iDocument:
Using Ontologies for Extracting Information from Text

Benjamin Adrian[1], Heiko Maus[1], and Andreas Dengel[1,2]
[1]KM Department, German Research Center for Artificial Intelligence (DFKI)
[2]CS Department, University of Kaiserslautern
FirstName.LastName@dfki.de

Abstract: This work outlines system and usage principles of the ontology-based information extraction system iDocument. Ontology-based information extraction reuses existing domain knowledge for extracting and annotating relevant information from domain-related text. iDocument provides an architecture, an API, and a user interface for supporting users and developers in ontology based knowledge annotation and acquisition tasks. The main contribution of this work is a generic and standardized information extraction template interface for retrieving relevant information from text by using existing domain ontologies.

1 Introduction

The extraction of relevant information from unstructured text is an important but knowledge intensive and therefore complex and expensive task. Existing information extraction (IE) systems are specialized in limited domain and hard to query with ad hoc questions about relevant text content. While learning approaches require training corpora with expensive ground truths data, knowledge engineering approaches suffer from the knowledge engineering bottleneck which means an expert has to provide and maintain a rule base. Thus, common IE systems do not provide scalability, adaptability, and maintainability for being used in cost saving and generic business scenarios.

To overcome these shortcomings, we present iDocument, a generic ontology-based information extraction (OBIE) system that uses ontological background knowledge in terms of existing vocabularies and instance knowledge. iDocument uses existing knowledge from personal or business domains (e.g. relational databases, concept maps, taxonomies, etc.). Following Semantic Web, iDocument exchanges and extracts knowledge based on the W3C standard RDF. Existing knowledge is used as input in a serial IE pipeline of extraction tasks for extracting possible answers concerning user specified ad hoc queries on a given text collection. The final goal of iDocument is defined and solved as follows (see Fig. 1):

By using an existing domain ontology, users formulize questions about a text collection. With respect to existing domain instance knowledge, iDocument responses a ranked answer list consisting of extracted and existing facts that are represented in the domain ontology's vocabulary.

The structure of this paper is as follows: We compare iDocument with other OBIE systems in Section 2 and describe the components, the pipeline, and the user interface of iDocument in Section 3. In Section 4 we summarize benefits of this approach, list scenarios, where iDocument has been applied to successfully and describe future goals.

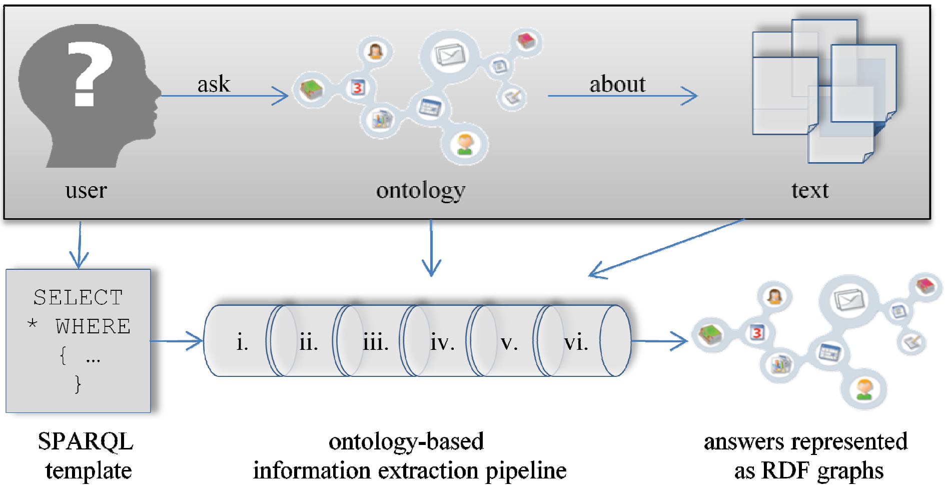

ask
about
user
ontology
text
SELECT
* WHERE
{ ...
}
i.
ii.
iii.
iv.
v.
vi.
SPARQL template
ontology-based information extraction pipeline
answers represented as RDF graphs

Figure 1: iDocument's architecture

2 Related Work

iDocument is based on IE principles that are well presented in [AI99]. Comparable and approved OBIE systems are GATE [Bu06], or SOBA [Bu06]. In difference to these, iDocument does not use the ontology as input gazetteer that is a plain list of relevant labels but as model for semantic analyses such as instance disambiguation, discourse analysis. iDocument also supports the population of extraction templates. The concept of querying ontologies with IE templates in order to extract information from text is completely missing in existing OBIE systems. Other OBIE approaches can be found in the proceedings of the OBIES workshop 2008 [Ad08]. The technique of using existing domain ontologies as input for information extraction tasks and extraction results for ontology population and therefore knowledge acquisition was presented in [Si01].

