iDocument: Using Ontologies for Extracting and Annotating Information from Unstructured Text

Benjamin Adrian[1], Jörn Hees[2], Ludger van Elst[1], and Andreas Dengel[1,2]

[1] Knowledge Management Department, DFKI,
Kaiserslautern, Germany
[2] CS Department, University of Kaiserslautern
Kaiserslautern, Germany
firstname.lastname@dfki.de

Abstract. Due to the huge amount of text data in the WWW, annotating unstructured text with semantic markup is a crucial topic in Semantic Web research. This work formally analyzes the incorporation of domain ontologies into information extraction tasks in iDocument. Ontology-based information extraction exploits domain ontologies with formalized and structured domain knowledge for extracting domain-relevant information from un-annotated and unstructured text. iDocument provides a pipeline architecture, an extraction template interface and the ability of exchanging domain ontologies for performing information extraction tasks. This work outlines iDocument's ontology-based architecture, the use of SPARQL queries as extraction templates and an evaluation of iDocument in an automatic document annotation scenario.

1 Introduction

Automatically or semi-automatically extracting structured information from unstructured text is an important step towards text understanding. Existing information extraction (IE) systems are mostly specialized in limited domains. A scenario where end users may rapidly query a text base with ad hoc queries[3] is nearly impossible to implement with existing IE technologies. (e.g., *Select scientific conferences in 2009 and their deadlines for paper submissions with a focus on information extraction.*)

Knowledge engineering approaches (e.g., Textmarker [1]) suffer from the knowledge engineering bottleneck which means an expert has to provide and maintain a rule base. Existing machine learning approaches avoid these efforts. Instead they require previously annotated training corpora with expensive ground truth data [2].

Thus, common IE systems do not provide scalability, adaptability, and maintainability for being used in cost saving and generic user scenarios.

[3] According to Ralph Grishman (2002) about the project Proteus: *Our long-term goal is to build systems that automatically find the information you're looking for, pick out the most useful bits, and present it in your preferred language, at the right level of detail.*

In order to overcome these shortcomings, we present iDocument, a flexible ontology-based information extraction (OBIE) system. It uses terminology and instance knowledge of domain ontologies that are written in RDF/S [3]. Exchanging a domain ontology customizes the system on a completely new domain of interest. Instead of using expensive IE template definition specifications, iDocument provides a generic and standardized IE template interface based on the syntax of the RDF query language SPARQL. Thus, extraction templates can be expressed by SPARQLing the ontology's RDFS scheme. As result, iDocument generates an RDF graph that contains RDF triples which were extracted from given text documents and comply with the SPARQL template.

The structure of this paper is as follows: Related work on using ontologies in information extraction is summarized in Section 2. Next, Section 3 formally describes, how RDFS ontologies may be used in IE tasks. Section 4 shows iDocument's architecture, extraction pipeline, and template population algorithm. In Section 5, an evaluation presents the quality of annotated instances and facts.

2 Related Work

iDocument extends basic IE principles by using ontologies. Comparable OBIE systems are GATE [4], SOBA [5] or SummIt-BMT [6]. In difference to these, iDocument does not use the ontology as simple input gazetteer that is a plain list of relevant labels but as model for semantic analysis such as instance disambiguation and discourse analysis. The technique of using existing domain ontologies as input for information extraction tasks and extraction results as base for ontology population was first presented by Empley in [7] by using relational database technologies. Sintek et al. [8] applied a similar scenario on Semantic Web ontologies. A bootstrapping approach was presented by Maedche, Neumann, and Staab in [9] by learning new domain ontologies semi-automatically and populating these in a final step. The previously mentioned OBIE systems do not support any template mechanisms. In the Message Understanding Conference series (MUC), extraction templates were defined by using named slots that were annotated with constraints and rules [10]. These template specifications tended to be very long and complex and were hard to create by system users. Hobbs and Israel claimed in [11] that template design is related to ontology engineering. Following this assumption, the use of extraction ontologies as sort of template for IE is presented by Labský in [12]. Here the entire ontology specifies those information elements (e.g., fields and entities) that should be extracted from text. In iDocument, SPARQL queries that are formulated by using the ontology's RDFS vocabulary [3] serve as technique for expressing extraction templates.

