Computing and Informatics, Vol. 28, 2009, 555–579, V 2009-Sep-16

ONTEA: PLATFORM FOR PATTERN BASED AUTOMATED SEMANTIC ANNOTATION

Michal Laclavík, Martin Šeleng, Marek Ciglan
Ladislav Hluchý

Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9
845 07 Bratislava, Slovakia
e-mail: michal.laclavik@savba.sk

Revised manuscript received 24 February 2009

Abstract. Automated annotation of web documents is a key challenge of the Semantic Web effort. Semantic metadata can be created manually or using automated annotation or tagging tools. Automated semantic annotation tools with best results are built on various machine learning algorithms which require training sets. Other approach is to use pattern based semantic annotation solutions built on natural language processing, information retrieval or information extraction methods. The paper presents Ontea platform for automated semantic annotation or semantic tagging. Implementation based on regular expression patterns is presented with evaluation of results. Extensible architecture for integrating pattern based approaches is presented. Most of existing semi-automatic annotation solutions can not prove it real usage on large scale data such as web or email communication, but semantic web can be exploited only when computer understandable metadata will reach critical mass. Thus we also present approach to large scale pattern based annotation.

Keywords: Semantics, semantic annotation, patterns, large scale processing

Mathematics Subject Classification 2000: 68T30, 68T10, 68Nxx

1 INTRODUCTION

Web, enterprise documents or email communication is understandable only for humans. Semantic web [1] and semantic annotation effort try to change this situation. Automated annotation tools can provide semantic metadata for semantic web as well as for knowledge management [2] or other enterprise applications [25]. Annotation can also provide semantic meta-data for other application such as e-Science [4], recommendation [38], personalization [39] or navigation [5]. Annotation solutions usually focus on the following main tasks:

- to provide annotation infrastructure and protocols
- to provide manual annotation or visualization tools
- to provide automated methods for annotation.

Annotation approaches can overlap these tasks since in every approach we need to:

- relate annotation or tags with original document
- provide customization or visualization tools for annotation process and its results
- support automation of annotation process.

Usually, whenever dealing with semantic annotation or tagging, we need to cover all of the aspects to some extent. Different strategy for annotation depends on the use of the annotation. There are a 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 [8] or to provide infrastructure and protocols for manual stamping documents with semantic tags such as Annotea[1], Rubby[2] or RDF annotation[3]. Even we always have to deal with infrastructure, protocols or some customization and visualization tools, in this paper we focus on automated annotation approaches for semantic metadata search and creation.

1.1 Automated Annotation State of the Art

Information Extraction – IE [9] is close to automated semantic annotation or tagging by Named Entity recognition – NE defined by series of Message Understanding Conferences (MUC). Semi-automatic annotation approaches can be divided into two groups with regard to produced results [9]:

- identification of concept instances[4] from the ontology in the text
- automatic population of ontologies with instances in the text.

[1] `http://www.w3.org/2001/Annotea/`
[2] `http://www.w3.org/TR/ruby/`
[3] `http://ilrt.org/discovery/2001/04/annotations/`
[4] In this paper instances and individuals are used with the same meaning

The Ontea method presented in the paper can provide results for both semantic annotation tasks. Semi-automatic solutions focus on creating semantic metadata for further computer processing, using semantic data in knowledge management [2] or in an enterprise application. Semi-automatic approaches are based on natural language processing [10] [11], adocument structure analysis [12] or learning requiring training sets or supervision [13]. Moreover, other annotation approaches exist, e.g. KIM[5] [21] which uses information extraction based on GATE[6] [28] information extraction system, GATE with Annie[7] extenssion or pattern-based semi-automatic solutions such as PANKOW and C-PANKOW [16] using QTag[8] patterns and Google API. To our best knowledge the only semantic annotation solution operating within distributed architecture and able to process large scale data is SemTag [14]. It uses the Seeker [14] information retrieval platform to support annotation tasks. SemTag annotates web pages using Stanford TAP ontology [15]. 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. For details survey on semantic annotation please see [13] [2] [20]. So far, automatic or semi-automatic approaches failed to deliver usable semantic data for semantic web applications. The main problem we see is to create general purpose semantic annotation tool usable through all domains of use – similarly as it is impossible to create one ontology which describes all domains. Thus we believe much *simple*, *scalable* and *customizable* approaches need to be used for semantic annotation. These are the gaps we are trying to address by the Ontea approach presented in the paper.

1.2 Article Structure

The structure of the article is divided into 5 chapters:

Introduction: Explains the state of the art and motivation for such work.

Ontea: Describes the Ontea aproach for semantic annotation on examples, explains the architecture, added value and integration with existing tools.

Evaluation: Discusses the success rate of Ontea.

Large Scale: Explains how Ontea can be used for large scale document annotation and evaluates the performance.

Conclusion: Summarizes results of the presented work.

2 ONTEA

Ontea identifies objects, their properties or their position in the text by applying patterns on a text. The input of the method is text and patterns and the output is

[5] `http://www.ontotext.com/kim/semanticannotation.html`
[6] `http://gate.ac.uk/`
[7] `http://gate.ac.uk/ie/annie.html`
[8] `http://www.english.bham.ac.uk/staff/omason/software/qtag.html`

key-value pairs (type – individual, individual – property), which can be transformed into RDF/OWL individuals. Ontea supports both semi-automatic annotation types:

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

The method used in Ontea [22, 23, 24] is comparable particularly with methods such as GATE, C-PANKOW, KIM, or SemTag. While these methods try to be general purpose annotation tools through the application domains, Ontea uses simpler approach and needs to be adapted on the domain by definition or customization of used patterns. Ontea works over text applicable to an application problem 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 or integration of other methods based on patterns such as wrappers, solutions using document structure, XPath, language patterns, e.g. C-PANKOW or GATE.

