Ontology based Text Annotation – OnTeA

Michal Laclavik[1], Martin Seleng[1], Emil Gatial[1], Zoltan Balogh[1], Ladislav Hluchy[1],
[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 semi-automatic **on**tology based **te**xt **a**nnotation (OnTeA) tool. The tool analyzes a document or text using regular expression patterns and detects equivalent semantics elements according to the defined domain ontology.

1. Introduction

Automated annotation of the Web documents is a key challenge of the Semantic Web effort. Web documents are structured but their structure is understandable only for humans, which is the major problem of the Semantic Web.

Annotation solutions can be divided into manual and semi-automatic methods. This different strategy depends on a use of the annotation. There is number of annotation tools and approaches such as CREAM [6] or Magpie [7] which follow the idea to provide users with useful visual tools for manual annotation, web page navigation, reading semantic tags and browsing [9] or provide infrastructure and protocols for manual stamping documents with semantic tags such as Annotea[1], Rubby[2] or RDF annotation[3].

Semi-automatic solutions focus on creating semantic metadata for further computer processing, using semantic data in knowledge management [8] or in Semantic Organization[4] applications (see chapter 4). Semi-automatic approaches are based on natural language processing [2] [3], a document structure analysis [4] or learning requiring training sets or supervision [5]. Moreover, other pattern-based semi-automatic solutions such as PANKOW and C-PANKOW [1] exist, using also Google API for automatic annotation. The algorithm seems to be slow when annotating a large number of documents needed in knowledge management or Semantic Organization applications. There is no evaluation of performance but description of the algorithm with frequent connections to Google API does not seem to be fast enough.

Ontea works on text, in particular domain described by domain ontology and uses regular expression patterns for semi-automatic semantic annotation. In Ontea we try to detect ontology elements within the existing application/domain ontology 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

2. Methodology and the Approach

The Ontea tool analyzes a document or text using 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

[1] http://www.w3.org/2001/Annotea/
[2] http://www.w3.org/TR/ruby/
[3] http://ilrt.org/discovery/2001/04/annotations/
[4] By “Semantic Organization” we understand applying semantic web ideas & technologies in organizations

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.

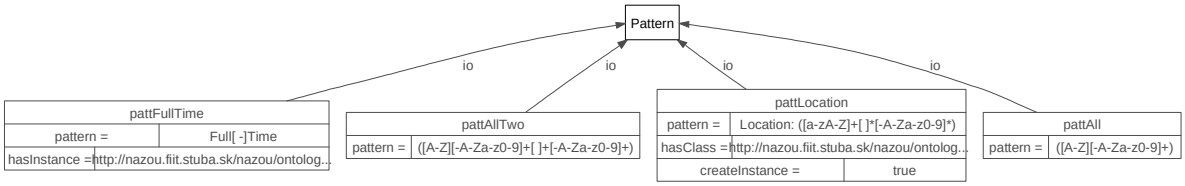

Figure 1: Pattern ontology with several individuals from NAZOU project domain ontology

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 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 in ontology metadata, 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. This problem can be overcome if algorithm is used only on creation of individuals of different property types. 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.

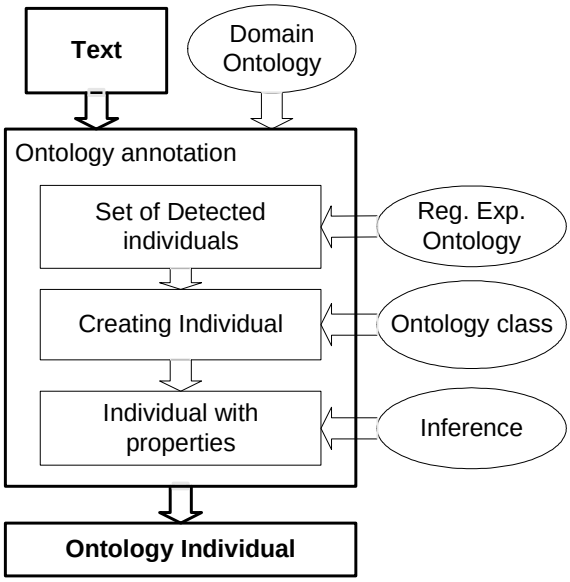

Figure 2: Ontea Tool Architecture

4. Examples of Use

Ontea has been created in the NAZOU[8] and K-Wf Grid[9] 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 domain 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 using text notes for collaboration and knowledge sharing [11]. 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.

