

Ontology based Text Annotation – OnTeA

Michal Laclavik¹, Martin Šeleng¹, Emil Gatiaľ¹, Zoltan Balogh¹, Ladislav Hluchy¹,
¹*Institute of Informatics, Slovak Academy of Sciences, Dubravska cesta 9,
Bratislava, 845 07, Slovakia*

Abstract: In this paper we describe a solution for the **ontology based text annotation** (OnTeA) tool. The tool analyzes a document or text using regular expression patterns and detects equivalent semantics elements according to the defined domain ontology. The solution has been used, evaluated and it is further developed within the K-Wf Grid project and the NAZOU Slovak national project.

1. Introduction

When documents such as HTML, or text are processed by a computer system, it needs to understand the document structure. Web documents are structured but their structure is understandable mainly for humans, which is the major problem of the Semantic Web. The OnTeA tool tries to create structured semantic metadata from such documents according to the application domain ontology model. Thus OnTeA does not create a new ontology, but tries to map documents with its equivalent in the defined application ontology.

Annotating is a writing-to-learn strategy to be used while reading or rereading. An annotated text helps readers/processors to reach a deeper level of understanding. Several annotation tools exist. Annotea [1] is a system for creating and publishing shareable annotations of Web documents. Built on HTTP, RDF, and XML, Annotea provides an interoperable protocol suitable for implementation within Web browsers to permit users to attach data to Web pages so that other users may, on their own choice, see the attached data when they later browse the same pages. The Annotea project is a part of the project Semantic Web Advanced Development (SWAD). Unlike the Annotea system, the Ruby annotation is stored along with the text that has to be annotated as XML tags. Some user agents might not understand ruby mark-up or may not be able to render ruby text correctly. In either situation, it is generally preferable to render ruby text, so that information is not lost [2]. Other solutions and tools exist [3][4] and have been evaluated. Most of them provide infrastructure and protocols for manual stamping documents with semantic tags. OnTea works on documents, in particular domain described by domain ontology and use regular expression patterns for automatic semantic annotation, where it tries to create semantic version of text/document according to the domain ontology.

2. Methodology and the Approach

While most of annotation solutions try to find and create an object in the text or to provide semantic tags for the reader, in OnTea we try to detect ontology elements within existing application/domain ontology knowledge model. It means that by the OnTea annotation engine we want to achieve the following objectives:

- Detecting Meta data from Text
- Preparing improved structured data for later computer processing
- Structured data are based on application ontology model

The tool creates semantic metadata according to the domain ontology from text. OnTeA tool analyzes a document or text using a regular expression patterns and detects equivalent

semantics elements according to the defined domain ontology. Several cross application patterns are defined but in order to achieve good results, new patterns need to be defined for each application. In addition, OnTeA creates a new ontology individual of a defined class and assigns detected ontology elements/individuals as properties of the defined ontology class. The domain ontology needs to incorporate special ontology extension (Figure 1) used by Ontea. This extension contains one class *Pattern* with several properties.

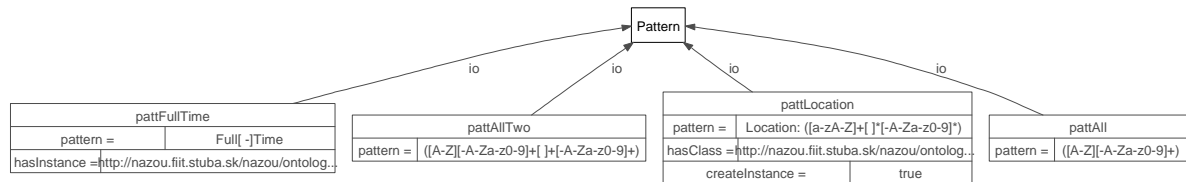


Figure 1: Pattern ontology with several individuals from NAZOU project domain ontology

The *Pattern* class represents regular expression patterns which are used to annotate plain text with ontology elements. The *Pattern* individual $\{pattern\}$ is evaluated by a semantic annotation algorithm. On Figure 1 we can see several simple patterns which can detect ontology individuals by matching String properties of such individuals.