3 Domain Ontologies in Information Extraction Tasks

"In common sense, a domain ontology (or domain-specific ontology) models a specific domain, or part of the world. It represents the particular meanings of

terms as they apply to that domain."[4] Domains may be for example: the Olympic Summer Games 2004, the Knowledge Management Department at DFKI, or a Personal Information Model [13] inside a Semantic Desktop.

By using the RDFS [3] vocabulary the main components of a domain ontology $\mathcal{O}$ can be defined as $\mathcal{O}(H_C, H_P, I, S, A)$:

Hierarchy of classes (H_C) A resource c can be defined as `rdfs:class`. A class c_1 may specialize class c_2 by expressing `rdfs:subClassOf`(c_1, c_2). The transitive closure of all `rdfs:subClassOf` expressions builds the class hierarchy H_C.

Hierarchy of properties (H_p) A property p expresses a semantic relation between two classes $p(c_1, c_2)$. A property p_1 may specialize property p_2 if `rdfs:subPropertyOf`(p_1, p_2). The transitive closure of `rdfs:subPropertyOf` expressions builds the hierarchy of properties H_P.

Object Properties (P_O) A property $p \in P_O$ is called object property if `rdfs:range` is defined as `rdfs:range`(p, c) with $c \in H_C \backslash$`rdfs:Literal`^{T}, where `rdfs:Literal`^{T} denotes the reflexive, transitive subhierarchy of class `rdfs:Literal`.

Datatype Properties (P_{DT}) A property $p \in P_{DT}$ is a datatype property if `rdfs:range` is defined as `rdfs:range`$(p,$ `rdfs:Literal`$)$.

Instances (I) Instances are resources i with an `rdf:type` property that is defined as `rdf:type`(i, c) with $c \in H_C \setminus \{$`rdfs:Literal`$^T\}$.

Symbols (S) symbols are resources s with an `rdf:type` property that is defined as `rdf:type`(s, c) with `rdfs:subClassOf`$(c,$ `rdfs:Literal`$)$.

Assertions (A) Assertions are triple expressions in the form of $p(i, r)$ with $p \in H_P$ and $i, r \in H_C \cup I \cup S$.

Based on such an ontology, it is possible to define four major tasks for ontology-based information extraction.

Symbol Recognition If a similarity function $sim(s, s_e)$ decides that a phrase s inside a text matches an existing symbol $s_e \in S$, s is recognized as symbol. Each datatype property $p \in P_{DT}$ inside a valid assertion $p(i, s_e) \in A$ about an instance $i \in I$ is called the symbol's type.

- If $sim(s, s_e)$ bases on content similarity, s is called *content symbol* and denoted as s_c. Each content symbol s_c has to exist in S.
  (e.g., assuming `foaf:mbox`(urn:BenjaminAdrian, "adrian@dfki.de") $\in A$, all occurrences of "adrian@dfki.de" in text will be recognized as content symbol with type `foaf:mbox`)
  In traditional IE systems this task is called *Named Entity Recognition*.
- If $sim(s, s_e)$ bases on structural similarity, s is called *structure symbol* and denoted as s_s.
  (e.g., assuming `foaf:mbox`(urn:BenjaminAdrian, "adrian@dfki.de") $\in A$, the occurrence of "dengel@dfki.de" in text will be recognized as structure symbol with type `foaf:mbox`).
  In traditional IE systems this task is called *Structured Entity Recognition*.

[4] as described in Wikipedia, online available at `http://en.wikipedia.org/w/index.php?title=Ontology_(information_science)&oldid=284405492`, 2009-04-21

Instance Recognition The unification of recognized content symbols to existing instances i_{exist} is called instance recognition. For a *content symbol* s_c, i_{exist} is a recognized instance if an assertion $p(i_{exist}, s_c) \in A$ exists where at least one type of s_c matches p. Thus it is possible that for single content symbols more than one recognized instance exist (e.g., content symbols of type `foaf:firstName`).
In case of a *structure symbol* s_s, instance recognition may either be solved with unification that adds a new assertion $p(i_e, s_s) \notin A$ to an existing instance $i_e \in I$ or as instantiation of a new instance $i_{new} \notin I$ with an assertion $p(i_{new}, s_s) \notin A$, with p is a type of symbol s_s.
Traditional IE systems call this task *Template Unification* or *Template Merge*.

