OnTeA: Semi-automatic Ontology based Text Annotation Method

Michal Laclavík[1], Martin Šeleng[1], and Marián Babík[1]

Institute of Informatics, Slovak Academy of Sciences, Dúbravská cesta 9, 845 07 Bratislava, Slovakia,
laclavik.ui@savba.sk,
WWW home page: http://ikt.ui.sav.sk/

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 semantic elements according to the defined domain ontology. Assuming the analyzed text belongs to a specific application domain, with a defined ontological model, we can create a semantic version of the text which can serve as a basis for further computer processing, where structured, formalized version of document is needed. This can help e.g. in categorization, common visualization of documents, searching and knowledge inference or reasoning. The solution has been used, evaluated and it is further developed within the K-Wf Grid, NAZOU, and Raport projects.

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 mainly for humans, which is the major problem of the Semantic Web. The Ontea tool tries to create structured semantic metadata from Web documents according to the application domain ontology model using semantic annotation techniques. Ontea does not create a new ontology, but rather tries to identify relevant ontological concepts in the domain application ontology based on the given text. Ontea can either find relevant classes and individuals in text or based on these classes and individuals create the individual of a given ontology class representing such text or the document. Found concepts and individuals are in that case assigned as properties of the document representing the individual. The semantic structure of the text can then be used for further computer processing, to improve different aspects of document processing such as searching, categorizing or reasoning. While most of annotation solutions try to improve web content with semantic tags, Ontea is more suitable in closed applications where problem domain is well defined by application ontology. The application area of Ontea is thus Knowledge Management systems, web and semantic bases application and context detection systems.

Annotation solutions can be divided to manual and semi-automatic. This different strategy depends on a use of the annotation. Usually in the manual

annotation solutions annotating is understood as 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, while several manual 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 markup 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]. There are also other annotation tools and approaches such as CREAM [15] or Magpie [16] which follow the idea to provide users with useful visual tools for manual annotation, web page navigation, reading of semantic tags and browsing [18] or provide infrastructure and protocols for manual stamping documents with semantic tags [1][2][3].

Semi-automatic solutions usually follow a different priority, which is creating of semantic metadata for further computer processing, using semantic data in knowledge management [17] or in semantic driven problem specific applications [8] [9] [12]. Semi-automatic approaches are based on natural language processing [21] [22], document structure analysis [23] or learning which require learning sets and need to be trained and supervised [24]. Moreover, other semi-automatic solutions such as PANKOW [19] and C-PANKOW [20] exist using pattern matching and Google API for automatic annotation, which is similar to Ontea but algorithm seems to be quite slow when annotating a large number of documents (there is no evaluation of performance but description of algorithm with frequent connections to Google API does not seem to be fast enough) which is needed in knowledge management or semantic driven problem specific applications.

Ontea [10] 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 with the Ontea annotation engine we want to achieve the following objectives:

- Detect meta data from the text
- Prepare improved structured data for later computer processing
- Structured data are based on the existing application ontology model

The Ontea annotation was created as one of tools is the NAZOU project [9]. 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 [25].

2 Methodology and the Approach

Ontea tool analyzes a text using a regular expression patterns and detects equivalent semantic 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.

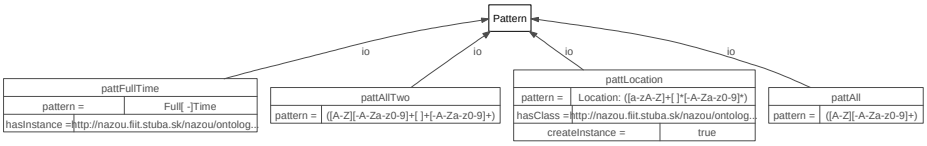


Fig. 1. Pattern ontology with several individuals from Job Offer domain ontology

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 literal properties of such individuals. The properties of *Pattern* class are *hasClass.Pattern*, *hasInstance.Pattern*, *pattern.Pattern*, *pattern.createInstance* (1).

$$\begin{aligned} &Pattern \subseteq \\ &\quad hasClass.Pattern(Thing) \cap \\ &\quad hasInstance.Pattern(\{Thing\}) \cap \\ &\quad pattern.Pattern(String) \cap \\ &\quad pattern.createInstance(Boolean) \\ &\{pattern\} \in Pattern \end{aligned} \tag{1}$$

The instances of the *Pattern* class are used to define and identify relations between the text 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 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 or SeRQL 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 regular expression or keyword is not found in existing ontology model and its meta-data, such individual of *hasClass* type is created.

