Towards Large Scale Semantic Annotation Built on MapReduce Architecture*

Michal Laclavík, Martin Šeleng, and Ladislav Hluchý

Institute of Informatics, Slovak Academy of Sciences,
Dúbravská cesta 9, Bratislava, 845 07
laclavik.ui@savba.sk

Abstract. 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 a human that is the major problem of the Semantic Web. Semantic Web can be exploited only if metadata understood by a computer reach critical mass. Semantic metadata can be created manually, using automated annotation or tagging tools. Automated semantic annotation tools with the best results are built on different machine learning algorithms requiring training sets. Another approach is to use pattern based semantic annotation solutions built on NLP, information retrieval or information extraction methods. Most of developed methods are tested and evaluated on hundreds of documents which cannot prove its real usage on large scale data such as web or email communication in enterprise or community environment. In this paper we present how a pattern based annotation tool can benefit from Google's MapReduce architecture to process large amount of text data.

Keywords: semantic annotation, information extraction, metadata, MapReduce.

1 Introduction

Automated annotation tools can provide semantic metadata for semantic web as well as for knowledge management [4] or other enterprise applications [11].

Pattern based automatic or semi-automatic solutions for semantic annotation or tagging are usually based on NLP, information retrieval or information extraction fields or minimally method algorithms common in the mentioned fields are applied.

Information Extraction - IE [1] is closed to semantic annotation or tagging by Named Entity recognition – NE defined by series of MUC conferences.

Semi automatic annotation approaches can be divided into two groups with regards to produced results [1]:

- identification of concept instances from the ontology in the text
- automatic population of ontologies with instances in the text

One of pattern based solutions for semi-automatic annotation is Ontea [2] [3] that uses regular expression patterns to detect or create instances in ontology. In our previous

* This work is supported by projects NAZOU SPVV 1025/2004, Commius FP7-213876, SEMCO-WS APVV-0391-06, VEGA 2/7098/27.

M. Bubak et al. (Eds.): ICCS 2008, Part III, LNCS 5103, pp. 331–338, 2008.

works [2] [3] we compared Ontea with other annotation methods and we conducted experiments to demonstrate its success rate above 60% that is comparable to well known annotation methods with easier applicability on concrete domain specific application due to relatively simple method built on regular expressions. This is another reason behind our decision to port Ontea into MapReduce architecture. We believe other well known semantic annotation or IE solutions such as C-PANKOW, KIM, GATE or different wrappers can be ported into MapReduce architecture. For survey on semantic annotation please see [4] [5] [14].

To our best knowledge the only semantic annotation solution which runs on distributed architecture is SemTag [6]. It uses the Seeker [6] information retrieval platform to support annotation tasks. SemTag annotates web pages using Stanford TAP ontology [7]. However, SemTag is able to identify but not create new instances in the ontology. Moreover, its results as well as TAP ontology are not available on the web for a longer period of time.

In our previous work we ported semantic annotation into Grid [3] with good results but with no easy and direct implementation and results integration. Thus we have focused on different parallel and distributed architectures.

Google's MapReduce [8] architecture seems to be a good choice for several reasons:

- Information processing tasks can benefit from parallel and distributed architecture with simply programming of Map and Reduce methods
- Architecture can process Terabytes of data on PC clusters with handling failures
- Most of information retrieval and information extraction tasks can be ported into MapReduce architecture, similar to pattern based annotation algorithms. E.g. distributed grep using regular expressions, one of basic examples for MapReduce, is similar to Ontea pattern approach using regular expressions as well.