2.1 Added Value of Ontea Approach

Simplicity: Ontea can be quite easily used since it is built on regular expression patterns (regexes). Even if not each developer is able to write regexes, regexes are widely used in text processing field. Furthermore, Ontea uses key-value pair approach to encode the result of annotation, where a *key* corresponds with a class/concept/type or a property name and a *value* corresponds with an instance/individual or a property value. More details can be found in Section 2.2.

Usability for any language: While most of advanced information extraction or annotation techniques use natural language processing (NLP) methods such as sentence decomposition or part of speech tagging (POS), such techniques do not exists or libraries (such as QTag) are not available for most of European languages, while regular expression patterns can be applicable for any language. However, NLP methods can be integrated via key-value pair result transformation.

Extensible and modular architecture: allows integration with existing techniques and allows to use simple building blocks based on key-value pairs and its transformation to prepare more advanced semantic meta data instances with properties using inference techniques; see Section 2.3.

External Data Integration: Ontea allows key-value pair transformation using external data sources to enrich annotation with extra values, properties not directly extracted from text. Such extra data can be used in ontology instance creation or search. For more details see Section 2.6 on System Connectors.

Scalability: Unlike the other annotation techniques the Ontea approach is scalable to process large data sets such as web documents, enterprise data repositories or email communication archives. See Section 4 for more details.

Suitability for "Semantic Enterprise": Information overload and problems encountered with managing available digital information is present in most of nowadays enterprises. Semantic annotation can help manage [2] information and knowledge within an enterprise. Solution such as Ontea built on reusable and configurable patterns can support contextualization and better management of information, which was partially proved and tested on email communication processing and email based recommendation system Acoma [3, 25]. Ontea implementation also allows to define patterns for objects boundaries and then group and create RDF/OWL individuals with assigned literal properties. Ontea is also able to assign discovered or created object properties to individual which represents processed text or its part. In addition, Ontea is able to enrich annotations using data from other sources such as databases, intranet systems or spreadsheets connected through system connector transformers.

2.2 Example of Use

The Ontea tool can be used in three different scenarios depending on application needs:

Ontea: searching for relevant instances in local knowledge base[9] on the basis of common patterns

Ontea creation: creation of new instances of the objects found in text

Ontea creation IR: as in the preceding example but with the use of information retrieval techniques, e.g. Lucene[10] or RFTS[11], used to identify a relevance value of created instance (see Section 2.5 for more details).

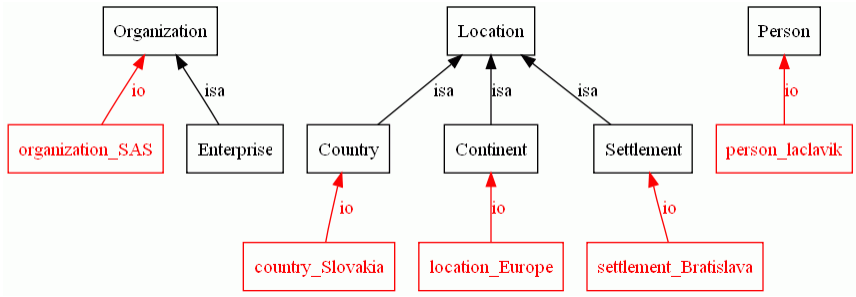


Fig. 1. Simple ontology used in examples

[9] By *knowledge base* we mean ontology repository, e.g. Sesame or Jena RDF/OWL models within this paper

[10] `http://lucene.apache.org/java/docs/index.html`

[11] `http://nazou.fiit.stuba.sk/home/?page=dairfts`

The following texts, two in English and one in Slovak, in Table 1 and ontology with several instances (Figure 1, instances are connected to their concepts via "io" arrows) illustrate the Ontea method. Moreover, the table shows examples of used patterns based on regular expressions. For first English text only one regular expression was used to find one or two words beginning with a capital letter.

#	Text	Patterns – regular expressions
1	Michal Laclavik works for Slovak Academy of Sciences located in Bratislava, the capital of Slovakia	\b([A-Z][a-z]+ +[A-Z][a-z]+|[A-Z][a-z]+)\b
2	Automobilka KIA sa rozhodla investovať pri Žiline, kde vybudovala svoju prvú továreň vEurópe. Kontakt: Kia Motors Slovakia, s.r.o. P. O. Box 2 013 01 Tepličkа nad Váhom Slovakia	*Organization* \b([\p{Lu}][-&\p{L}]+[ ]*[-&\p{L}]*[ ]*[-&\p{L}]*)[,]+s\.r\.o\. *Settlement* \b[0-9]{3}[ ]*[0-9]{2} +(\p{Lu}\p{L}+[ ]*\p{L}*[ ]*\p{L}*)[ ]*[0-9\n,]+ *Location* (v|pri) +(\p{Lu}\p{L}+)
3	Car maker KIA Motors decided to build new plant in Slovakia near the town of Zilina. It is its first plant in Europe. Contact: Kia Motors Slovakia, Ltd. P. O. Box 2 013 01 Teplicka nad Vahom Slovakia	*Location* (in|near) +([A-Z][a-z]+) *Settlement* (city|town) of ([A-Z][a-z]+ *[A-Z][a-z]+ *) *Organization* \b([A-Z][a-z]+[ ]*[A-Za-z]*[ ]*[A-Za-z]*)[,]+(Inc|Ltd)[.\s]+

Table 1. Example text

In the example #1 we have showed only finding of instances present in a knowledge base (see Figure 1 and Table 2, first row). The Slovak text example #2 and the same text in English (example #3) demonstrated not only searching for instances within a knowledge base but also their creation. The examples #2 and #3 use three patterns[12]:

- Searching for a company by *s. r. o.*, *Inc.* or *Ltd.*
- Searching for residence/domicile by ZIP code
- Searching for geographical location by means of prepositions in, near followed by a word beginning with a capital letter.