The underlying principles of the algorithm are the following:

1. The text of a document is loaded.
2. The text is processed by the 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 processed 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 as well as for all detected individuals.

The algorithm also uses inference in order to 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 to 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 properties type. Crucial steps of the algorithms as well as inputs and outputs can be seen also on Figure 2.

2.1 Regular Expression Patterns

Regular expression patterns are the key element of the Ontea algorithm. Usually for each problem domain we need to define new, problem specific patterns to match the ontology elements in the text. However several cross application patterns exist:

– Matching one word starting with capital letter: *([A-Z][-A-Za-z0-9]+)*
– Two words pattern: *([A-Z][A-Za-z0-9]+[\s]+[A-Z][A-Za-z0-9]+)*
– Similar for 3 and 4 words pattern.

When individuals in the reference domain ontology contain plain text labels describing or naming the individuals, they can be detected by Ontea using the

patterns mentioned above. Even when using such simple patterns, we can achieve satisfactory results. This is demonstrated in the last chapter. If the reference ontology contains critical amount of individuals with assigned text labels, results of annotation are satisfactory with above mentioned cross application patterns. A good example is the location ontology, which is used in both examples provided in this paper. The location ontology contains concepts such as regions, countries, settlements, mountains, rivers or lakes as well as individuals of such classes. It is possible to create such ontology with concrete individuals of towns, settlements, mountains or rivers. Such data are available on the internet [14]. We have also created such ontology containing all geographical data for Slovakia.

When keywords such as "Danube" or "Bratislava" appear in the text, correct individuals are detected by Ontea, where the Danube is an individual of a river and Bratislava is an individual of the Capital subclass of the settlement and the town. Similar detection can be achieved also with concepts such as a skill, a company or a category when ontology consists of a critical mass of individuals.

As already described, Ontea not only detects but also creates individuals if patterns are set up that way. A good example is again the location ontology. In many web pages for instance job offers, location is referenced by text "Location: City or Region Name". When we convert web pages to a plain text, a regular expression pattern can be easily set up as follows: *Location:[\s]*([A-Z][-a-zA-Z]+[\s]*[A-Za-z0-9]*)*. This will match one or two words after the "Location:" string. If document contains e.g. "Location: New York" and "New York" text is not found in any individual in the reference ontology, we can create a new simple individual of the region type (this is set up in hasClass property of pattern) with rdfs:label "New York". In the future if New York is found in another document, the same individual is detected. Note that if we would create e.g. "New York City" individual, one can be sure that New York City is not a region but rather it's subclass - the town in our location ontology. If we would change manually such individual to be an individual of the town class, it would be updated automatically in all detected data.

We think that with Ontea it is possible to detect or create ontology elements within the reference ontology with satisfactory results. This can be achieved automatically or semiautomatically, when an expert can review and update the results.

3 Architecture and Technology

Architecture of the system (Figure 2) contains similar elements as the main annotation algorithm described above. Inputs are documents or text resources (HTML, email, plain text) which need to be annotated as well as corresponding domain ontology with defined patterns individuals (Figure 1). The output is an individual, which represents the annotated text. Properties of this individual are filled with detected ontology individuals according to defined patterns.

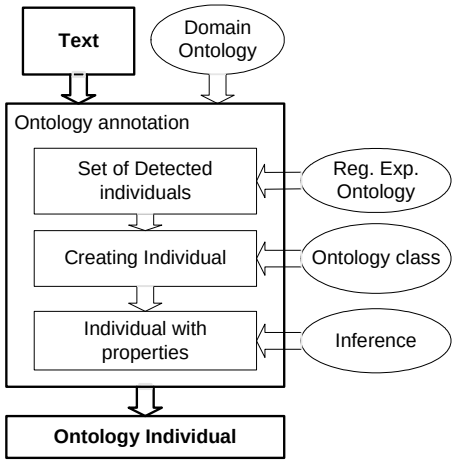


Fig. 2. Ontea Tool Architecture

Ontea works with RDF/OWL Ontologies [4]. It is implemented in Java using Jena Semantic Web Library [5] or Sesame library [6]. In both implementation inference is used to achieve better results.