4.1 Use of Ontea in Job Offer Application

The Ontea annotation was created as one of tools is the NAZOU project. It is used to create ontology metadata of offer HTML documents. The ontology metadata are then processed by other NAZOU tools as well as presented to the user [10]. 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. Main components of Job Offer ontology are: a job category, a duty location, a position type, required skills or an offering company, which can be then detected by the Ontea algorithm.

On the right side on 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 sillSQL 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-

[5] http://www.w3.org/TR/owlfeatures/
[6] http://jena.sf.net/
[7] http://www.openrdf.org/
[8] http://nazou.fiit.stuba.sk/
[9] http://www.kwfgrid.net/

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

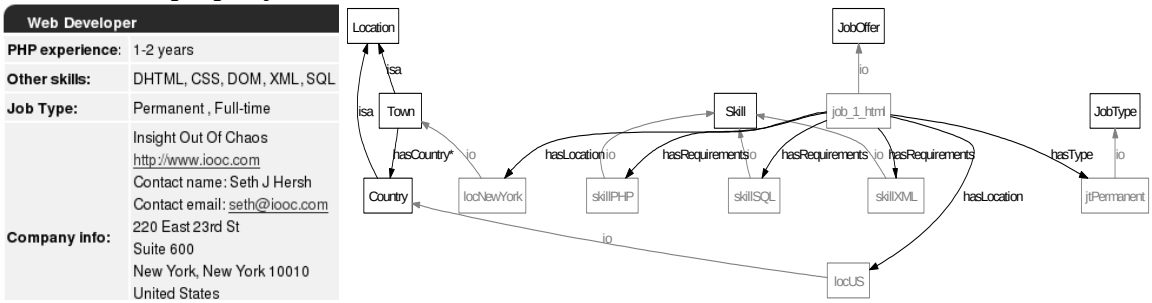

Figure 3: On left: Web Document; On the right: Job Offer Individual Created by Ontea

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 used 500 job offers filled in a defined ontology manually according to 500 html documents representing reference job offers. Ontea processed reference html documents using the reference ontology resulting in new ontology metadata consisting of 500 job offers, which were automatically compared with manually entered job offers ontology metadata. In this test, Ontea used only simple regular expressions matching from 1 to 4 words starting with a capital letter and Ontea did not create extra new property individuals.

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. The precision, recall and F1-measure rows represent the performance of annotation.

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
Precision	0,5	0,57	0,75	0,75	0,33	0,75	0,5	0,67	0,33	…	0,71	0,71	0,5	0,5	0,57	0,67
Recall	1	1	1	1	1	1	0,83	1	0,6	…	0,83	0,83	0,75	0,75	0,8	1
F_1-measure	0,67	0,73	0,86	0,86	0,5	0,86	0,62	0,8	0,43	…	0,77	0,77	0,6	0,6	0,67	0,8

To evaluate the performance of annotation, we used the standard recall, precision and F_1 measures (Table 1). Recall is defined as the ratio of correct positive predictions made by the system and the total number of positive examples. Precision is defined as the ratio of correct positive predictions made by the system and the total number of positive predictions made by the system:

$$Recall = \frac{Match}{Count} = \frac{Relevant\ retrieved}{All\ relevant},\ Precision = \frac{Match}{Ontea} = \frac{Relevant\ retrieved}{All\ retrieved} \qquad (1)$$