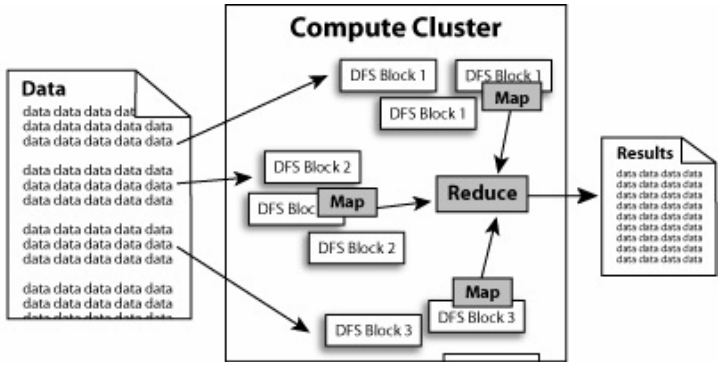

Fig. 1. MapReduce Architecture figure (source: Hadoop website)

On Figure 1 we can see main components of the MapReduce architecture: Map and Reduce methods, data in distributed file system (DFS), inputs and outputs. Several replicas of data are created on different nodes, when data are copied to DFS. Map tasks are executed on the nodes where data are available. Results of Map tasks are key value pairs which are reduced to results produced by Reduce method. All developer

need to do is implement Map and Reduce method and architecture will take care of distribution, execution of tasks as well as fault tolerance. For more details on MapReduce please see [8].

Two open source implementation of MapReduce are available:

- Hadoop [9], developed as Apache project with relation to Lucene and Nuch information retrieval systems, implemented in Java. Hadoop is well tested on many nodes. Yahoo! is currently running Hadoop on 10,000 nodes [15] in production environment [16].
- Phoenix [10], developed at Stanford University, implemented in C++.

In this paper we discuss work in progress - porting of pattern based semantic annotation solution Ontea into MapReduce architecture and its Hadoop implementation. We provide preliminary results on 8 nodes Hadoop cluster on email documents.

2 Ontea

The method used in Ontea [2] [3] is comparable particularly with methods such as those used in GATE, C-PANKOW, KIM, or SemTag. It process texts or documents of an application domain that is described by a domain ontological model and uses regular expressions to identify relations between text and a semantic model. In addition to having pattern implementation over regular expressions, created Ontea's architecture allows simply implementation of other methods based on patterns such as wrapers, solutions using document structure, language patterns similar to GATE, C-PANKOW and many others. Ontea [17] is being created as an Open source project under Sourceforge.net.

2.1 Ontea Scenarios and Results Examples

Current Ontea implementation can be executed in 3 different scenarios:

- *Ontea*: searching relevant individuals in knowledge base (KB) according to generic patterns
- *Ontea creation*: creating new individuals of objects found in text
- *Ontea IR*: Similar as previous with the feedback of information retrieval methods and tools (e.g. Lucene) to get relevance computed above word occurrence and decide weather to create instance or not.

Table 1. Examples of Instances and Patterns

#	Text	Instance	Patterns – regular expressions
1	Apple, Inc.	Company: Apple	*Company:* ([A-Za-z0-9]+)[,]+(Incl|Ltd)
2	Mountain View, CA 94043	Settlement: Mountain View	*Settlement:* ([A-Z][a-z]+[ ]*[A-Za-z]*)[ ]+[A-Z]{2}[ ]*[0-9]{5}
3	laclavik.ui@savba.sk	Email: laclavik.ui@savba.sk	*Email:* [-_.a-z0-9]+@[-_.a-zA-Z0-9]+\.[a-z]{2,8}
4	Mr. Michal Laclavik	Person: Michal Laclavik	*Person:* (Mr.|Mrs.|Dr.) ([A-Z][a-z]+ [A-Z][a-z]+)

New application scenarios can be created by combination of Result Transformers, which is discussed in next chapter.

2.2 Ontea Architecture

The fundamental building elements of the tool are the following java interfaces and extended and implemented objects:

- *ontea.core.Pattern*: interface for adaptation for different pattern search. Currently implemented pattern search uses regular expressions *PatternRegExp*.
- *onetea.core.Result*: a class representing annotation results by means object instance of defined type/class. Its extensions are different types of instances depending on implementation in ontology (Jena, Sesame) or as value and type pairs.
- *ontea.transform.ResultTransformer*: interface that after implementation contains different types of transformations among annotation results. Thus it can transforms set of results and include in transformation various scenarios of annotation such as relevance, result lemmatization, transformation of found value/type pairs (Table 1) into OWL instances in sesame or Jena API implementation. It is used to transform type value pairs into different type value pairs represented e.g. by URI or lemmatized text value. It can be also used to eliminate irrelevant annotation results.

