Empowering Automatic Semantic Annotation in Grid

Michal Laclavík, Marek Ciglan, Martin Šeleng and Ladislav Hluchý

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. Nowadays, capturing the knowledge in ontological structures is one of the primary focuses of the semantic web research. To exploit the knowledge from the vast quantity of existing unstructured texts available in natural languages in ontologies, tools for automatic semantic annotation (ASA) are heavily needed. In this paper, we present the ASA tool Ontea and empowering of the method by Grid technology for performance increase, which help us in delivering formalized semantic data in shorter time. We have adjusted Ontea annotation algorithm to be executable in the distributed grid environment. We also give performance evaluation of Ontea algorithm and experimental results from cluster and grid implementation.

Key words: semantic annotation, grid, Ontea

1 Introduction

Automated annotation of the Web documents is a key challenge of the Semantic Web [1] effort. Web documents are structured but their structure is understandable only for humans, which is the major problem of the Semantic Web. In this paper, we present a tool for automatic semantic annotation named Ontea. We describe the methods implemented in the tool and present the results of experimental evaluation. Although Ontea's results are promising, we face the problem of the performance, as the process of Ontea's annotation is rather time consuming. This led us to the adoption of the computational grid technology, which allows us to reduce the delivery time of annotation results. The distributed version of Ontea is described in the paper and experimental evaluation of the time savings is presented.

1.1 Related Work

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 [2] or Magpie [3] which follow the idea to provide users with useful visual tools for manual annotation, web page navigation, reading semantic tags and browsing [4] or provide infrastructure and protocols for manual stamping documents with semantic tags such as Annotea[5], Ruby[6] or RDF annotation[7]. Semi-automatic solutions focus on creating semantic metadata for further computer processing, using semantic data in knowledge management [8] or in information extraction application. Semi-automatic approaches are based on natural language processing [9] [10], a document structure analysis [11] or learning requiring training sets or supervision [12]. Moreover, other annotation approaches exist, e.g. KIM [14] which uses information extraction, or pattern-based semi-automatic solutions such as PANKOW and C-PANKOW [13], using Google API for automatic annotation. One of relevant automatic semantic annotation solution and the only one which runs on distributed architecture is SemTag [15]. It uses Seeker [15] information retrieval platform to support annotation tasks. SemTag annotates web pages using Stanford TAP ontology [16].

1.2 Ontea

One of pattern based solutions is Ontea [17] [18] working on text, in particular domain described by domain ontology and using regular expression patterns for semi-automatic semantic annotation. Ontea detects or create ontology elements/individuals within the existing application/domain ontology model according to defined patterns. Several cross application patterns are defined but in order to achieve good results, new patterns need to be defined for each application. Main difference e.g. with SemTag is that solution can also create, not only detect ontology individuals within defined ontology model. This functionality makes Ontea algorithm slower than SemTag [15], but suitable for wider scope of applications where semantics of documents is needed. Ontea is similar to C-PANKOW [13] annotation with better annotation results and better performance. By the Ontea annotation engine we want to achieve the following objectives:

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

1.3 Use of Information Retrieval Techniques

Ontea uses also RFTS (Rich full-text search) [19] - the tool for document indexing and document search. Ontea uses RFTS functionality when creating a new ontology individual for evaluating the relevance of the newly created instances. Ontea identifies part of text related to semantic context and match the subsequent sequence of characters to create an instance of the concept. Let us denote the sequence of words related to semantic context by C and word sequence identified as a candidate instance as I. We evaluate the relevance of the new instance by computing the ration of the close occurrence of C and I and occurrence of I in the whole collection of documents: *close_occurrence(C, I) / occurrence(I).*

The Ontea annotation method can be also used with Lucene [20] information retrieval library. When connected with RFTS indexing, Ontea asks for relevance based on words distance. When connecting with Lucene, Ontea asks for percentage of occurrence of matched regular expression pattern to detected element represented by word. Example can be *Google, Inc.* matched by pattern for company search: *[\s]+([-A-Za-z0-9][ ]*[A-Za-z0-9]*),[ ]*Inc[\.\s]+*, where relevance is computed as *Google, Inc.* occurrence divided by *Google* occurrence. RFTS indexing tool also supports this type of queries, however Lucene can achieve better performance.