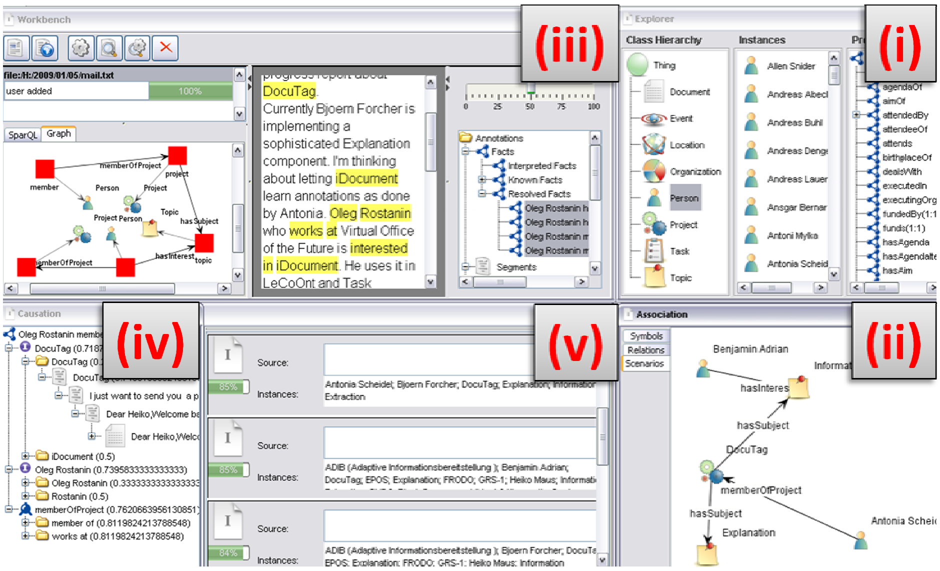
3 System Description

As outlined in Figure 1, iDocument’s architecture comprises of following components: domain ontology, a text collection, SPARQL queries, an ontology-based extraction pipeline, and finally query results.

- Existing background knowledge about a domain of interest resides in a **domain ontology** that is written in Semantic Web standards RDFS or OWL. The ontology has to be annotated with metadata for ontology-based information extraction (MOBIE[1]). These annotations are easy to create, but necessary in order to tell iDocument what parts of the ontology have to be used inside the information extraction tasks.
- A **text** collection contains content that is relevant for the current question and domain of interest. As iDocument is based on the Aperture[2] framework, it supports common document formats. (e.g. PDF, DOC, HTML, etc.)
- A user defined **query** as extraction template. This template can be defined in the W3C standard SparQL.
  (e.g., `SELECT * WHERE {?p rdf:type foaf:Person; foaf:member ?o. ?o rdf:type foaf:Organisation}`
- An OBIE pipeline consisting of six OBIE tasks, namely (i) *Normalization*, (ii) *Segmentation*, (iii) *Symbolization*, (iv) *Instantiation*, (v) *Contextualization*, and (vi) *Population*. This pipeline has been implemented by using Believing Finite-State Cascades [AD08]. Each task is based on either text or results of preceding tasks and produces weighted hypotheses. The *Normalization* task transforms a text document to RDF representation that consists of its plain text and existing metadata such as author, date, title. The *Segmentation* task partitions text passages to units of paragraphs, sentences, or tokens. During *Symbolization*, token sequences that match either labels of individuals and roles, or patterns inside the ontology are annotated as symbols. With respect to the queried template, *Instantiation* resolves symbols as instance and role candidates, if the individual's type and the role as such is part of the template. During *Contextualization*, iDocument classifies not yet instantiated symbols, resolves roles between resolved individuals, and finally populates the input template in form of multiple template instances. A template instance is a possible and weighted query result. These may be incomplete in the sense of query evaluation in database systems. The final *Population* step transfers the knowledge of valid template instances to the domain ontology.
- Results of the OBIE pipeline are transcripted in RDF graphs. These may be visualized and approved by users and/or be handled to specific applications.
- iDocument is implemented as Java library with a dedicated API that can be deployed as web service easily. The system provides a UI (see Figure 2) with visualizations of and interactions of main OBIE components for instance: (i) an ontology browser that visualizes the class hierarchy and lists instances of classes and possible roles; (ii) graph visualization panels for inspecting template queries, filled template instances, or instance properties, (iii) an annotation workbench that highlights extracted information inside the input text; (iv) explanation panels for exploring chains of cause and effect between hypotheses and finally (v) result list for browsing populated templates.