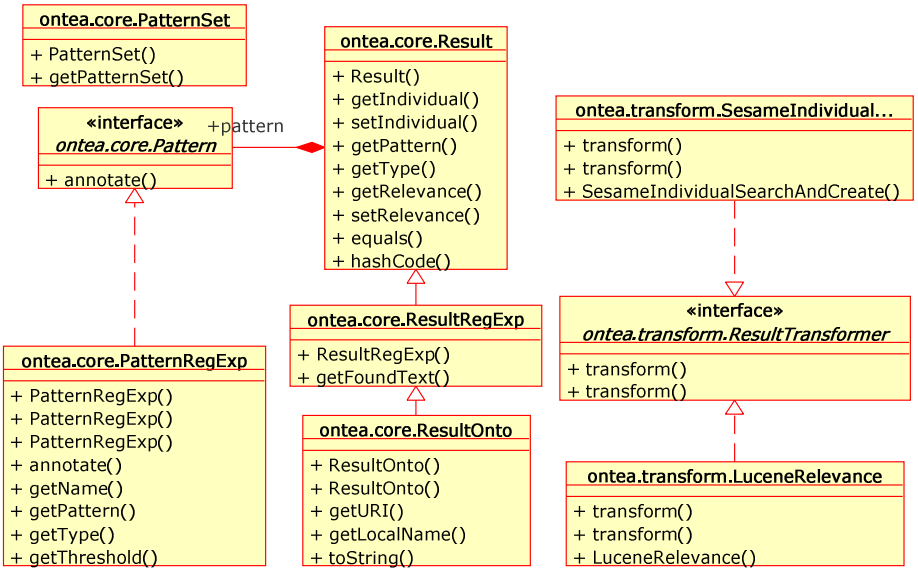

Fig. 2. Basic classes of Ontea platform

On the Figure 2 you can see *Result* class, *Pattern* and *ResultTransformer* interfaces. Such design allows extending Ontea for different patterns implementations or for the integrations of existing pattern annotation solutions. Also it is possible to implement various result transformations by implementing *ResultTransformer*, which can be used also as inputs and outputs between Map tasks in MapReduce architecture.

2.3 Integration of Ontea with External Tools

Ontea tool can be easily integrated with external tools. Some tools can be integrated by implementation of result transformers and other need to be integrated directly.

- *MapReduce*: Large scale semantic annotation using MapReduce Architecture – is main topic of this article. Integration with Hadoop requires implementation of Map and Reduce methods as described in next chapter.
- *Language Identification*: In order to use correct regexes or other patterns, often we need to identify language of use. For this reason it is convenient to integrate Ontea with language detection tool. We have tested Ontea with Nalit [11]. Nalit is able to identify Slovak and English texts as well as others if trained.

As already mentioned some integration can be done by implementing Result transformers:

- *Lemmatization:* When concrete text is extracted as representation of an individual, often we need to lemmatize found text to found or create correct instance. For example capital of Slovakia can be identified in different morphological forms: *Bratislava*, *Bratislave*, *Bratislavu*, or *Bratislovou* and by lemmatization we can identify it always as individual *Bratislava*. We have tested Ontea with Slovak lemmatizer Morphonary [12]. It is also possible to use lemmatizers or stemmers from Snowball project [18], where java code can be generated.
- *Relevance Identification*: When new instance is being created or found, it is important to decide on instance relevance. This can be solved using information retrieval methods and tools such as Lucene [19]. When connecting with Lucene, Ontea asks for percentage of occurrence of matched regular expression pattern to detected element represented by word on used document set. Document set need to be indexed by Lucene. 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. Use of Lucene is related to *Ontea IR* scenario and *LuceneRelevance* implementation of *ResultTransformer* interface. Similarly, other relevance algorithms such as cosine measure can be implemented. This was used for example in SemTag [6].
- *OWL Instance Transformation*: Sesame, Jena: Transformation of found key – value pairs into RDFS or OWL instances in Sesame or Jena API. With this integration, Ontea is able to find existing instances in knowledge base if existing and create new once if no instance found in DB. Ontea also use inference to found appropriate instance. For example if Ontea process sentence "Slovakia is in Europe." using pattern for location detection *(in|near) +(\\p{Lu}\\p{L}+)* following type value pair is detected *Location: Europe*. If we have Location ontology with Subclasses as Continents, Settlements, Countries or Cities and Europe is already present as instance of continent, Ontea can detect existing Europe instance in knowledge base using inference.