Use of Lucene or RFTS is related to case *Ontea creation IR* in evaluation.

2 Evaluation

In this chapter we describe evaluation of Ontea annotation success rate as well as performance evaluation on a single machine. Performance evaluation on the grid infrastructure is not yet available but will be at time of the conference. We also describe a test set of documents.

2.1 Test Set of Documents

As reference test data, we used 500 job offers downloaded from web using wrapper which prepared us some structured data. This was converted to a defined ontology, manually checked and edited 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 reference, manually checked job offers ontology metadata.

2.2 Target Ontological Concepts for Identification

In this test, Ontea used simple regular expressions matching from 1 to 4 words starting with a capital letter. This experiment is referred to as *Ontea* in next chapter. In the second case we used domain specific regular expressions which identified locations and company names in text of job offers and Ontea also created individuals in knowledge base, while in the first case Ontea did not create extra new property individuals only searched for relevant individuals in knowledge base. This second case is referred to as *Ontea creation.* The third case used also previously described RFTS indexing tool or Lucene to find out if it is feasible to create a new individual using relevance techniques described earlier. This case is referred to as *Ontea creation IR.* To sum up, we conducted our experiments in 3 cases:

– *Ontea*: searching relevant concepts in knowledge base (KB) according to generic patterns

- *Ontea creation*: creating new individuals of concrete application specific objects found in text
- *Ontea creation IR*: Similar as previous with the feedback of RFTS or Lucene to get relevance computed above word occurrence. Individuals were created only when relevance was above defined threshold which was set up to 10%

We used following regular expressions:

- Generic expression matching one or more words in text. This was used only to search concepts in KB
  ([A-Z][-A-Za-z0-9]+[\s]+ [-a-zA-Z]+)
- Identifying geographical location in text and if not found in KB individual was created
  Location:[\s]*([A-Z][-a-zA-Z]+[ ]*[A-Za-z0-9]*) used for English
  [0-9]{3}[ ]*[0-9]{2}[ ]+([A-Z][^\s,\.]+[ ]*[^0-9\s,\.]*)[ ]*[0-9\n,]+ used for Slovak text where settlement name is usually next to ZIP code
- Identifying company in the text, this was used also with other abbreviations such as "Ltd" or "a.s.", "s.r.o." for the Slovak language
  [\s]+([-A-Za-z0-9][ ]*[A-Za-z0-9]*),[ ]*Inc[\.\s]+ for English
  [\s]+([A-Z][^\s,\.]+[ ]*[^\s,\.]*[ ]*[^\s,\.]*)[,]*s\.r\.o\.[\s]+ used for Slovak texts

2.3 Success Rate of the Ontea Algorithm

In this chapter we discuss the algorithm evaluation and success rate. To evaluate success of annotation, we used the standard recall, precision and F_1 measures. 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 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 [21]. To obtain a better measure to describe performance, we use the F_1 measure (first introduced by van Rijsbergen [22]) which combines precision and recall measures, with equal importance, into a single parameter for optimization. F_1 measure is weighted average of the precision and recall measures.

2.4 Experimental Results of Annotation Success Rate

Experimental results using precession, recall and F_1-measures are in a table bellow. In the table we compare our results with other semantic annotation approaches and we also list some advantages and disadvantages. The column *relevance* is in case of Ontea F_1-measures but in case of other methods it can be evaluated by other techniques and usually it is not common. For example for C-PANKOW, relevance is referred as recall. Rows relevant to our annotation

	Method	relevance %	precision %	recall %	Disadvantages	Advantages
Ontea	regular expresions, search in knowledge base (KB)	**71**	**64**	**83**	high recall, lower precesion	high succes rate, generic solution, solved duplicity problem, fast algorithm
SemTag	disambiguatiy check, searching in KB	high	high		works only for TAP KB	fast and generic solution
Ontea creation	regular expresions (RE), creation of individuals in KB	**41**	**28**	**81**	aplication specific patterns are needed low precision	support Slovak language
Ontea creation IR	RE, creation of individuals in KB + RFTS or Lucene relevance	**62**	**53**	**79**	some good results are killed by relevance identification	disambiguities are found and not annotated, good results
Wrapper	document structure	high	high		zero success with unknown structure	high success with known structure
PANKOW	pattern matching	59			low success rate	generic solution
C-PANKOW	POS taging and pattern matching Qtag library	74		74	suitable only for English, slow algorithm	generic solution
Hahn et al.	semantic and syntactic analysis	76			works only for English not Slovak	
Evans	clustering	41			low success rate	
Human	manual annotation	high	high	high	problem with creation of individuals duplicities, inacuracy	high recall and precesion