Table 2 gives also the results of examples #2 and #3. Instances created from the second row are also shown in Figure 2.

To produce results in third row of Table 2 we used lemmatization of found texts; this enables more precise creation of instances in the nominative in cases *Žilina* and

[12] Some of the patterns were simplified in Table 1 to become more readable.

Example #	Method	Annotation Results
1	Ontea	*Person* Michal Laclavik *Organization* Slovak Academy *Settlement* Bratislava *Country* Slovakia
2	Ontea create	*Location* Žiline *Location* Európe *Organization* Kia Motors Slovakia *Settlement* Teplička nad Váhom
2	Ontea create, lemmatization	*Location* Žilina *Continent* Európa *Organization* Kia Motors Slovakia *Settlement* Teplička nad Váh
3	Ontea create	*Country* Slovakia *Settlement* Zilina *Continent* Europe *Organization* Kia Motors Slovakia

Table 2. Annotation result

Európa, however, location *Teplička nad Váhom* came up wrong, as *Teplička nad Váh*. It would be appropriate to use lemmatization for several patterns only, not for all of them. Furthermore, we can direct our attention to *Európa* that is not of a *Location* but *Continent* type because the algorithm found it in a knowledge base using inference since *Continent* is a subclass of *Location*. In the creation process, the algorithm first looks into the knowledge base to find out whether instance of such type or inferred instance already exists.

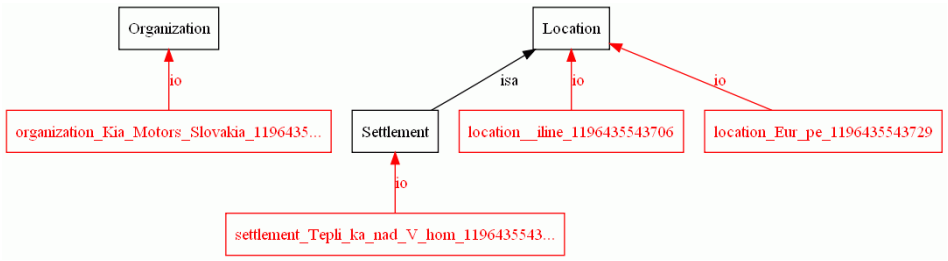


Fig. 2. Created instances in Slovak text – Ontea create

2.3 Ontea Architecture

Architecture is built to allow customizable and extensible transformation chain of key-value pairs extracted from text. 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. The currently implemented pattern search uses regular expressions *PatternRegExp* and *PatternXPath* to extract information from, or annotate and tag text, XML or HTML files. This interface can be used to implement more advanced patterns as well as to be used for integration of existing semantic annotation techniques.Curently we also are preparing integration with GATE.

onetea.core.Result: a class representing annotation results by means of individual of defined type/concept. Examples can be seen in Table 2. Its extensions are different types of individuals depending on implementation in ontology (Jena, Sesame) or as value and type (key-value) pairs.

ontea.transform.ResultTransformer: the interface, which after implementation, contains different types of transformations among annotation results. Thus it can transform set of results and include in transformation various scenarios of annotation such as relevance, result lemmatization, transformation of found key-value pairs (Table 2) into OWL individual in Sesame[13] or Jena[14] API implementation. It is used to transform key-value pairs into different key-value pairs represented e.g. by URI or lemmatized text value. It can be also used to eliminate irrelevant annotation results using relevance identification described in Section 2.5.

In Figure 3 you can see *Result* class, *Pattern* and*ResultTransformer* interfaces. Such design allows extending Ontea for different pattern implementations or for the integrations of existing pattern annotation solutions. It is also possible to implement various result transformations by implementing *ResultTransformer* interface, which can be used also as inputs and outputs between Map tasks in MapReduce architecture, when using Ontea algorithm on large scale data. This is discussed in Section 4.

2.4 Integration of Ontea with External Tools

The 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 described in Section 4. Integration with Hadoop requires implementation of Map and Reduce methods.

Language Identification: In order to use correct regular expressions or other patterns, we often need to identify the language of use. For this reason it is convenient to integrate Ontea with the language detection tool. We have tested Ontea with Nalit [29]. Nalit is able to identify Slovak and English texts as well as other ones, if trained.

