neg A|E)}{P(A|E)} \Pi_{F \in \Phi_A^-} \frac{P(\neg A|\neg F)}{P(A|\neg F)}} \quad (2)$$

where

$$P(A|\neg F) = \frac{P(\neg F|A)P(A)}{1 - \frac{P(F|A)P(A)}{P(A|F)}}. \quad (3)$$

Derivation of Eq. 2 and 3 is shown in Appendix A.

## 5 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 selecting 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.

## 6 Related Work

The number of existing IE tools is quite high. Recently reported tools include *S-CREAM* (Handschuh et al., 2002) and *MnM* (Vargas-Vera et al., 2002). They attempt to integrate the processes of training data labeling and subsequent automated extraction from new data. *Armadillo* (Ciravegna et al., 2004) and *Pankow* (Cimiano and Staab, 2004), 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 4 and 5.

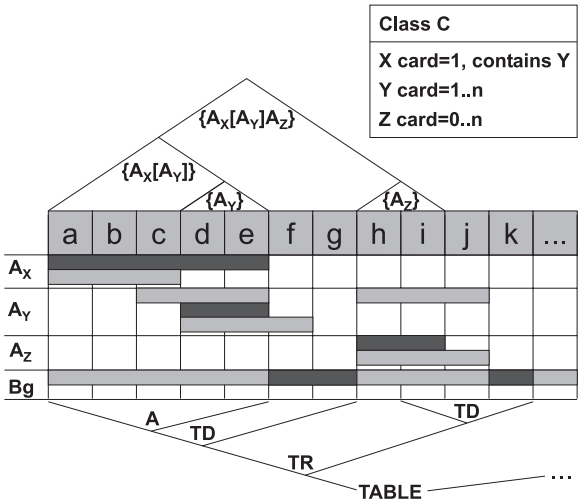


Figure 2: Parsed instance example

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 (Embley et al., 2002), however, we plan to put more emphasis on using training data and on providing probability estimates. As in the case of wrapper induction systems (Kushmerick et al., 1997; Knoblock et al., 1998; Gottlob and Koch, 2004), we try to exploit common structure exhibited by collections of analysed documents. The GATE NLP system (Cunningham et al., 2002) implements an extraction language called JAPE that serves a similar purpose as our extraction ontologies. We share our application area with the *CROSSMARC* project (Pazienza et al., 2003), 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. Our next steps 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. 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 (Cimiano and Staab, 2004).

**Acknowledgement.** This work has been partially supported by the EU under the IST 6th FP, Network of Excellence K-Space.

## References

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. *GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications.* In: ACL 2002.

P. Cimiano and S. Staab. *Learning by Googling.* In: SIGKDD Explorations 2004.

F. Ciravegna, S. Chapman, A. Dingli, and Y. Wilks. *Learning to Harvest Information for the Semantic Web.* In: ESWS 2004.

D.W. Embley, C. Tao, and S.W. Liddle. *Automatically extracting ontologically specified data from HTML tables with unknown structure.* In: Proc. ER 2002.

G. Gottlob and C. Koch. *Logic-based Web Information Extraction.* In: SIGMOD 2004.

S. Handschuh, S. Staab, and F. Ciravegna. *S-CREAM – Semi-automatic CREAtion of Metadata.* In: Proc. EKAW 2002.

C.A. Knoblock, S. Minton, J.L. Ambite, N. Ashish, P.J. Modi, I. Muslea, A.G. Philpot, and S.Tejada. *Modeling Web Sources for Information Integration.* In: Proc. AAAI WI 1998.

N. Kushmerick, D. S. Weld, and R. Doorenbos. *Wrapper Induction for Information Extraction.* In: Proc. Intl. Joint Conference on Artificial Intelligence 1997.

M. Labský, V. Svátek, O. Šváb: *Information Extraction from HTML Product Catalogues: from Source Code and Images to RDF.* In: Proc. WI 2005.

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

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

**Appendix A. Derivation of $P(A|E \in \Phi_A)$:**

First we derive a special case of the combined probability where all evidences $E \in \Phi_A$ hold, which we will denote as $P(A|E_1 \dots E_n)$:

$$\begin{aligned} P(A|E_1 \dots E_n) &= \frac{P(A, E_1 \dots E_n)}{P(E_1 \dots E_n)} \\ &= \frac{P(E_1 \dots E_n|A)P(A)}{P(E_1 \dots E_n|A)P(A) + P(E_1 \dots E_n|\neg A)P(\neg A)} \\ &= \frac{P(E_1|A) \dots P(E_n|A)P(A)}{P(E_1|A) \dots P(E_n|A)P(A) + P(E_1|\neg A) \dots P(E_n|\neg A)P(\neg A)} \end{aligned} \tag{4}$$

where

$$P(E_i|\neg A) = \frac{P(\neg A|E_i)P(E_i)}{P(\neg A)} \tag{5}$$

where

$$P(E_i) = \frac{P(E_i|A)P(A)}{P(A|E_i)} \tag{6}$$

Substituting Eq. 6 into 5 and Eq. 5 into 4 yields

$$\begin{aligned} P(A|E_1 \dots E_n) &= \frac{P(E_1|A) \dots P(E_n|A)P(A)}{P(E_1|A) \dots P(E_n|A)P(A) + P(\neg A)\Pi_{i=1\dots n}\frac{P(\neg A|E_i)P(E_i|A)P(A)}{P(\neg A)P(A|E_i)}} \\ &= \frac{1}{1 + \frac{P(A)}{P(\neg A)}^{n-1}\Pi_{i=1\dots n}\frac{P(\neg A|E_i)}{P(A|E_i)}} \end{aligned} \tag{7}$$

The above case only applies when all $E \in \Phi_A$ hold, and only uses evidence precisions (recalls are effectively ignored by Eq. 7). Next, we will derive a more general formula $P(A|E \in \Phi_A)$ that takes into account both evidences that hold and those that do not. Let $Phi_A$ be split into $\Phi_A^+$, the set of evidences $E \in \Phi_A$ that hold for some to-be-classified example, and $\Phi_A^-$, the set of missing evidences $F \in \Phi_A$. Then we get

$$P(A|E \in \Phi_A) = \frac{1}{1 + \frac{P(A)}{P(\neg A)}^{|\Phi_A|-1}\Pi_{E\in\Phi_A^+}\frac{P(\neg A|E)}{P(A|E)}\Pi_{F\in\Phi_A^-}\frac{P(\neg A|\neg F)}{P(A|\neg F)}} \tag{8}$$

This is analogous to Eq. 7 – in addition, we need to obtain $P(A|\neg F)$:

$$\begin{aligned} P(A|\neg F) &= \frac{P(\neg F|A)P(A)}{P(\neg F)} \\ &= \frac{P(\neg F|A)P(A)}{1 - P(F)} \\ &= \frac{P(\neg F|A)P(A)}{1 - \frac{P(F|A)P(A)}{P(A|F)}} \end{aligned} \tag{9}$$