4 Examples of Use

Ontea has been deployed in the K-Wf Grid [8], NAZOU [9] and Raport [12] 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 when users use text notes for collaboration and knowledge sharing [7]. Notes are showed to the user in appropriate context, which is detected by Ontea. This is useful in Flood forecasting application [11]

In NAZOU project, Ontea is used for creating semantic version of offer document. For testing purposes job offers domain was used, where the success rate of the Ontea algorithm was measured. In Raport project [12], Ontea detects formalized context of emails. Context is then used to display relevant notes [7] (similar as in K-Wf Grid project) in outgoing and incoming email communication. A user can receive email with additional information (text attachments) at the end or beginning of the email message. This information can contain relevant categories, hints or links (similar as Gmail [13]) to related organizational resources such as document repositories, databases or information systems.

4.1 Use of the Ontea in Job Offer Application

NAZOU (Research and Development of Tools for Knowledge Discovery, Maintenance and Presentation, SPVV 1025/04) 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.

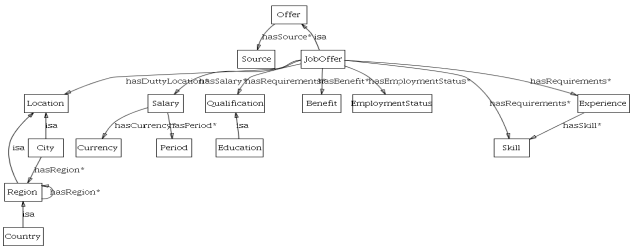
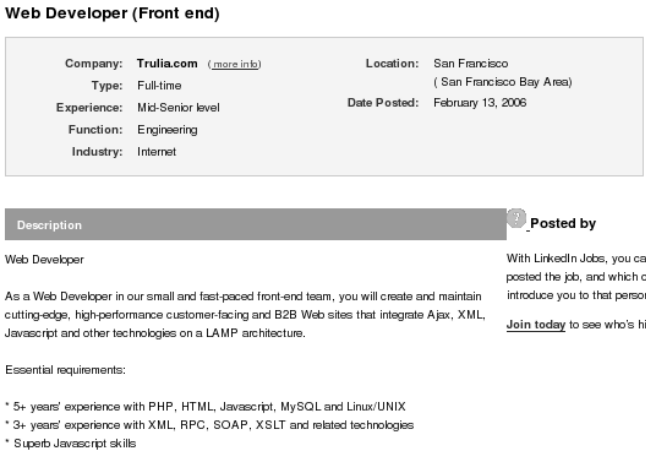


Fig. 3. Job Offer Ontology

On Figure 3 main components of Job Offer ontology are displayed. Important fragments on ontology are Location or Skills individuals, which can be then detected by annotation.



Web Developer (Front end)

Company:	**Trulia.com** (more info)	**Location:**	San Francisco (San Francisco Bay Area)
Type:	Full-time	**Date Posted:**	February 13, 2006
Experience:	Mid-Senior level		
Function:	Engineering		
Industry:	Internet		

Description

Web Developer

As a Web Developer in our small and fast-paced front-end team, you will create and maintain cutting-edge, high-performance customer-facing and B2B Web sites that integrate Ajax, XML, Javascript and other technologies on a LAMP architecture.

Essential requirements:

* 5+ years' experience with PHP, HTML, Javascript, MySQL and Linux/UNIX
* 3+ years' experience with XML, RPC, SOAP, XSLT and related technologies
* Superb Javascript skills

Posted by

With LinkedIn Jobs, you ca
posted the job, and which c
introduce you to that perso

Join today to see who's hi

Fig. 4. Web Document of Job Offer

On figure 5 the individual of the Job Offer is created based on the semantic annotation of a Job Offer document (figure 4), using simple regular expression patterns as showed on Figure 1 where main individuals can be detected by the title property such as skillXML or skillPHP individuals. Other specialized patterns such as pattFullTime are used to detect concrete job offer properties - jtPermanent individual, which represents a permanent job position. In this example the job offer location - San Francisco id identified by a regular expression, *([-A-Za-z0-9]+ [ ]+[-A-Za-z0-9]+)*, because individual locSF has the property title ‚San Francisco". Similarly, other ontology elements are detected. Some regular expressions search for ontology individuals, other ontology classes and others such as pattFullTime annotate a job offer by a concrete individual jtPermanent if expression *(Full[-]Time)* is found. Systems detect ontology elements based on domain ontology. In this example it is ontology of job offers.

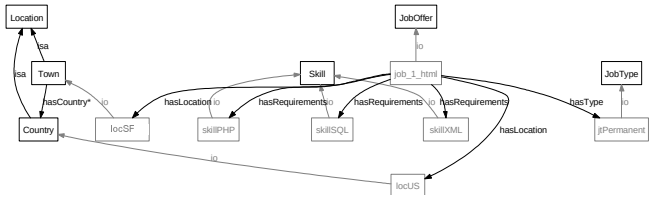


Fig. 5. Job Offer Individual Created by Ontea

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.