Table 1. Annotation experimental results

approach are in grey color, where we show success rate of three evaluation cases mentioned in the previous chapter.

The row *Ontea creation IR* case is the most important considering evaluation where we combined information retrieval (IR) and annotation techniques. By using this combination we could eliminate some not correctly annotated results. For example by using _[Cc]ompany[:\s]*([A-Z][-A-Za-z0-9][ ]*[A-Za-z0-9]*_ regular expression in the second case we have created and identified companies such as *This position* or *International company* which were identified as not relevant in the third case with use of IR. Similarly *Ontea creation* identified also companies as Microsoft or Oracle which is correct and in combination with IR eliminated. This issue decreases recall and increases precession. Here it seems that IR case is not successful but opposite is true because in many texts Microsoft is identified as products e.g. *Microsoft Office* and if we take more text to annotate it is better not to annotate Microsoft as a company and decrease recall. If we would annotate Microsoft as a company in other texts, used in context of *Microsoft Office* we will decrease precision of annotation. It means it is very powerful to use presented annotation technique in combination with indexing in applications where precession need to be high.

2.5 Performance Evaluation on Single Machine

In this section we will evaluate performance of our solution on the same document set as described above. Annotation was executed on Centrino notebook 1.66 GHz, 1GB RAM, Linux OS. The algorithm is written in Java using Sesame [23] semantic engine. An average size of documents from test set was from 2000 to 4000 bytes. We will measure the average time of document annotating and duration of the single ontology query to knowledge base. This depends on the ontology and knowledge base size but in our case this can be omitted. In the future evaluation on a larger set of document we will need to take this into account, since query duration will increase by each individual added to the ontology.

First we run the test on the empty ontology with only regular expressions which can create individuals (see *Ontea creation* case). If the individual returned by the regular expression exists in the ontology system, continue to the next regular expression result (query) but if the result do not exist in the ontology, a new individual will be created by the system. Stats after annotating of 500 documents by the system are in the following table

Total time duration	Average time duration per document	Total queries executed	Average executed queries per document	Total individuals created	Average created individuals per document
220 s	439 ms	1533	3	1241	2.482

Table 2. Performance in Ontea Creation case

Next test has been run on the results ontology from previous case. Now we search for ontology individuals within knowledge base and not creating them (see *Ontea* case)

Total time duration	Average time duration per document	Total queries executed	Average executed queries per document	Total expressions found	Average found expression per document
2399 s	4797 ms	103570	207	4984	10

Table 3. Performance in Ontea case

In the tables above we can see performance on document set on one machine. Both tables describe annotation tasks, which can be put to Grid environment. Annotation of one document takes approximately 5 seconds.

3 Putting Ontea into Grid

While Ontea yields interesting results, its annotation method is rather time consuming and annotating large number of documents or periodical re-annotating of updated document collection is highly impractical when performing the computation on a single server. The process of Ontea computation is easy parallelized and parallel threads can run independently with only two synchronization steps needed. This allows us to use distributed computing technology to speed up annotation process for large document collections. In this section, we describe the technique we use for Ontea annotation method distribution, exploiting existing grid computing infrastructure.