Fact Recognition Assume that $P_Q \subseteq H_P$ is a set of queried properties inside the extraction template Q. Recognized facts are assertions of type $p(i_1, i_2)$ or $p(i_1, s)$ with $i_1, i_2 \in I, p \in P_Q, s \in S$. If all components i_1, i_2 or s of a recognized fact were recognized in text in the previous steps and the recognized fact is not an existing assertion $f_r \notin A$, the fact is called *extracted fact*. Otherwise if $f_r \in A$ it is called *completed fact* as the template is completed with known assertions of the domain ontology.
Traditional IE systems call this task *Fact Extraction*.

Template Population A given extraction template Q is populated with recognized facts. It is possible that a single template is populated with multiple populations. (e.g., In case of "Select scientific conferences in 2009 and their deadlines for paper submissions with a focus on information extraction", each recognized conference with its paper due forms a single populated template.)
Traditional IE systems call this task *Scenario Extraction*.

The following section describes the architecture, IE task implementations, and extraction templates of the OBIE system iDocument.

4 The OBIE system iDocument

As outlined in Figure 1, iDocument's architecture comprises the following components: a domain ontology, a text collection, SPARQL templates, an OBIE pipeline, and finally results in populated templates. A text collection contains content that is relevant for the current question and domain of interest.

4.1 Alignment Metadata

For using a domain ontology in iDocument, relevant parts for extraction purpose have to be annotated with the MOBIE vocabulary[5] (Metadata for OBIE) in a mapping file:

`mobie:Entity` For using a class of instances for instance recognition purpose, it has to be assigned to mobie:Entity by using rdfs:subClassOf. (e.g., foaf:Person rdfs:subClassOf mobie:Entity)

[5] refer to `http://ontologies.opendfki.de/repos/ontologies/obie/mobie`

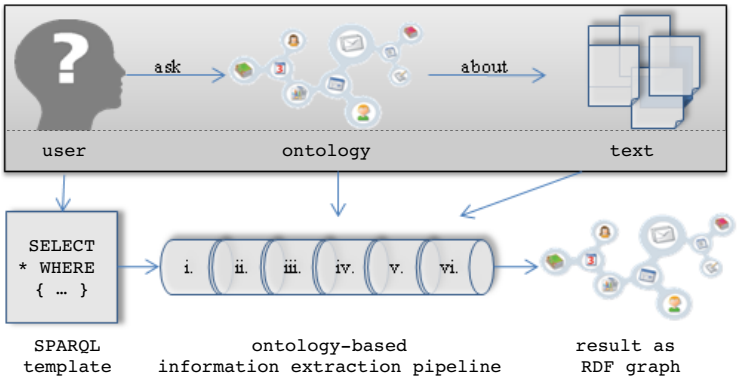

Fig. 1. OBIE scenario

`mobie:symbol` For using a datatype property for symbol recognition purpose, it has to be assigned to mobie:symbol by using rdfs:subPropertyOf. (e.g., foaf:firstName rdfs:subPropertyOf mobie:symbol)

`mobie:relates` For using an object property for fact recognition purpose, it has to be assigned to mobie:relates by using rdfs:subPropertyOf. (e.g., foaf:knows rdfs:subPropertyOf mobie:relates)

4.2 Extraction Templates

iDocument provides an extraction template interface that processes templates written in SPARQL. Users may write templates by using the ontology's RDFS vocabulary. (e.g., SELECT * WHERE {?person rdf:type foaf:Person. ?person foaf:member ?org. ?org rdf:type foaf:Organisation.} extracts persons and organisations and facts about memberships from text.)

4.3 Template Population

The algorithm for populating templates follows a most constrained variable heuristic. A SPARQL query can be represented as forest of join expressions. The algorithm transforms the forest into a list of paths from root to leaf and sorts this list in a descending order by the length of paths. Iteratively, it removes the longest path from the list and tries to populate it with recognized facts.

4.4 Extraction Pipeline

iDocument is built upon an OBIE pipeline consisting of six OBIE tasks. The first two are standard text based analyzes for instance (i) *Normalization* and (ii) *Segmentation*. Succeeding tasks are OBIE tasks as described in Section 3, namely (iii) *Symbol Recognition*, (iv) *Instance Recognition*, (v) *Fact Recognition*, and (vi) *Template Population*.

This pipeline was implemented with Believing Finite-State Cascades [14]. Each task is based on text or results of preceding tasks and produces weighted hypotheses. *Normalization* transforms a text document to RDF representation that consists of its plain text and existing metadata such as author, date, title. *Segmentation* partitions text passages into paragraphs, sentences, or tokens.