The properties of *Pattern* class are *hasClass.Pattern*, *hasInstance.Pattern*, *pattern.Pattern*, *pattern.createInstance*. The instances of the *Pattern* class are used to define and identify relations between a text/document and its semantic version according to the domain ontology, where the *pattern* property contains the regular expression which describes textual representation of the relevant ontology element to be detected. The examined text/document is processed with the regular expression for every pattern. If property *hasInstance* is not empty, an individual included in this property is added to a set of detected ontology elements. Moreover, when the *hasClass* property exists in the *Pattern*, the RDQL query is constructed and processed to find the individuals that match the condition:

- The individual is the class of *hasClass*
- a *property* of individual contains the matched word

When property *createIndividual* is set *True* and corresponding individual with found keyword is not found, such individual of *hasClass* type is created.

The underlying principle of the Ontea algorithm can be described by the following steps:

1. The text of a document is loaded.
2. The text is proceed by defined regular expressions and if they are found, corresponding ontology individual according to rest of pattern properties is added to a set of found ontology individuals.
3. If no individual was found for matched pattern and createInstance property is set, a simple individual of the class type contained in the hasClass property is created with only property rdf:label containing matched text.
4. Such process is repeated for all regular expressions and the result is a set of found individuals.
5. An empty individual of the class representing proceed text is created and all possible properties of such ontology class are detected from the class definition.
6. The detected individual is compared with the property type and if the property type is the same as the individual type (class), such individual is assigned as this property.
7. Such comparison is done for all properties of a new individual corresponding with the text/document as well as for all detected individuals.

The algorithm also uses inference in order enable assignment of a found individual to the corresponding property also if the inferred type of a found individual is the same as the property type. The weak point of the algorithm is that if the ontology definition corresponding with the detected text contains several properties of the same type, in this

case detected individuals cannot be properly assigned. Crucial steps of the algorithms as well as inputs and outputs can be seen also on Figure 2.

3. Architecture and Technology

Architecture of the system contains similar elements as the main annotation algorithm described above.

Inputs are text resources (HTML, email, plain text) which need to be annotated as well as corresponding domain ontology with defined patterns individuals (Figure 1). An output is a new ontology individual, which corresponds to the annotated text. Properties of this individual are filled with detected ontology individuals according to defined patterns.

Ontea works with RDF/OWL Ontologies [5]. It is implemented in Java using Jena Semantic Web Library [6] or Sesame library [7]. In both implementation inference is used to achieve better results.

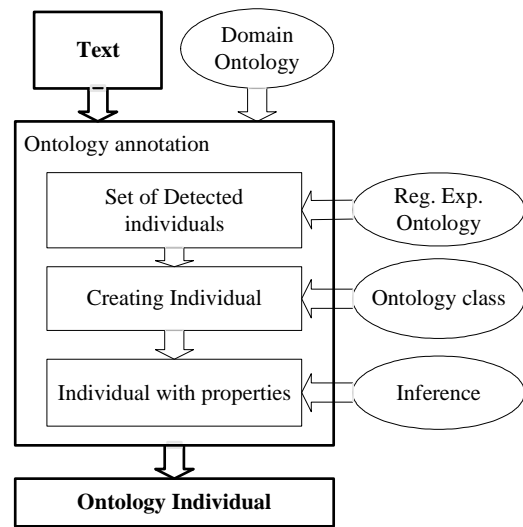


Figure 2: Ontea Tool Architecture

4. Examples of Use

Ontea has been created in the K-Wf Grid [9] and NAZOU [10] projects. The semantic text annotation is an important subtask in both projects. In K-Wf Grid, Ontea is used to translate or associate text input from a user to do-main ontology elements. This is used in two cases:

- When a user wants to define his/her problem by typing free text – Ontea detects relevant ontology elements and creates a semantic version of the problem understandable for further computer processing.
- The second case is when users use text notes for collaboration and knowledge sharing [8]. Notes are showed to the user in appropriate context, which is detected by Ontea.

A specific use of Ontea in the NAZOU project is described in next chapter. We provide more detailed examples on the Job Offer Application domain because the success rate of algorithm was measured on this problem domain.