3 Ontea Ported into Hadoop

For porting Ontea or any semantic annotation solution it is important to understand results of annotations as well as how they can correspond to key/value pairs - outputs of Map and Reduce methods to be implemented in MapReduce architecture. In table 1 we show a simple example of Ontea possible annotation results such as settlements, company names, persons or email addresses. Used regular expressions are simplified to be more readable and understandable.

In the Map method, input is a text line which is processed by Ontea's regex patterns and outputs are key value pairs:

- Key: string starting with detected instance type and continue with instance value similar to *instance* row in table 1. This can be extended to return also instance properties e.g. address, phone or email as properties of company.
- Value: File name with detection of instance. It can be extended with position in file e.g. line number and character line position if needed.

Basic building blocks of Ontea are the following java classes and interfaces described earlier, which can be extended. Here we describe them in scope of MapReduce architecture:

- *ontea.core.Pattern*: interface for adaptation of pattern based searching in text. Main Pattern method *Pattern.annotate()* runs inside of Map method in MapReduce implementation.
- *onetea.core.Result*: a class which represents the result of annotation – an ontology instance. It is based on the type and value pairs as in table 1, *instance* column. Ontology results extension contains also URI of ontology individual created or found in ontology. Results are transformed into text keys as output of Map method in MapReduce implementation.
- *ontea.transform.ResultTransformer*: interface which transform results of annotation. Transformers are used in Map or Reduce methods in MapReduce implementation to transform individuals into OWL file or eliminate some results using Ontea IR scenario.

3.1 Ontea Running on Hadoop MapReduce Cluster

We wrapped up Ontea functionality into Hadoop MapReduce library. We tested it on Enron email corpus [20] containing of 88MB of data and our personal email containing of 770MB of data. We run same annotation patterns on both email data sets, on single machine as well as 8 node Hadoop cluster. We have used *Intel(R) Core(TM)2 CPU 2.40GHz* with *2GB RAM* hardware on all machines.

As you can see from Table 2, the performance increased 12 times on 16 CPUs in case of large data set. In case of smaller data set it was only twice faster then on single machine and MapReduce overhead is much more visible. In the table 2 we present only 2 concrete runs on 2 different datasets, but in reality we have executed several runs on these datasets and computational time was very similar so we can conclude that times presented in table 2 are very close to average.

Table 2. Performance and execution results

Description	**Enron corpus** (88MB)	**Personal email** (770MB)
Time on single machine	2min, 5sec	3hours, 37mins, 4sec
Time on 8 nodes hadoop cluster	1min, 6sec	18mins, 4sec
Performance increased	1.9 times	12 times
Launched map tasks	45	187
Launched reduce tasks	1	1
Data-local map tasks	44	186
Map input records	2,205,910	10,656,904
Map output records	23,571	37,571
Map input bytes	88,171,505	770,924,437
Map output bytes	1,257,795	1,959,363
Combine input records	23,571	37,571
Combine output records	10,214	3,511
Reduce input groups	7,445	861
Reduce input records	10,214	3,511
Reduce output records	7,445	861

In our tests we run only one Map method implementation and one Reduce method implementation. We would like to implement also passing Map results to another Map method as an input and thus fully exploit potential of *ResultTransformers* in Ontea architecture. However, we believe that this new tests does not change – decrees performance of semantic annotation on MapReduce architecture.

