Combining Multiple Sources of Evidence in Web 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. Extraction of meaningful content from collections of web pages with unknown structure is a challenging task, which can only be successfully accomplished by exploiting multiple heterogeneous resources. In the *Ex* information extraction tool, so-called extraction ontologies are used by human designers to specify the domain semantics, to manually provide extraction evidence, as well as to define extraction subtasks to be carried out via trainable classifiers. Elements of an extraction ontology can be endowed with probability estimates, which are used for selection and ranking of attribute and instance candidates to be extracted. At the same time, HTML formatting regularities are locally exploited.

1 Introduction

In the last decade, *web information extraction* (WIE) was dominated by two paradigms. One—*wrapper*-based—exploits regular surface-level structures found in HTML code, which can be used as anchors for the extraction. This approach is now widely adopted in industry, however, its dependence on formatting regularities 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; state-of-the-art trainable IE algorithms are surveyed e.g. in [10]. Again, however, sufficient amount of appropriate training data is rarely available in practice and manual labelling is often too expensive even with the help of active learning; statistical bootstrapping alleviates this problem to some degree but at the same time it 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* [2] enumerate attributes of the concepts to be extracted, 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 well-defined and rich conceptual structure 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 may cover a sensible part of target data and generate meaningful feedback for its own redesign. It seems that for web extraction tasks where the subject of extraction evolves and does not require sophisticated NLP, extraction ontologies are the first choice. However, to achieve competitive extraction results and to prevent overfitting to a few sample resources, one must not neglect available labelled data, formatting regularities and even pre-existing domain ontologies. This was the motivation for building our WIE tool named *Ex* [1], which exploits all the mentioned resources, with central role of extraction ontologies. It has been so far tested in three domains: product catalogues of computer monitors and TVs, contact information on medical pages, and weather forecasts. First experimental results were reported in [4]. In this paper we present the principle how multiple pieces of evidence from extraction ontologies are combined during the extraction process. Section 2 uses a real-world example to explain the most important features of extraction ontologies used in *Ex*. Section 3 describes the steps of the information extraction process and, especially, the underlying pseudo-probabilistic apparatus. Finally, section 4 surveys related research, and section 5 outlines future work.

2 Ex(traction) Ontology Content—Overview and Example

Extraction ontologies in *Ex* are designed so as to extract occurrences of *attributes*, i.e. standalone named entities, and occurrences of whole *instances* of *classes*, as groups of attributes that 'belong together', from HTML pages or texts in a domain of interest. An extraction ontology defines evidence of different types which is used to identify the extractable items. Token-, character- and formatting-level patterns may address both the content and context of attributes and instances to be extracted, axioms may encode their complex constraints and relations; there are also *formatting* constraints and ranges or distributions for *numeric attribute values* and for attribute *content lengths*. The extraction ontology language of *Ex* was introduced in [4].

In *Ex*, every piece of evidence may be equipped with two probability estimates: *precision* and *recall*. The *precision* $P(A|E)$ of evidence states how probable it is for the predicted attribute or class instance A to occur given that the evidence E 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* $P(E|A)$ 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 smoothed ratios of counts observed in text. Second, when no training data are available, the user specifies one or both parameters manually. Our initial experiments indicate that extraction ontology developers are often able to specify precision and recall values with accuracy sufficient to create useful prototype IE systems.

[1] Alpha version of *Ex* is available from `http://eso.vse.cz/~labsky/ex`.

```
<class id="Contact">
 <script src="contact.js" />

 <pattern recall="0.7"> ^ $title{0-3} .? $name ,? $title{0-3} </pattern>

 <axiom recall="0.8"> nameMatchesEmail($name, $email) > 0 </axiom>

 <classifier id="cls1" method="weka" classtype="attribute" features="ontology,ngram"
  name="weka.classifiers.rules.JRip" model="contacts.bin" elements="*"/>

 <attribute id="degree" type="name" card="0-4" eng="0.60">
  <content>
   <pattern recall="0.2" p="0.8"> ( Miss | Lady | Sir | MSc | MA | MPh ) .? </pattern>
   <pattern recall="1" type="format"> has_one_parent </pattern>
  </content>
 </attribute>

 <attribute id="name" type="name" card="1" eng="0.80">
  <pattern id="init"> (A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|R|S|T|U|V|W|X|Y|Z) .? </pattern>
  <content>
   <pattern recall="0.5" p="0.8">
    <pattern src="first.txt"/> <pattern ref="init"/>? <pattern src="last.txt"/> </pattern>
   <pattern recall="0.7" p="0.4">
    <pattern src="first.txt"/> <pattern ref="init"/>? <tok type="alpha" case="CA|UC"/>
   </pattern>
   <pattern p="0.7" recall="0.5"> <phr label="cls1.name" /> </pattern>
   <length> <distribution min="1" max="6" /> </length>
   <refers> nameRefersTo($, $other) </refers>
  </content>
  <context> <pattern recall="0.1" p="0.6"> person? name :? $ </pattern> </context>
 </attribute>

 <attribute id="email" type="name" card="0-1" eng="0.60">
  <content><pattern recall="0.9" p="0.9"> <pattern ref="gen.email"/></pattern> </content>
 </attribute>
</class>
```