[13] `http://openrdf.org/`
[14] `http://jena.sf.net/`

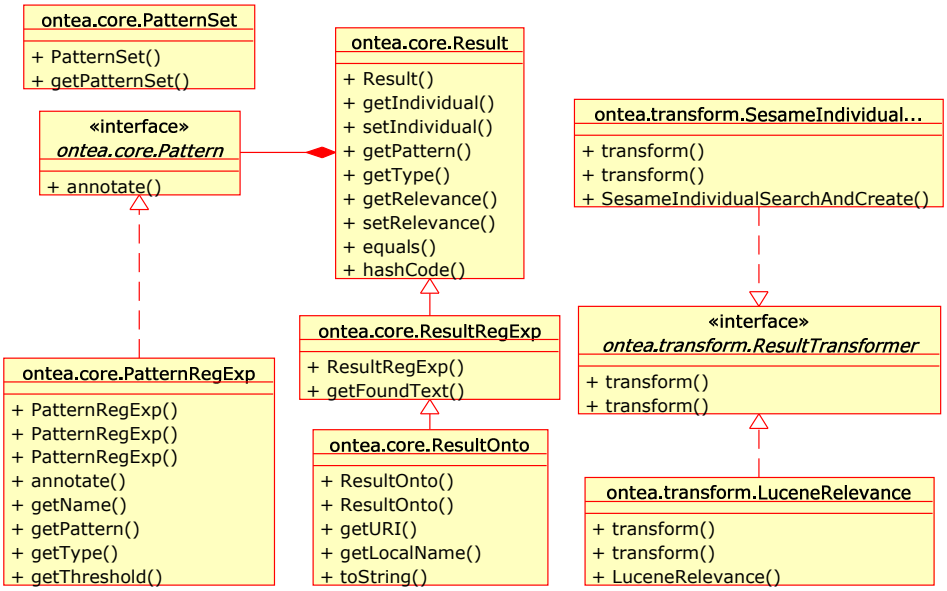


Fig. 3. Basic classes of Ontea platform

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

Lemmatization: When concrete text is extracted as representation of an individual, we often need to lemmatize found text to find or create correct instance. For example, the capital of Slovakia can be identified in different morphological forms: *Bratislava*, *Bratislave*, *Bratislavu*, or *Bratislavou* and by lemmatization we can identify it always as individual Bratislava. We have tested Ontea with Slovak lemmatizer Morphonary [30]. It is also possible to use lemmatizers or stemmers from Snowball project [31], 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. Our approach to relevance identification is described in Section 2.5.

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 local knowledge base if existing and to create new ones if no instance is found in knowledge base. Ontea also uses inference to found appropriate instance. For example if Ontea processes the sentence *Slovakia is in Europe* using pattern for location detection *(in|near) +(\p{Lu}\p{L}+)* the following key-value pair is detected: *Location – Europe*. If we have Location ontology with subclasses as Continents, Settlements, Coun-

tries or Cities and Europe is already present as instance of continent (see Figure 1), Ontea can detect existing *Europe* instance in the knowledge base using inference.

System Connectors Integration: Ontea allows key-value pair transformation using external data sources to enrich annotation with extra values, properties not directly extracted from text. Such extra data can be used in ontology instance creation or search. For more details see Section 2.6 on System Connectors.

2.5 Relevance Evaluation of Instances

Ontea's method of automatic annotation based on regular expressions matching showed promising results for domain specific texts. However, it suffers from frequent mismatching which leads to creation of imprecise instances of ontological concepts. We propose to overcome this obstacle by evaluating the relevance of candidate instances by the means of statistical analysis of the occurrence of the matched words in the document collection. Based on regular expression or other pattern type, Ontea identifies part of a text related to semantic context and matches 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:

$$\frac{close\_occurrence\,(C, I)}{occurrence\,(I)}$$

We used a prototype indexing tool – RFTS[15] that provides us with the functionality to retrieve required statistical values, computed from the whole collection of documents.

Let COLL be a collection of the documents $d_1, d_2, \ldots, d_n$: $COLL = d_1, d_2, \ldots, d_n$ Let d in COLL,$distance \in N$, and $w_1, w_2, \ldots, w_k$ are the words from natural language. Function $dist\,(d, distance, w_1, w_2, \ldots, w_k)$, where $k \leq distance$, denotes the number of distinct word sequences of the length distance containing the words $w_1, w_2, \ldots, w_k$. We compute the relevance of candidate instance as:

$$relevance\,(C, I, wordsdit) = \frac{\sum dist\,(d, wordsdist, C \cup I)}{\sum dist\,(d, (I), I)}$$

If the resulting relevance value exceeds defined threshold, the candidate word sequence I is considered to be a valid instance of the semantic concept related to sequence C. For the experimental evaluation of the approach, the threshold was set manually after inspecting the preliminary relevance values of the generated candidate instances.

[15] `http://nazou.fiit.stuba.sk/home/?page=dairfts`

Simplified relevance can also be computed using RFTS or Lucene. Percentage of occurrence of matched regular expression pattern to detected element represented by word on used document set is computed in this case as the relevance value. An example: *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. The idea is to omit annotation of objects which can have double meaning of the same text, e.g. *Microsoft* can be found in *Microsoft Inc.* and *Microsoft Office*. Such approach can increase the precision of annotation when needed, of course with decrease of recall. Use of RFTS or Lucene is related to *Ontea IR* scenario and *LuceneRFTS* or *LuceneRelevance* implementation of *ResultTransformer* interface. Similarly, other relevance algorithms such as cosine measure (used for example in SemTag [14]) can be implemented. Both approaches can be valuable and used with success depending on application, but gives the same results for chosen patterns within evaluation presented in Section 3.