3.1 Annotation Process Distribution

Ontea semantic annotation is performed in the following stages:

1. In the first stage, the instances of ontological concepts are created in the input text collection based on regular patterns matching. (case Ontea Creation) This stage will produce OWL ontology files which need to be integrated on a central machine.
2. After integration, instances created in the first stage are evaluated by computing their relevance using IR techniques. The instances with relevance value above given threshold are identified as relevant and filled in result domain ontology OWL file. (case Ontea creation IR)
3. Domain ontology OWL file is used in the third stage of the process for searching annotation tags within annotated text similarly to step one but using general keyword matching patterns. This results to executing more ontology queries and thus consuming more time. (case Ontea in evaluation)
4. Last stage integrated produced semantic metadata to one knowledge base represented by OWL file. The distribution of the computation is straightforward, it is sufficient to split the document collection and run the annotation process in a distributed manner, where different computation jobs process distinct subsets of the collection. The distribution is possible for the first and third stage of the annotation process, the second and forth stage is dedicated to integration of the jobs results and is performed on a single node. In the first step, ontology individuals are created. They are created with unique RDF ID based on the detected text and timestamp (e.g. http://...inst#region_San_Francisco_1179239158496). In case of distribution of annotation algorithm, same individuals can be created on different nodes. We had to change ID creation to be same at all nodes, due to final integration. Thus timestamp value was replaced by hash of detected text and ontology class URI.

3.2 Data Management Aspects

Following data have to be managed for running Ontea text annotation:

- executables and libraries
- document collection for text annotations
- domain ontology OWL files

In the first stage, the jobs for pattern matching are submitted to the grid. The subset of document collections must be replicated from central storage to grid sites where the jobs will be executed. After all of the first stage jobs are finished, the results are transferred to a central server, where the integration takes place.

To reduce transfer of collection data in the infrastructure, it would be advantageous to submit the third phase jobs to the same grid sites where the first stage jobs were executed. However, limiting sites for third phase jobs only to those sites where document collection subsets are replicated might be impractical when those sites are heavily loaded and the jobs would be queued for a long time. We address this problem by using job management routine which submits the jobs only to given subset of grid sites, after defined timeout it cancels those that are still in scheduled state and resubmits the jobs without restriction to the grid sites. In addition, the OWL ontology files have to be replicated to the execution sites of first and third annotation stage jobs.

3.3 Evaluation Environment

We have developed a distributed version of Ontea using the grid infrastructure and the approach was evaluated in the Gilda testbed [26] training and test grid infrastructure of EGEE [27] project. The grid testbed used for running Ontea is powered by gLite middleware and gridified version of Ontea is tailored to this grid middleware. Worker nodes were Intel(R) Pentium(R) 4 CPU 2.40GHz PCs. As the sites in Gilda testbed are usually low loaded, we have not experience any wait times in the batch system queues for sub-jobs of Ontea computation. We have experienced an average 10 minutes overhead from the grid middleware (the overhead includes the procedure of submitting job via resource broker, scheduling job from resource broker to the execution site, job processing in local batch system, transfer and unpacking of the input data set, notification of resource broker about the state of processing and retrieval of the results from the worker node after the processing is done.)

Total time duration on Grid	Average time duration per document	Total time on single node	Grid overhead	Results integration overhead	Number of documents
173 min	8.1 sec	678 min	10 min	25 min	5000

Table 4. Performance on the Grid

The evaluation was performed on the set of 5000 documents, split to five groups of approximately same size; five jobs were submitted to the grid infrastructure, each for one input data set. The same setting was used for 10 experiment runs. Average result delivery time for the whole input set was 173 minutes, with approximately 10 minutes grid middleware overhead and 25 minutes for results integration. The single machine processing of the whole input set took 678 minutes. The annotation of one document took approximately 8 seconds. This is longer than evaluation on a single machine due to a slower machine in testbed as well as changes in the code when allowing it for a grid. The utilization of the grid infrastructure for Ontea data processing brings us near linear speed-up of the results delivery, especially if used in real application where higher number of documents is annotated.

4 Conclusion and Future Work

In this paper we discussed how automatic semantic annotation solution Ontea can benefit from Grid environment to achieve faster results of semantic annotation. The Ontea method is quite a powerful method [17] with acceptable success rate and performance suitable for knowledge management applications or applications in organizations where semantics of documents is needed [8]. If applied on bigger document bases, performance is not acceptable and need to be improved. To our best knowledge, none of annotation solutions yet benefit of Grid infrastructure and only SemTag [15] annotation used distributed architecture to perform annotation tasks. We have shown that semantic annotation can be gridified and achieve its results much faster. Semantic web research can be successful only when number of annotated web documents will reach critical mass, we believe this can happen only when annotation solution will benefit from parallel and distributed computing which is widely used in the information retrieval field.