Fig. 1. Fragment of code of extraction ontology for contact information

Our example shows a simplified *contact information* ontology developed within the EU project MedIEQ[2], aiming to extract peoples' degrees, names and emails. Extraction results for this task were presented in [4]. The first pattern claims that 70% of *Contact* instances start with a person name or degree separated by punctuation. An axiom further claims that in 80% of cases, the person's name and email exhibit string similarity which is identified by a script function *nameMatchesEmail()*, introduced in a script "contact.js" above. Finally, a *classifier link* contracts an external trained classifier to classify all attributes of the *Contact* class. Classifications made by this classifier are used by some of the attribute content patterns defined below, e.g. for *name* we expect the classifier to have a 70% precision and a 50% recall. Other content and context patterns rely on token-level regular expressions including large named entity lists; e.g. the first content pattern for *name* claims that 80% of its matches correctly identify a person name, while it is only expected to cover about 50% of all person names since the lists are not exhaustive. In addition, a script is used by the *refers* section to match co-referring mentions of the same person *name* on a single page.

3 The Extraction Process

The inputs to the extraction process are the extraction ontology and a set of documents. First, analysed documents are tokenized and equipped with formatting structure DOM trees[3]. The actual extraction consists of four stages.

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

[3] To parse malformed HTML we use the CyberNeko parser `http://people.apache.org/~andyc/neko/doc/html`.

3.1 Attribute candidate generation

Attribute candidates (*ACs*) are created where at least one content or context pattern matches. Let Φ_A be the set of all evidence E_i known for an attribute A. To compute a *conditional probability estimate* $P_{AC} = P(A|E \in \Phi_A)$ of how likely the *AC* is given the presence of each evidence, we use these naive bayesian independence assumptions:

$$\forall_{E,F \in \Phi_A, F \neq E}: \ F \perp E|A, \ F \perp E|\neg A. \tag{1}$$

That is, evidence is assumed to be mutually independent within positive examples of A and outside of A. To compute P_{AC}, we use the precision $P(A|E_i)$ and recall $P(E_i|A)$ of each evidence $E_i \in \Phi_A$ and their truth values, and the prior probability $P(A)$ of encountering each attribute in text. Φ_A^+ denotes the set of evidence $E_i \in \Phi_A$ observed for that candidate, and Φ_A^- is the set of unobserved evidences $E_i \in \Phi_A$:

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

The $P(A|\neg F)$ member of Eq. 2 is computed according to Eq. 3; the remaining values are known from the extraction ontology. Derivation of both formulas is shown in [5].

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

The set of (possibly overlapping) *ACs* created during this phase is represented as an *AC lattice* spanning through the document, where each *AC* node is scored as $score(AC) = log(P_{AC})$. Apart from *ACs*, the lattice includes one 'background' node BG_w for each token w that takes part in at least one *AC*. Supposing $|AC|$ is the length of an *AC* in tokens, we define $score(BG_w) = \min_{AC, w \in AC} log(\frac{1-P(AC)}{|AC|})$ where $|AC|$ is the *AC* length in tokens. The extraction process can terminate here by extracting all *ACs* on the best path through this lattice or it may continue with instance parsing and formatting pattern induction.

3.2 Instance candidate generation

Initially, each *AC* is used to create a simple instance candidate *IC*=$\{$*AC*$\}$. Then, increasingly complex *ICs* are generated bottom up from the working set of *ICs*. At each step, the highest scoring (seed) *IC* is popped from the working set and added to a *valid IC set* if it satisfies *all* ontological constraints. The seed *IC* is then extended using its neighboring *ACs* if their inclusion does not break ontological constraints. Only a *subset* of constraints is considered at this time as e.g. minimum cardinality constraints or some axioms could never get satisfied initially. An *AC* is only added as a reference if a user-defined function determines its value may corefer with an *AC* that already is part of the *IC*. The newly created ICs are added to the working set. A limited number of ACs is allowed to be skipped (AC_{skip}) between the combined *IC* and *AC*, leading to a penalization of the created IC. The IC scores are computed based on their AC content and on the observed values of evidence E known for the IC class C:

$$sc_1(IC) = exp(\frac{\sum_{AC \in IC} log(P_{AC}) + \sum_{AC_{skip} \in IC}(1 - log(P_{AC_{skip}}))}{|IC|}) \tag{4}$$

$$sc_2(IC) = P(C|E \in \Omega_C) \quad (5)$$

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. 2. By experiment we chose the Prospector [1] pseudo-bayesian method to combine the above into the final IC score:

$$score(IC) = \frac{sc_1(IC)sc_2(IC)}{sc_1(IC)sc_2(IC) + (1 - sc_1(IC))(1 - sc_2(IC))} \quad (6)$$

IC generation ends when the working set becomes empty or on a 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.