5.1 Use of Ontea in Job Offer Application

NAZOU (Research and Development of Tools for Knowledge Discovery, Maintenance and Presentation, SPVV 1025/04) [10] is a Slovak project. The project has been launched in September 2004 and it is focused on discovery, maintenance and presentation of knowledge. The Pilot application is the Job search application, where tools are used to find, download, categorize, annotate, search and display job offers to job seekers. As stated in the title of the project, this project focuses on development of reusable tools. One of such tools is a tool Ontea .

Main components of Job Offer ontology are: job category, duty location, position type, required skills or offering company, which can be then detected by Ontea algorithm.

On right side of figure 3 the individual of the Job Offer is created based on the semantic annotation of a Job Offer document (left side of figure 3), using simple regular expression patterns as showed on Figure 1 where main individuals can be detected by the title property such as silSQL or skillPHP individuals. In this example the job offer location - New York and USA are identified by a regular expression „([A-Za-z]+)“ a „([-A-Za-z0-

9]+ []+[-A-Za-z0-9]+)“, because individual locNY has the property title „New York“, locUS has the property title „USA“.

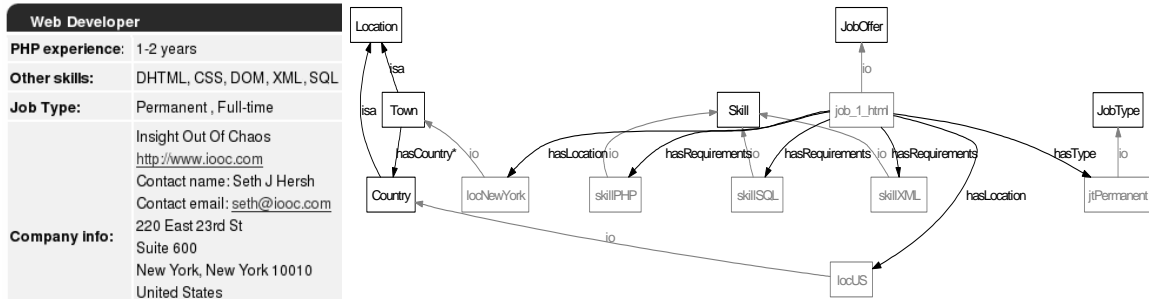


Figure 3: On left: Web Document; On the right: Job Offer Individual Created by Ontea

Similarly, other ontology elements are detected. Detected ontology individuals are then assigned as properties of job offer, thus ontology instance of job offer is created out of its text representation in the NAZOU pilot application.

5. Success Rate of Ontea Algorithm

In this chapter we discuss the algorithm success rate. As reference test data, we use 500 job offers filled in defined ontology manually according to 500 html documents representing reference job offers. Ontea was running on the reference ontology and reference html documents and the result was new ontology metadata of 500 job offers, which were automatically compared with manually entered job offers ontology metadata. In this test, Ontea used only simple regular expressions, which match from 1 to 4 words starting with a capital letter and Ontea did not create extra new property individuals in ontology.

Table 1. The comparison of results computed using the Ontea tool with reference data. The count row represents the number of job properties assigned to a job offer in reference data. The Ontea row represents the number of detected properties by the Ontea tool. The match row represents the number of same properties in the reference and Ontea ontology metadata.

Count	4	4	6	6	4	6	6	6	5	...	6	6	4	4	5	4
Ontea	8	7	8	8	12	8	10	9	9	...	7	7	6	6	7	6
Match	4	4	6	6	4	6	5	6	3	...	5	5	3	3	4	4
Success rate %	100	100	100	100	100	100	83,3	100	60	...	83,3	83,3	75	75	80	100

From the data in the table we can compute sample mean (2) sample variance (3) and sample standard deviation (4), which can be considered as basic measures of success rate.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = 83,16\bar{3}\% \quad (2)$$