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.

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

Table 1. The comparison of results computed using the Ontea tool with reference data - selection of 500 job offers. This selection is used as a random sample ($x_1,\ x_2,\ ...,\ x_{500}$). The success rate of the Ontea tool (in percent) will be considered to be a random variable X with cumulative distribution function (CDF) F(x). 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

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

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

As the success rate of Ontea tool is a random variable, we can create probability and cumulative distribution functions p(x) and F(x). The values of these functions are shown in the following table:

Table 2. Frequency, probability function and cumulative distribution function of random variable X.

Success rate % x	25%	50%	75%	100%
Frequency	3	48	136	313
Probability *Pr(X=x)*	0,006	0,096	0,272	0,626
CDFF(x) *F(x)*	0,006	0,102	0,374	1

From distribution function we can calculate the probability that the success rate of Ontea tool is higher than 75%, which is:

$$\Pr\left(X > 75\%\right) = 1 - \Pr\left(X \leq 75\%\right) = 1 - F\left(75\%\right) = 0,626 \tag{5}$$

The probability that success rate of Ontea is higher than 50% is:

$$\Pr\left(X > 50\%\right) = 0,898 \tag{6}$$

The most probable value of the success rate of Ontea tool (modus) is 100% (with the probability of 0,626). Figure 6 displays histogram and distribution function F(x) for random variable X (success rate of Ontea).

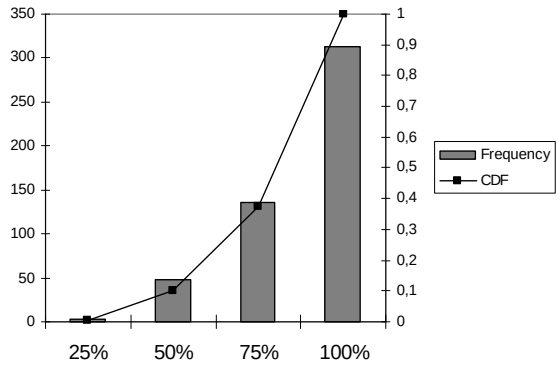


Fig. 6. Frequency histogram and cumulative distribution function of success rate. Left axis is frequency histogram and right axis is a cumulative distribution function.

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. In other cases Ontea detected for example "Financial Services" as an extra category for the job because relevant text was found in the job offer. We can say that this was added value to the job offer because it was relevant to text of the offer, however different cases may occur, when the found text is irrelevant; as an example can serve the "PHP" language detected as relevant needed expertise which could be detected from an advertisement or a header. We can say that 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 when extra found data are relevant or irrelevant.

The basic sample characteristics can be calculated according to above equations (2) (3) (4). If the extra results are all relevant, the values are:

$$\bar{x} = 88,523\%,\ s^2 = 1,592\%, s = 12,619\% \tag{7}$$

$$\bar{x} = 56,48\%, s^2 = 3,298\%, s = 18,162\% \tag{8}$$

6 Conclusions and Future Work

The described solution is used in the K-Wf Grid [8], NAZOU [9] and Raport[12] projects to detect relevant structured knowledge described by a domain specific ontology model in an unstructured text. The most similar annotation solution to Ontea is the PANKOW [19] [20]. We think that Ontea is more simple, faster (however performance was not compared) solution with better success rate, suitable for knowledge management or semantic driven embedded applications. The archived results are quite satisfactory since the Ontea tool works with an average success over 80In our future work we will try to evaluate the algorithm on different application domains using new strategies where Ontea will not only detect but also create new property individuals and we will evaluate also the performance of the tool.

Acknowledgment This work was partially supported by the Slovak State Programme of Research and Development "Establishing of Information Society" under the contract No. 1025/04, K-Wf Grid EU RTD IST FP6-511385 and RAPORT APVT-51-024604 projects.