3.3 Formatting pattern induction

When extracting from a single web page or web site, it often happens that a large part of valid *ICs* satisfies some unforeseen *HTML formatting pattern*. E.g. all person names and emails could reside in two dedicated columns of a table. We try to *induce* these local formatting patterns as follows. First, the best scoring path of non-overlapping *ICs* is found through the valid *IC* lattice. For each *IC* on the path, we find its nearest containing formatting *block* element (e.g. paragraph, div, table cell). We then create a *subtree of formatting elements* between the block element (inclusive) and the *ACs* comprising the *IC*. Each subtree consists of the formatting element names and their order within parent. Formatting subtrees whose relative and absolute frequencies satisfy certain thresholds are transformed into new *context patterns* indicating presence of the corresponding class, with precision and recall based on their relative frequencies. The *AC* and *IC* generation phases are then re-run for the newly created local context patterns, re-scoring and possibly yielding new *ACs* and *ICs*.

3.4 Attribute and instance parsing

All valid *ICs* are merged as additional nodes into the *AC* lattice created in previous steps, so that each *IC* node can be avoided by taking a path through standalone *ACs* or through background states. In the merged lattice, each *IC* node is scored by $|IC| \times score(IC)$. The merged lattice is searched for the best scoring node sequence from which all instances and standalone attributes are extracted.

4 Related Work

Most state-of-the-art WIE approaches use inductively learned models and only consider ontologies as additional structures to which extracted data are to be adapted [3], rather than directly using rich ontologies to guide the extraction process. An exception is the approach taken by Embley and colleagues [2], which even inspired our early work; the main difference is our effort to combine manually encoded extraction knowledge with HTML formatting and trained classifiers, the possibility to equip extraction evidence with probability estimates, and the pragmatic distinction we see between extraction ontologies and domain ontologies: extraction ontologies can be adapted to the way data are typically *presented* on the web while domain ontologies describe the domain semantics. A parallel stream to ontology-based IE is that relying on automated discovery of new extractable attributes from large amounts of documents using statistical and NLP methods, as in [6]. On the other hand, formatting information is heavily exploited in IE from tables [7]. Our system has a slightly different target from both these; it should allow for fast IE prototyping even in domains where there are few documents available and the content is semi-structured. Advanced automatic methods also exist for coreference resolution in attribute values [9] or for the estimate of mutual affinity among these values [8]. Our approach, again, relies on the author to supply such knowledge.

5 Conclusions and Future Work

The *Ex* information extraction system is capable of combining, in a pseudo-probabilistic manner, multiple pieces of extraction evidence, provided by the user within an extraction ontology as well as learned from annotated data and from local formatting regularities. As the next step we want to focus on bootstrapping techniques and to compare our extraction results with other approaches using standard datasets. 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. 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.
2. 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.
3. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. *J. Web Sem.*, volume 2, pp. 49–79, 2004.
4. Labský, M., Nekvasil, M., Svátek, V., Rak, D.: The Ex Project: Web Information Extraction using Extraction Ontologies. In: Proc. PriCKL workshop, ECML/PKDD 2007.
5. Labský, M., Svátek, V: Information extraction with presentation ontologies. Technical report, KEG UEP, `http://eso.vse.cz/~labsky/ex/ex.pdf`.
6. Popescu, A., Etzioni, O.: Extracting Product Features and Opinions from Reviews. In: Proc. EMNLP 2005.
7. Wei, X., Croft, B., McCallum, A.: Table Extraction for Answer Retrieval. *Information Retrieval Journal*, vol. 9, issue 5, pp. 589-611, 2006.
8. Wick, M., Culotta, A., McCallum, A.: Learning Field Compatibilities to Extract Database Records from Unstructured Text. In: Proc. EMNLP 2006.
9. Yates, A., Etzioni, O.: Unsupervised Resolution of Objects and Relations on the Web. In: Proc. HLT 2007.
10. Dietterich, T. G.: Machine Learning for Sequential Data: A Review. In: Proc. SSPR 2002.