Results of the OBIE pipeline are transcribed in RDF graphs. These may be visualized and approved by users and/or be handed to specific applications. In the current state of implementation, the focus of iDocument is set on annotating text with existing instances and facts from domain ontologies. Upcoming versions will enhance its extraction functionalities.

5 Evaluation

The evaluation of iDocument was done by analyzing the quality of extracted results of instance and fact recognition tasks with precision and recall. As populating and maintaining domain ontologies is burdensome, the evaluation was designed to show how the degree of extraction quality correlates with the amount of assertions inside the domain ontology. As corpus data, the DFKI/OCAS 2008[6] corpus was used. It contains a domain ontology about the Olympic Summer games 2004 and fifty documents that were annotated with symbols, instances and facts [15]. In an adaption of leave-one-out cross validation, one document was taken as candidate, the remaining documents with their annotations built the assertions of the domain ontology. For analyzing the learning behaviour, assertions of the domain ontology were divided into two parts, i.e. training set and test set. The impact of existing knowledge about a single document's content or knowledge about other documents in the same domain was analyzed by creating four types of training sets, i.e.:

X-Doc, 0-Ont Training set contains just a ratio of x % of the currently analyzed document's annotations.

0-Doc, x-Ont Training set just contains a ratio of x % of the all other documents' annotations, except annotations of the currently analyzed document.

x-Doc, x-Ont Training set contains a ratio of x % of the all existing documents' annotations.

x-Doc, 100-Onto Training set contains a ratio of 100 % of the all other documents' annotations and a ratio of x % of the currently analyzed document's annotations.

The training sets were used as assertions inside the domain ontology. During each test run the ratio x was increased from 0 % to 100 % in steps of 10 %. The test set always contained 100 % of annotations about the currently analyzed document. The evaluation task was to check the degree of extracted instances and facts. Figure 2 shows results of the instance recognition task for instances of type (Person, Nation, Sport, and Discipline) in the upper row. At left it shows

[6] `http://idocument.opendfki.de/wiki/Evaluation/Corpus/OlympicGames2004`

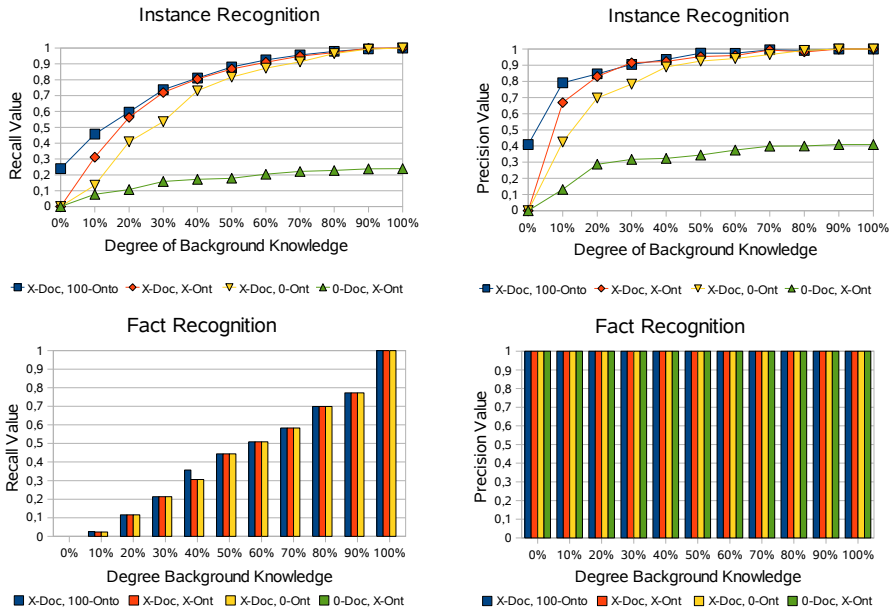

Fig. 2. Recall and precision progressions for (i) instance recognition as shown on top and (ii) fact recognition as shown on bottom

the progression of recall regarding an increasing amount of assertions inside the domain ontology. The logarithmic increase can be explained by the existence of multiple symbols per instance inside the ontology that were recognized during symbol recognition. If the ontology contains all relevant instances and facts that occur inside a test document, iDocument is able to recognize these with 100 % recall and precision. The recall progression of recognizing facts about a person's nationality follows a linear behavior as shown in the lower row. The step from 90 % to 100 % backround knowledge can be explained by rounding double values, as the amount of these facts per document were often below ten. The precision of recognized facts is always at 100 %, because iDocument only annotated existing facts that are asserted inside the domain ontology.