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. W3C: Web Ontology Language OWL, 2005, http://www.w3.org/TR/owlfeatures/
5. HP Labs and Open Source Community, Jena Semantic Web Library, 2005, http://www.sf.net/
6. OpenRDF.org, Sesame RDF Database, 2006, http://www.openrdf.org/
7. Laclavik M., Gatial E., Balogh Z., Habala O., Nguyen G., Hluchy L.: Experience Management Based on Text Notes (EMBET); Proc. of eChallenges 2005 Conference, 19 - 21 October 2005, Ljubljana, Slovenia, Innovation and the Knowledge Economy, Volume 2, Part 1: Issues, Applications, Case Studies; Edited by Paul Cunnigham and Miriam Cunnigham; IOS Press, pp.261-268. ISSN 1574-1230, ISBN 1-58603-563-0.
8. K-Wf Grid Consortium: K -Wf Grid IST Project Website, 2005, http://www.kwfgrid.net/
9. NAZOU Project Website, 2006, http://nazou.fiit.stuba.sk/
10. Laclavik M., Seleng M., Gatial E., Balogh Z., Hluchy L.: Ontology based Text Annotation - OnTeA; In: Proc. of 16-th European-Japanese Conference on Information Modelling and Knowledge Bases, EJC'2006, Y.Kiyoki et.al. eds., 2006, Dept.of Computer Science, VSB - Technical University of Ostrava, pp. 280-284, ISBN 80-248-1023-9. Trojanovice, Czech Republic.
11. Hluchy L., Tran V.D., Habala O., Simo B., Gatial E., Astalos J., Dobrucky M.: Flood Forecasting in CrossGrid project. In: Grid Computing, 2nd European Across Grids Conference, Nicosia, Cyprus, January 28-30, 2004, LNCS 3165, Springer-Verlag, 2004, pp. 51-60, ISSN 0302-9743, ISBN 3-540-22888-8

12. RAPORT Project Website, 2005, http://raport.ui.sav.sk/
13. Google, Gmail, 2005, http://gmail.com/
14. GEOnet Names Server, Names, 2006, http://earth-info.nga.mil/gns/html/cntry_files.html
15. Handschuh S., Staab S.: Authoring and annotation of web pages in cream. In WWW '02: Proceedings of the 11th international conference on World Wide Web, pages 462-473, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511506.
16. Domingue J., Dzbor M.: Magpie: supporting browsing and navigation on the semantic web. In IUI '04: Proceedings of the 9th international conference on Intelligent user interface, pages 191-197, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-815-6. doi: http://doi.acm.org/10.1145/964442.964479.
17. Uren V., Cimiano P.,Iria J., Handschuh S., Vargas-Vera M., Motta E.,Ciravegna F.: 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 World Wide Web, 4(1):14-28, 2005.
18. Uren V., Motta E, Dzbor M.,Cimiano P.: Browsing for information by highlighting automatically generated annotations: a user study and evaluation. In K-CAP '05: Proceedings of the 3rd international conference on Knowledge capture, pages 75-82, New York, NY, USA, 2005b. ACM Press. ISBN 1-59593-163-5. doi: http://doi.acm.org/10.1145/1088622.1088637.
19. Cimiano P., Handschuh S., Staab S.: Towards the self-annotating web. In WWW '04: Proceedings of the 13th international conference on World Wide Web, pages 462-471, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988735.
20. Cimiano P., Ladwig G.,Staab S.: Gimme' the context: context-driven automatic semantic annotation with c-pankow. In WWW '05: Proceedings of the 14th international conference on World Wide Web, pages 332-341, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-046-9. doi: http://doi.acm.org/10.1145/1060745.1060796.
21. Madche A., Staab S.: Ontology learning for the semantic web. IEEE Intelligent Sys-tems, 16(2):72-79, March/April 2001.
22. Charniak E., Berland M.: Finding parts in very large corpora. In Proceedings of the 37th Annual Meeting of the ACL, pages 57-64, 1999. Semi-automatic CREAtion of Metadata. In Proceedings of EKAW 2002, LNCS, pages 358-372, 2002.
23. Glover E., Tsioutsiouliklis K., Lawrence S., Pennock D., Flake G.: Using web structure for classifying and describing web pages. In Proceedings of the Eleventh International Conference on World Wide Web, pages 562-569. ACM Press, 2002.
24. Reeve L., Hyoil Han: Survey of semantic annotation platforms. In SAC '05: Proceedings of the 2005 ACM symposium on Applied computing, pages 1634-1638, New York, NY, USA, 2005. ACM Press. ISBN 1-58113-964-0. doi: http://doi.acm.org/10.1145/1066677.1067049.
25. Návrat, P. - Bieliková M. - Rozinajová, V.: Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: CompSysTech 2005, B. Rachev, A. Smrikarov (Eds.), Varna, Bulgaria, June 2005. - pp. IIIB.7.1-IIIB.7.6.