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 = 3,222\% \quad (3)$$

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} = 17,95\% \quad (4)$$

As we can see in Table 1, Ontea tool finds more results than it is present in reference data. These extra found results can be relevant or irrelevant but we would have to check all data manually. We have checked extra detected results for several randomly chosen cases and we can say that most of found results is duplicity due to duplicity in reference data. For example, in reference data, a company offering job was in one case “Google” and in another case “Google, Inc.”. Ontea detected both values and assigned them to a new job offer as the job offering company, however different cases may occur, when

the found text is irrelevant; as an example can serve the “PHP” language detected as relevant needed expertise detected from an advertisement or a header. New found elements are in most cases relevant and they appear if there is inconsistency in reference data or if regular expressions for a selected problem domain are not set up carefully. Due to these facts, we show two more cases of a success rate. The basic sample characteristics can be calculated according to above equations (2), (3), (4). If the extra results are all relevant, the values are:

$$\bar{x} = 88,523\% , s^2 = 1,592\% , s = 12,619\% \quad (5)$$

If they are all irrelevant, the values are:

$$\bar{x} = 56,48\% , s^2 = 3,298\% , s = 18,162\% \quad (6)$$

6. Conclusions and Future Work

The described solution is used in the K-Wf Grid [2] and the Znalosti project to detect relevant structured knowledge described by a domain specific ontology model in the unstructured text. The main difference between existing annotation solutions such as Anotea [1] is detection of ontology elements from existing domain ontology, while other annotation solutions try to create such ontology. In the NAZOU project our solution is used to detect structured information about job offers. In the K-Wf Grid project our solution is used to detect a user context/problem from the text description as well as annotate user knowledge entered in a form of text notes [8]. The results archived using Ontea tool are satisfactory but in our future work we will try to create a testing set of text documents for semantic annotation and to evaluate a tool success rate more precisely.

This work is supported by projects K-Wf Grid EU RTD IST FP6-511385, NAZOU SPVV 1025/2004, RAPORT APVT-51-024604, VEGA No. 2/6103/6.

References

- [1] Annotea Project, <http://www.w3.org/2001/Annotea/>, (2001)
- [2] W3C, Ruby Annotation, 2001, <http://www.w3.org/TR/ruby/>
- [3] The Institute for Learning and Research Technology, RDF Annotations, 2001, <http://ilrt.org/discovery/2001/04/annotations/>
- [4] Patriarche R., Gedzelman S., Diallo G., Bernhard D., Bassolet C-G., Ferriol S., Girard A., Mouries M., Palmer P., Simonet A., Simonet M.: A Tool for Textual and Conceptual Annotations of Documents; Innovation and the Knowledge Economy, Volume 2, Issues, Applications, Case Studies; Edited by Paul Cunningham and Miriam Cunningham; IOS Press, pp.865-872. ISSN 1574-1230, ISBN 1-58603-563-0.
- [5] W3C: Web Ontology Language OWL, 2005, <http://www.w3.org/TR/owlfeatures/>
- [6] HP Labs and Open Source Community, Jena Semantic Web Library, 2005, <http://www.sf.net/>
- [7] OpenRDF.org, Sesame RDF Database, 2006, <http://www.openrdf.org/>
- [8] Laclavik M., Gatial E., Balogh Z., Habala O., Nguyen G., Hluchy L.: Experience Management Based on Text Notes (EMBET); Innovation and the Knowledge Economy, Volume 2, Issues, Applications, Case Studies; Edited by Paul Cunningham and Miriam Cunningham; IOS Press, pp.261-268. ISSN 1574-1230, ISBN 1-58603-563-0.
- [9] K –Wf Grid Consortium: K –Wf Grid IST Project Website, 2005, <http://www.kwfgrid.net/>
- [10] NAZOU Project Website, 2006, <http://nazou.fiit.stuba.sk/>
- [11] Laclavik M., Gatial E., Balogh Z., Habala O., Nguyen G., Hluchy L.: Semantic Annotation based on Regular Expressions; Proceedings of ITAT 2005 Information Technologies - Applications and Theory, Peter Vojtas (Ed.), Pririodovedecka fakulta UPJS v Kosiciach, 2005, pp.305-306. ISBN 80-7097-609-8.