In both cases instance and fact recognition, the progression of an ontology with assertions shows, that precision does not decrease if the ontology knows more than just the document annotations.

6 Conclusion and Outlook

This work described the use of RDF/S domain ontologies for information extraction and annotation. The OBIE system iDocument was presented, including a pipeline-based architecture of OBIE tasks and a SPARQL-based template interface for defining which information to extract from text. The evaluation showed

that iDocument retrieved instances and facts in text if they exist inside a domain ontology. Future and ongoing efforts in iDocument are spent on increasing the ability of extracting new information from text.
This work was financed by the BMBF project Perspecting (Grant 01IW08002).

References

1. Atzmüller, M., Klügl, P., Puppe, F.: Rule-Based Information Extraction for Structured Data Acquisition using TextMarker. In: Proc. LWA-2008 (Special Track on Knowledge Discovery and Machine Learning). (2008)
2. Ireson, N., Ciravegna, F., Califf, M.E., Freitag, D., Kushmerick, N., Lavelli, A.: Evaluating Machine Learning for Information Extraction. In Raedt, L.D., Wrobel, S., eds.: ICML. Volume 119 of ACM Int. Conf. Proc. Series., ACM (2005) 345–352
3. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C recommendation, World Wide Web Consortium (2004)
4. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving GATE to meet new challenges in language engineering. JNLE **10**(3-4) (2004) 349–373
5. Buitelaar, P., Cimiano, P., Frank, A., Hartung, M., Racioppa, S.: Ontology-based Information Extraction and Integration from Heterogeneous Data Sources. Int. Journal of Human-Computer Studies (11) (2008) 759–788
6. Endres-Niggemeyer, B., Jauris-Heipke, S., Pinsky, M., Ulbricht, U.: Wissen gewinnen durch Wissen: Ontologiebasierte Informationsextraktion. Information - Wissenschaft & Praxis **57**(1) (2006) 301–308
7. Embley, D.W., Campbell, D.M., Smith, R.D., Liddle, S.W.: Ontology-based Extraction and Structuring of Information from Data-Rich Unstructured Documents. In: CIKM '98: Proc. of the 7th Int. Conf. on Information and Knowledge Management, New York, NY, USA, ACM (1998) 52–59
8. Sintek, M., Junker, M., van Elst, L., Abecker, A.: Using Information Extraction Rules for Extending Domain Ontologies. In: Workshop on Ontology Learning. CEUR-WS.org (2001)
9. Maedche, A., Neumann, G., Staab, S.: Bootstrapping an Ontology-based Information Extraction System. In Szczepaniak, P., Segovia, J., Kacprzyk, J., Zadeh, L.A., eds.: Intelligent Exploration of the Web. Springer, Berlin (2002)
10. Grishman, R., Sundheim, B.: Design of the MUC-6 evaluation. In: Proc. of a workshop held at Vienna, Virginia, Morristown, NJ, USA, Association for Computational Linguistics (1996) 413–422
11. Hobbs, J., Israel, D.: Principles of Template Design. In: HLT '94: Proc. of the workshop on HLT, Morristown, NJ, USA, ACL (1994) 177–181
12. Labský, M., Svátek, V., Nekvasil, M., Rak, D.: The Ex Project: Web Information Extraction using Extraction Ontologies. In: Proc. Workshop on Prior Conceptual Knowledge in Machine Learning and Knowledge Discovery (PriCKL'07). (2007)
13. Sauermann, L., van Elst, L., Dengel, A.: PIMO - a Framework for Representing Personal Information Models. In: Proc. of I-Semantics' 07, JUCS (2007) 270–277
14. Adrian, B., Dengel, A.: Believing Finite-State cascades in Knowledge-based Information Extraction. In: KI 2008: Advances in Artificial Intelligence. Volume 5243 of Lecture Notes in Computer Science., Springer (2008) 152–159
15. Grothkast, A., Adrian, B., Schumacher, K., Dengel, A.: OCAS: Ontology-Based Corpus and Annotation Scheme. In: Proc. of the HLIE Workshop 2008, ECML PKDD (2008) 25–35