4 Conclusion and Future Work

In this paper we discussed briefly how pattern based semantic annotation could benefit from MapReduce architecture to process a large collection of data. We demonstrated how Ontea pattern solution could be ported to implement basic Map and Reduce methods. Furthermore we provided preliminary results on 8 node Hadoop cluster. As we can see from preliminary results, performance on large datasets is very reasonable on Hadoop. MapReduce architecture is scalable to thousands machines. We believe semantic annotation can be successful only if able to annotate or tag large collections of documents.

In our future work we would like to test MapReduce also on several Map tasks in a row and publish implemented code under Ontea.sourceforrge.net project. We also want to use MapReduce architecture to solve concrete application domains such as geographical location identification of web pages and large scale email processing to improve automated email management and semantic searching.

References

1. Cunningham, H.: Information Extraction, Automatic. Encyclopedia of Language and Linguistics, 2nd edn. (2005)
2. Laclavik, M., Seleng, M., Gatial, E., Balogh, Z., Hluchy, L.: Ontology based Text Annotation OnTeA; Information Modelling and Knowledge Bases XVIII. Frontiers in AI, vol. 154, pp. 311–315. IOS Press, Amsterdam (2007)
3. Laclavik, M., Ciglan, M., Seleng, M., Hluchy, L.: Ontea: Empowering Automatic Semantic Annotation in Grid. In: Wyrzykowski, R., et al. (eds.) PPAM 2007, LNCS, vol. 4967, Springer, Heidelberg (2008)
4. 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 4(1), 14–28 (2005)
5. Reeve, L., Han, H.: Survey of semantic annotation platforms. In: SAC 2005: Proceedings of the 2005 ACM symposium on Applied computing, pp. 1634–1638. ACM Press, New York (2005)
6. Dill, S., Eiron, N., et al.: A Case for Automated Large-Scale Semantic Annotation. Journal of Web Semantics (2003)
7. Guha, R., McCool R.: Tap: Towards a web of data, http://tap.stanford.edu/
8. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters, Google, Inc. OSDI 2004, San Francisco, CA (2004)
9. Lucene-hadoop Wiki, HadoopMapReduce (2008), http://wiki.apache.org/lucene-hadoop/HadoopMapReduce
10. The Phoenix system for MapReduce programming (2008), http://csl.stanford.edu/~christos/sw/phoenix/.
11. Laclavik, M., Seleng, M., Hluchy, L.: ACoMA: Network Enterprise Interoperability and Collaboration using E-mail Communication. In: Expanding the Knowledge Economy: Issues, Applications, Case Studies. IOS Press, Amsterdam (2007)
12. Vojtek, P., Bieliková, M.: Comparing Natural Language Identification Methods based on Markov Processes. In: Slovko - International Seminar on Computer Treatment of Slavic and East European Languages, Bratislava (2007)
13. Krajči, S., Novotný, R.: Lemmatization of Slovak words by a tool Morphonary. In: TAOPIK (2), Vydavateľstvo STU, pp. 115–118 (2007) ISBN 978-80-227-2716-7
14. Corcho, O.: Ontology-based document annotation: trends and open research problems. International Journal of Metadata, Semantics and Ontologies 1(1), 47–57 (2006)
15. Open Source Distributed Computing: Yahoo's Hadoop Support, Developer Network blog (2007), http://developer.yahoo.net/blog/archives/2007/07/yahoo-hadoop.html
16. Yahoo! Launches World's Largest Hadoop Production Application, Yahoo! Developer Network (2008), http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html
17. Ontea: Pattern based Semantic Annotation Platform, SourceForge.net project (2008), http://ontea.sourceforge.net/
18. Snowball Project (2008), http://snowball.tartarus.org/
19. Apache Lucene project (2008), http://lucene.apache.org/
20. Klimt B., Yang Y.: Introducing the Enron Corpus. In: CEAS, 2004 (2008), http://www.ceas.cc/papers-2004/168.pdf, http://www.cs.cmu.edu/~enron/