2.6 Enriching Annotation with Information from External Systems

System connectors (SC) are middleware components providing access to legacy enterprise systems and public information systems. The purpose is to retrieve additional parameters for the input parameters specified by the software components exploiting SC. Connectors were designed to allow enriching the outputs from annotation tools and information retrieval with the data from external systems. For example, if a text annotation tool identifies the name of a person in input text (e.g. e-mail message/purchase order), the system connectors can be used to access company's partner database to retrieve the name and business ID of the company the identified person is employed in; another system connector can then be used to retrieve additional information on company from public business registry, or query billing system to retrieve last n transactions with the given company. The additional information is then processed by the annotation tool and presented to the user.

Each SC is specified by the type of resource it provides connection to (e.g. relational database, spreadsheet data resource, web application), the identification of concrete resource and metadata describing its input and output. Input metadata is a tuple (an ordered list) of parameters (name and data type). SC accepts as input tuples matching input metadata or set of key-value pairs that can be matched to input metadata. SC output is a set of tuples; the output tuples are also described by metadata (list parameter names and data types).

The metadata description of SC's input and output were introduced to allow construction of meta-connectors. Meta-connector, given a set of system connectors, is designed to match an input set of key-value pairs to the input tuple of one or several subordinate systems connectors and aggregate their outputs. The meta-connectors facilitate the integration with annotation or information retrieval systems – e.g. annotation tool configured to identify set of concepts in texts (such as e-mails, person names, company names, addresses) can identify all or only a subset of those concepts. When using meta-connector, the tool only passes the identified

values to meta-connector that automatically chooses appropriate connectors and enriches the information from available resources.

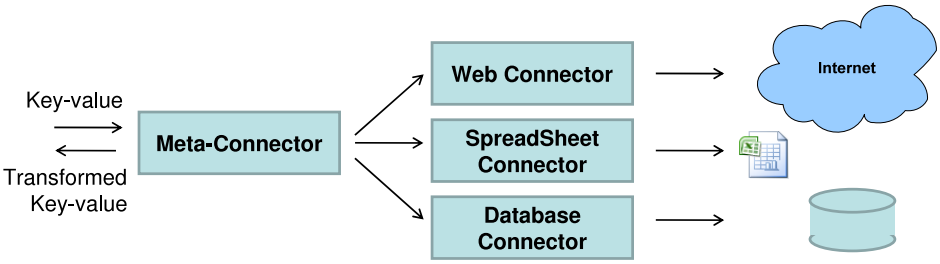


Fig. 4. System Conectors Connected to Meta-connector

To conclude, Ontea uses meta-connector to transform key-value pairs to identify instance or to gather instance properties from external systems as seen in Figure 4. For example key-value pair from Table 2 *Organization – Kia Motors Slovakia* can be transformed to *OrganizationRegistryNumber – 35876832* using web connector to Slovak Enterprise Register[16]. Instance properties can then be used when creating new ontology instance in knowledge base.

3 EVALUATION OF ONTEA ALGORITHM SUCCESS RATE

In this section we describe evaluation of Ontea annotation success rate. We also describe a test set of documents. Performance evaluation is presented in Section 4, where we also provide results on porting Ontea on MapReduce distributed architecture.

3.1 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 job offer application ontology[17], 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.

[16] `http://www.orsr.sk/vypis.asp?ID=11731&SID=5&P=`
[17] `http://nazou.fiit.stuba.sk/home/?page=ontologies`

3.2 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 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 the knowledge base. This second case is referred to as *Ontea creation*. The third case used also the 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 local 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 the previous case with the feedback of RFTS or Lucene to get relevance computed using word occurrence. Individuals were created only when relevance was above the defined threshold which was set up to 10 % (see Section 2.5 for more details)

These cases were described also in Section 2.2 *Example of Use* and evaluation results for these cases are shown in Section 3.4.

We used the following regular expressions:

- Generic expression matching one or more words in text. This was used only to search concepts in the KB
  ([A-Z][-A-Za-z0-9]+[\s]+ [-a-zA-Z]+)
- Identifying geographical location in text and if not found in the KB, an individual was created
  Location: *[\s]*([A-Z][-a-zA-Z]+[ ]*[A-Za-z0-9]*)*
- Identifying a company in the text, this was used also with other abbreviations such as *Inc.* or *Ltd.*
  [\s]+([-A-Za-z0-9][ ][A-Za-z0-9]*),[ ]*Inc[.\s]+*
  or in the job offers often referenced as *Company: Company name*
  [Cc]ompany[:\s]([A-Z][-A-Za-z0-9][ ]*[A-Za-z0-9]*.*

3.3 Success Rate of the Ontea Algorithm

In this section we discuss the algorithm evaluation and success rate. To evaluate the success of annotation, we have 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 evaluate techniques in Information Retrieval (IR) field by Cleverdon [17]. To obtain a better measure to describe performance, we use the F_1 measure (first introduced by van Rijsbergen [18]) 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.

3.4 Experimental Results of Annotation Success Rate

Ontea experimental results using precision, recall and F_1-measures are in the Table 3. In the Table 4 we show results of other semantic annotation approaches and we also list some advantages and disadvantages. The information concerning relevant annotation techniques are based on experiments presented in survey papers [13, 2], SemTag [14] and C-PANKOW [16] approaches. The success rate numbers for different tools were not evaluated on the same documents data sets and also some numbers are not available. Thus the table rows for other then Ontea tool should be taken as extra information about the method but can not be taken as direct measures to compare Ontea success rate with other methods. The column *Relevance* is similar to F_1-measures culumn in Ontea related Table 3 but in case of other methods it was evaluated by other techniques. For example for C-PANKOW, relevance is referred to as recall.