Recall and precision measures reflect the different aspects of annotation performance. Usually, if one of the two measures is increasing, the other will decrease. These measures were first used to measure IR (Information retrieval) system by Cleverdon [11]. To obtain a better measure to describe performance, we use the F1 measure (first introduced by van Rijsbergen [12]) which combines precision and recall measures, with equal importance, into a single parameter for optimization. F1 measure is weighted average of the precision and recall measures and is defined as follows:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (2)$$

We computed global estimates of performance using macro-averaging. Then the performance of classification for all 500 job offers is:

$$Precision = 0,63683025\,,\ Recall = 0,8316\overline{3}\,,\ F_1 = 0,704550462 \quad (3)$$

As we can see, the F_1 measure is high (over 70%), which means that Ontea tool gives satisfactory results.

6. Conclusions and Future Work

The described solution is used and evaluated in the K-Wf Grid and the NAZOU projects to detect relevant structured knowledge described by a domain specific ontology model in unstructured text. The most similar annotation solution to Ontea is PANKOW [1]. While PANKOW is a more generic solution, we think that Ontea is a simpler, faster (though the performance was not compared) solution with a better success rate, suitable for knowledge management or Semantic Organization applications.

The achieved results are quite satisfactory since the Ontea tool works with an average success over 70%, which is shown in the previous chapter. We believe that Ontea can be successfully used in a text analysis as well as in providing improved services for automatic text annotation, searching, categorizing, knowledge inference or reasoning.

In our future work we will strive to evaluate the algorithm on different application domains where we will be changing the number and quality of regular expression patterns, to find a good balance between precision and recall values.

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

References

[1] Cimiano P., Ladwig G., Staab S.: Gimme' the context: context-driven automatic semantic annotation with c-pankow. In WWW '05, pages 332-341, NY, USA, 2005. ACM Press. ISBN 1-59593-046-9.

[2] Madche A., Staab S.: Ontology learning for the semantic web. IEEE Intelligent Syst., 16(2):72-79, 2001

[3] Charniak E., Berland M.: Finding parts in very large corpora. In Proceedings of the 37th Annual Meeting of the ACL, pages 57-64, 1999.

[4] Glover E., Tsioutsiouliklis K., Lawrence S., Pennock D., Flake G.: Using web structure for classifying and describing web pages. In Proc. of the 11th WWW Conference, pages 562-569. ACM Press, 2002.

[5] Reeve L., Hyoil Han: Survey of semantic annotation platforms. In SAC '05, pages 1634-1638, NY, USA, 2005. ACM Press. ISBN 1-58113-964-0. doi: http://doi.acm.org/10.1145/1066677.1067049.

[6] Handschuh S., Staab S.: Authoring and annotation of web pages in cream. In WWW '02, pages 462-473, NY, USA, 2002. ACM Press. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511506.

[7] Domingue J., Dzbor M.: Magpie: supporting browsing and navigation on the semantic web. In IUI '04, pages 191-197, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-815-6.

[8] Uren V. et al.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Journal of Web Semantics: Science, Services and Agents on the WWW, 4(1):14-28, 2005.

[9] Uren V. et al.: Browsing for information by highlighting automatically generated annotations: a user study and evaluation. In K-CAP '05, pages 75-82, NY, USA, 2005b. ACM Press. ISBN 1-59593-163-5

[10] Návrat P., Bieliková M., Rozinajová V.: Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: CompSysTech 2005, Varna, Bulgaria, June 2005. – pp. IIIB.7.1-IIIB.7.6.

[11] Laclavik M. et al.: Experience Management Based on Text Notes (EMBET); Innovation and the Knowledge Economy; IOS Press, pp.261-268. ISSN 1574-1230, ISBN 1-58603-563-0.

[12] Cleverdon, C. W.; Mills, J. & Keen, E. M. (1966). Factors determining the performance of indexing systems. Vol. 1-2. Cranfield, U.K.: College of Aeronautics.

[13] C. J. Van Rijsbergen, Information Retrieval, Butterworth-Heinemann, Newton, MA, 1979