In our future work we would like to parallelize Ontea algorithm using MapReduce algorithm [24] and its open source implementation Hadoop [25].

Acknowledgments This work is supported by projects NAZOU SPVV 1025/2004, VEGA 2/6103/6, VEGA 2/7098/27, EGEE-II EU 6FP RI-031688 and RAPORT APVT-51-024604.

References

1. Berners-Lee T., Hendler J., Lassila O.: Semantic web. Scientific American, 1 (2000) 68–88
2. 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, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511506 (2002) 462–473
3. 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, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-815-6. doi: http://doi.acm.org/10.1145/964442.964479 (2004) 191–197
4. 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, New York, NY, USA, 2005b. ACM Press. ISBN 1-59593-163-5. doi: http://doi.acm.org/10.1145/1088622.1088637. (2005) 75–82
5. Annotea Project, http://www.w3.org/2001/Annotea/, (2001)
6. W3C, Ruby Annotation, http://www.w3.org/TR/ruby/ (2001)

7. The Institute for Learning and Research Technology, RDF Annotations, http://ilrt.org/discovery/2001/04/annotations/ (2001)
8. 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) (2005) 14–28
9. Madche A., Staab S.: Ontology learning for the semantic web. IEEE Intelligent Sys-tems, 16(2) (2001) 72–79.
10. 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 (2002) 358–372
11. 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, ACM Press (2002) 562–569
12. 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, ACM Press. ISBN 1-58113-964-0. doi: http://doi.acm.org/10.1145/1066677.1067049. (2005)
13. 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, New York, NY, USA. ACM Press. ISBN 1-59593-046-9 (2005) 332–341
14. Kiryakov A., Popov B., Terziev I., Manov D., Ognyanoff D.: Semantic Annotation, Indexing, and Retrieval. Elsevier's Journal of Web Semantics, Vol. 2, Issue (1), http://www.ontotext.com/kim/semanticannotation.html (2005)
15. Dill S., Eiron N., et al.: A Case for Automated Large-Scale Semantic Annotation; Journal of Web Semantics (2003)
16. R.Guha and R. McCool. Tap: Towards a web of data. http://tap.stanford.edu/.
17. Laclavik M., Seleng M., Gatial E., Balogh Z., Hluchy L.: Ontology based Text Annotation OnTeA; Information Modelling and Knowledge Bases XVIII. IOS Press, Amsterdam, Marie Duzi, Hannu Jaakkola, Yasushi Kiyoki, Hannu Kangassalo (Eds.), Frontiers in Artificial Intelligence and Applications, Vol. 154, ISBN 978-1-58603-710-9, ISSN 0922-6389, (2007) 311–315
18. NAZOU Project Website, http://nazou.fiit.stuba.sk/ (2006)
19. Ciglan M.: Documents Content Indexing for Supporting Knowledge Acquisition Tools, In: Tools for Acquisition, Organisation and Presenting of Information and Knowledge. P.Navrat et al. (Eds.), Vydavatelstvo STU, Bratislava, ISBN 80-227-2468-8. (2006) 49–63
20. Hatcher E., Gospodnetic O., Lucene in Action, Manning, ISBN: 1932394281, (2005)
21. Cleverdon, C. W.; Mills, J.; Keen, E. M.: Factors determining the performance of indexing systems. Vol. 1-2. Cranfield, U.K.: College of Aeronautics. (1966)
22. C. J. Van Rijsbergen, Information Retrieval, Butterworth-Heinemann, Newton, MA (1979)
23. OpenRDF.org, Sesame RDF Database, http://www.openrdf.org/ (2006)
24. Dean J., Ghemawat S.: MapReduce: Simplified Data Processing on Large Clusters, Google, Inc. OSDI'04, San Francisco, CA (2004)
25. Lucene-hadoop Wiki, HadoopMapReduce, http://wiki.apache.org/lucene-hadoop/HadoopMapReduce (2007)
26. GILDA, Grid Infn Laboratory for Dissemination Activities, https://gilda.ct.infn.it/ (2007)
27. EGEE, Enabling Grids for E-sciencE, http://www.eu-egee.org/ (2007)