Method	Description	F_1 %	P %	R %	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
Ontea creation	regular expresions (RE), creation of individuals in KB	41	28	81	aplication specific patterns are needed low precision	support for any 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 identified and not annotated, good results

Table 3. Ontea success rate evaluation (P column stands for precision, R for recall)

Method	Description	relevance %	P %	R %	Disadvantages	Advantages
SemTag	disambiguatiy check, searching in KB	high	high		works only for TAP KB	fast and generic solution
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 4. Annotation tools success rate summary (P column is precision, R is recall)

In Table 3 you can see results for both semantic annotation use cases:

- identification of concept instances from the ontology in the text: row *Ontea*
- automatic population of ontologies with instances in the text: rows *Ontea creation* and *Ontea creation IR*.

The row *Ontea creation IR* case is the most important considering evaluation where we combined information retrieval 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 the use of IR. Similarly *Ontea creation* identified also companies as *Microsoft* or *Oracle* which is correct and eliminated in combination with IR. This issue decreases recall and increases precision. Here it seems that IR case is not successful but the 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 high precision is required.

3.5 Success Rate Evaluation Summary

Evaluation of Ontea algorithm showed quite satisfactory results especially when identifying instances in local knowledge base – *Ontea* case. Recall was higher than 80 % and precision higher than 60 %. In *Ontea creation* case, we have achieved quite high recall (over 80 %) but precision was very low (28 %) due to the fact that patterns discovered often the same text with different meaning (similarly as in the example with *Microsoft* or *Microsoft Office* mentioned earlier). Because of such problems we have introduced IR supported instance creation where relevance whether to create instance is computed – *Ontea creation IR* case. In this case we achieved precision 53 % and recall 79 %. Such results are quite satisfactory since recall decreased only by 2 %. Ontea success rate depends much on definition of regular expression patterns and on the document collection to be annotated. We can achieve very high recall with simple patterns, but precision will be poor. When regular expression patterns will be defined carefully, precision can increase dramatically. One can argue that this method depends too much on how well the patterns are defined. This is absolutely true, but on the other hand such solution is very powerful in enterprise environment where business specific patterns need to be defined to identify products, services, invoice codes, customers or other business objects. Patterns can also be shared within a community when defined and tested for common types of objects such as people names, companies, geographical locations, addresses or contact details. In this case, one can use them directly and know expected success rates. To conclude, the state of the art semantic annotation tools declare to achieve similar success rate as Ontea. Ontea also supports both annotation tasks – population of ontologies by instance creation as well as identification of instances. However, main gaps we address by the Ontea method are *simplicity*, *customizability* and *scalability*; this would have no impact and use if algorithm success rate would be low.

4 LARGE SCALE ANNOTATION AND PERFORMANCE EVALUATION

According to our best knowledge none of the state of the art semi-automatic annotation tools except SemTag [14] provides performance evaluation results. We believe that Ontea method is one of the fastest, due to its relative simplicity – regular expression approach. For example, C-PANKOW [16] uses more advanced patterns via QTag, and also Google API for relevance calculation and thus seems to be slower. However, annotating large number of documents or periodical re-annotating of updated document collection is still time consuming when performing the computation on a single server. This is why, we have performed several experiments on parallel architectures.

We have successfully ported semantic annotation into Grid [23] with good results, but porting of application, data management and results integration was difficult and time consuming task. Thus we have also focused on different parallel and distributed architectures. In this section we discuss porting of Ontea into MapReduce architecture and its Hadoop implementation. We provide performance results on 8 nodes Hadoop cluster on email data repository.

4.1 MapReduce Architecture

MapReduce [19] architecture developed by Google was used with success on information retrieval tasks. Information extraction and pattern based annotation use similar methods such as information retrieval. This is another reason 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 various wrappers can be ported into MapReduce architecture as well, following similar approach as described in this section. Google's MapReduce [19] 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 information retrieval and information extraction tasks can be ported into MapReduce architecture, similar to pattern based annotation algorithms. E.g. distributed grep[18] using regular expressions, one of basic examples for MapReduce, is similar to Ontea pattern approach using regular expressions as well.
- Input and output of Map and Reduce functions are key-value pairs. Porting of Ontea as well as using Ontea transformers is thus straightforward.

In Figure 5 the main components of the MapReduce architecture are shown: Map and Reduce methods, data in distributed file system (DFS), inputs and outputs. Several data replicas 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 what a developer needs to do is to implement the Map and Reduce methods. The architecture will take care of distribution, execution of tasks as well as of fault tolerance. For more details on MapReduce see [19]. Two open source implementations of MapReduce are available:

- Hadoop [32], developed as Apache project with relation to Lucene and Nuch information retrieval systems, implemented in Java. Hadoop is well tested on

[18] Grep is a flexible search-and-replace function that can search one or more files for specified characters and/or strings