[1] Please inspect http://ontologies.opendfki.de/ for more information about the MOBIE vocabulary.

[2] Aperture is an open source Java framework that extracts data and metadata from documents in a standard vocabulary, http://aperture.sourceforge.net/



file:/H:/2009/01/05/mail.txt

user added 100%

progress report about DocuTag. Currently Bjoern Forcher is implementing a sophisticated Explanation component. I'm thinking about letting iDocument learn annotations as done by Antonia. Oleg Rostanin who works at Virtual Office of the Future is interested in iDocument. He uses it in LeCoOnt and Task

Oleg Rostanin membe
DocuTag (0.7187
DocuTag (0.1
DocuTa
I just want to send you a p
Dear Heiko,Welcome ba
Dear Heiko,Welco
iDocument (0.5)
Oleg Rostanin (0.7395833333333333)
Oleg Rostanin (0.3333333333333333
Rostanin (0.5)
memberOfProject (0.7620663956130851)
member of (0.8119824213788548)
works at (0.8119824213788548)

Source:
85% Instances: Antonia Scheidel; Bjoern Forcher; DocuTag; Explanation; Informatior Extraction

Source:
85% Instances: ADIB (Adaptive Informationsbereitstellung); Benjamin Adrian; DocuTag; EPOS; Explanation; FRODO; GRS-1; Heiko Maus; Informati

Source:
84% Instances: ADIB (Adaptive Informationsbereitstellung); Bjoern Forcher; DocuTa EPOS; Explanation; FRODO; GRS-1; Heiko Maus; Information

Figure 2: User Interface of iDocument

4 Conclusion and Outlook

iDocument reuses vocabulary and knowledge existing from domain ontologies and provides a generic template interface for user defined queries who wants to extract relevant information from text. We already have been applied iDocument in several scenarios by using concept maps, organizational repositories, or personal knowledge models as input ontology. An evaluation was done in [AD08]. In the project Perspecting, iDocument is used for semi automatic knowledge acquisition for personal knowledge models. This work was supported by “Stiftung Rheinland-Pfalz für Innovation”.

Bibliography

[AD08] Adrian, B.; Dengel, A. Believing Finite-State cascades in Knowledge-based Information Extraction KI 2008: Advances in Artificial Intelligence, Springer, 2008, 5243, 152-159

[Si01] Sintek, M.; Junker, M.; van Elst, L.; Abecker, A. Using Information Extraction Rules for Extending Domain Ontologies. Workshop on Ontology Learning, CEUR-WS.org, 2001

[Bu06] Buitelaar, P.; Cimiano, P.; Racioppa, S.; Siegel, M. Ontology-based Information Extraction with SOBA Proc. of LREC, 2006

[Bo04] Bontcheva, K.; Tablan, V.; Maynard, D.; Cunningham, H. Evolving GATE to meet new challenges in language engineering, Cambridge University Press, 2004, 10, 349-373

[Ad08] Adrian, B.; Neumann, G.; Troussov, A.; Popov, B.(Eds) Proc. Workshop on Ontology-based Information Extraction Systems, CEUR-WS/Vol-400, 2008

[AI99] Appelt, D. E.; Israel, D. J. Introduction to Information Extraction Technology: A tutorial prepared for IJCAI-99, 1999