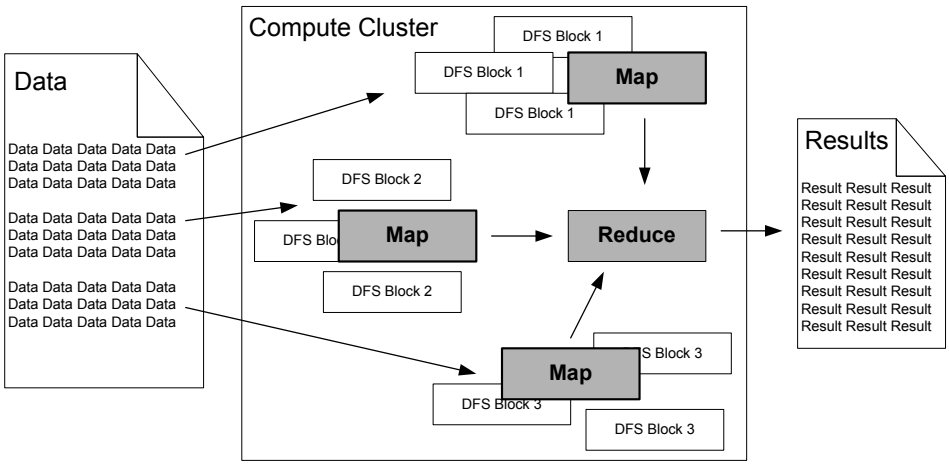


Fig. 5. MapReduce Architecture figure (source: Hadoop website)

many nodes. Yahoo! is currently running Hadoop on 10 000 nodes [33] in production environment [34].

- Phoenix [35], developed at Stanford University, implemented in C++.

4.2 Ontea Porting to Hadoop MapReduce

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 – input and outputs of Map and Reduce methods to be implemented in MapReduce architecture. Possible Ontea annotation results detected in test data were instances such as settlements, company names, persons or email addresses, similar as the examples shown in Table 2. In the Map method, input is a text line which is processed by Onteas regex patterns and outputs are key value pairs:

Key: string starting with detected instance type and continuing with instance value similar to Table 2 annotation results. This can be extended to return also instance properties, e.g. address, phone or email as company properties.

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.

The above key-value description is valid if we need to relate annotations with documents. In many cases key-value pairs in Ontea extraction and transformation are identical with key-value pairs in Ontea-Hadoop implementation, for example when extracting social network of communicating people from email communication archives [37].

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

ontea.core.Pattern: interface for adaptation of pattern based searching in text. The main Pattern method *Pattern.annotate()* runs inside the Map method in MapReduce implementation.

ontea.core.Result: a class which represents the result of annotation – an ontology individual. It is based on the type and value pairs similar to pairs in Table 2, *annotation results* 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, or directly as key-value pairs.

ontea.transform.ResultTransformer: interface which transforms annotation results. Transformers are used in Map or Reduce methods in MapReduce implementation to transform individuals into OWL file or to eliminate some results using *Ontea IR* scenario. Transformers can be used as different Map tasks to be executed in a row depending on application scenario. For example when extracting social network from email, email addresses can be transformed into persons, organizations or else using ontology transformers above proper ontology model and knowledge base [37].

4.3 Ontea Running on Hadoop MapReduce Cluster

We wrapped up Ontea functionality into Hadoop MapReduce library. We tested it on Enron email corpus [36] containing 88 MB of data and our personal email containing 770 MB of data. We run the same annotation patterns on both email data sets, on single machine as well as on 8 node Hadoop cluster. We have used *Intel(R) Core(TM) 2 CPU* 2.40 *GHz with 2 GB RAM* hardware on all machines.

As you can see from Table 5, the performance increased 12 times on 8 nodes (16 cores) in case of large data set. In case of smaller data set it was only twice faster than on single machine and MapReduce overhead is much more visible. In Table 5 we present only 2 concrete runs on 2 different datasets, but we have executed several runs on these datasets and computational time was very similar so we can conclude that the times presented in Table 3 are very close to average. In the above 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 the potential of *ResultTransformers* in Ontea architecture. Beside the above experiments we have performed also extraction of social network and its transformation via semantic model using Ontea and Hadoop. These experiments are described in [37]. We believe we have proved scalability of Ontea by porting it on Hadoop.

Description	Enron corpus (88 MB)	Personal email (770 MB)
Time on single machine	2 min, 5 sec	3 hours, 37 mins, 4 sec
Time on 8 nodes hadoop cluster	1 min, 6 sec	18 mins, 4 sec
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

Table 5. Performance and execution results

5 CONCLUSION AND FUTURE WORK

In this paper we describe briefly the state of the art in automated semantic annotation. We focus on pattern based annotation which we believe is valuable especially in enterprise applications, but can be applied also in specialized applications on the web or e-Science. We described Ontea annotation platform, which uses simple regex approach to extract key-value pairs from text and searches or creates ontology instances by various transformations of key-value pairs into ontology instances. We have described success rate of Ontea and also its modular and extensible architecture, where other pattern based annotation solution can be integrated and methods can be easily customized or extended. We provided also examples of Ontea integration with variety of tools from information retrieval or semantic web. Ontea solution was also tested and used (see demo[19]) on Job Offer application in the Nazou[20] project [26, 5]. It was also tested in enterprise environment on email communication data [3, 25]. In this paper we have also discussed how automatic semantic annotation solution such as Ontea can benefit from MapReduce distributed 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 performance results on single machine and Hadoop cluster and thus we have proved scalability of the solution.

In our future work we would like to prove Ontea usability and scalability on concrete application domains such as geographical location identification of web pages and large scale email processing to improve automated email management and se-

[19] `http://nazou.fiit.stuba.sk/home/video/ontea/nazou-ontea.htm`

[20] `http://nazou.fiit.stuba.sk/home/?page=about`

mantic searching. We would like to improve Ontea approach for instance searching by implementing gazetteer like approach used for example in GATE information extraction system. We also would like to continue developing Ontea as open source within the `Ontea.sourceforrge.net` [27] project addressing mainly *simplicity*, *customizability* and *scalability* of automated semantic annotation.

Acknowledgment

This work was partially supported by projects NAZOU SPVV 1025/2004, VEGA No. 2/6103/6, Commius FP7-213876, AIIA APVV-0216-07 and SEMCO-WS APVV-0391-06.

REFERENCES

[1] Berners-Lee, T.—Hendler, J.—Lassila, O. et al.: The Semantic Web. Scientific American, Vol. 284, 2001, No. 5, pp. 28–37.

[2] 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, Vol. 4, 2005, No. 1, pp. 14–28.

[3] Laclavík, M.—Šeleng, M.—Hluchý, L.: Network Enterprise Interoperability and Collaboration using E-Mail Communication. Expanding the Knowledge Economy: Issues, Applications, Case Studies, IOS Press, Amsterdam, 2007, ISBN 978-1-58603-801-4.

[4] Chen, H.—Ma, J.—Wang, Y.—Wu, Z.: A Survey on Semantic E-Science Applications. In Computing and Informatics, Vol. 27, 2008, No. 1, pp. 1–156.

[5] Bieliková, M.—Matušíková, K.: Social Navigation for Semantic Web Applications Using Space Maps. In Computing and Informatics, Vol. 26, 2007, No. 3, pp. 281–299.

[6] Handschuh, S.—Staab, S.: Authoring and Annotation of Web Pages in Cream. In WWW '02, pp. 462–473, NY, USA, 2002. ACM Press. ISBN 1-58113-449-5, `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, pp. 191–197, New York, NY, USA, ISBN 1-58113-815-6, ACM Press, 2004.

[8] Uren, V. et al.: Browsing for Information by Highlighting Automatically Generated Annotations: A User Study and Evaluation. In K-CAP '05, pp. 75–82, NY, USA, ACM Press, 2005, ISBN 1-59593-163-5.

[9] Cunningham, H.: Information Extraction, Automatic. In: Keith Brown (Editor-in-Chief): Encyclopedia of Language & Linguistics, Second Edition, Volume 5, pp. 665–677. Elsevier, Oxford, 2006.

[10] Madche, A.—Staab, S.: Ontology Learning for the Semantic Web. IEEE Intelligent Syst., Vol. 16, 2001, No. 2, pp. 72–79.

[11] Charniak, E.—Berland, M.: Finding Parts in Very Large Corpora. In Proceedings of the 37th Annual Meeting of the ACL, 1999, pp. 57–64.

[12] 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, pp. 562–569, ACM Press, 2002.

[13] Reeve, L.—Hyoil Han: Survey of Semantic Annotation Platforms. In SAC '05, pp. 1634–1638, ACM Press, NY, USA, 2005, ISBN 1-58113-964-0.

[14] Dill, S.—Eiron, N. et al.: A Case for Automated Large-Scale Semantic Annotation. Journal of Web Semantics, 2003.

[15] Guha, R.—McCool, R.: Tap: Towards a Web of Data. http://tap.stanford.edu/.

[16] 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, 2005, ISBN 1-59593-046-9, pp. 332–341.

[17] Cleverdon, C. W.—Mills, J.—Keen, E. M.: Factors Determining the Performance of Indexing Systems. Vol. 1–2. Cranfield, U.K., College of Aeronautics.

[18] van Rijsbergen, C. J.: Information Retrieval. Butterworth-Heinemann, Newton, MA, 1979.

[19] Dean, J.—Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Google, Inc., OSDI '04, San Francisco, CA, December 2004.

[20] Corcho, O.: Ontology-Based Document Annotation: Trends and Open Research Problems. International Journal of Metadata, Semantics and Ontologies Vol. 1, 2006, No. 1, pp. 47–57.

[21] Kiryakov, A.—Popov, B.—Terziev, I.—Manov, D.—Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. Elseviers Journal of Web Semantics, Vol. 2, 2005, No. 1, http://www.ontotext.com/kim/semanticannotation.html.

[22] Laclavík, M.—Šeleng, M.—Gatial, E.—Balogh, Z.—Hluchý L.: Ontology Based Text Annotation OnTeA. Information Modelling and Knowledge Bases XVIII, IOS Press, Amsterdam, 2007, 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, pp. 311–315.

[23] Laclavík, M.—Ciglan, M.—šeleng, M.—Hluchý, L.: Ontea: Empowering Automatic Semantic Annotation in Grid. In Paralell Processing and Applied Mathematics: 7th International Conference, PPAM 2007, R. Wyrzykowski, Jack Dongarra, Konrad Karczewski, Wasniewski (Eds.), Springer, Berlin, 2008, ISBN 978-3-540-68105-2, ISSN 0302-9743, pp. 302–311.

[24] Laclavík, M.— šeleng, M.—Hluchý, L.: Towards Large Scale Semantic Annotation Built on Mapreduce Architecture. In Computational Science, ICCS 2008, 8th International Conference, Marian Bubak, Geert Dick van Albada, Jack Dongarra, Peter M. A. Sloot (Eds.), Springer, Berlin, 2008, ISSN 0302-9743, Lecture Notes in Computer Science, Vol. 5102, Part III, pp. 331–338.

[25] Laclavík, M.—Ciglan, M.—šeleng, M.—Gatial, E.—Hluchý, L.: Future Email Services and Applications. In CEUR-WS, Proceedings of the poster and demon-

stration paper track of the 1st Future Internet Symposium (FIS '08). Vienna, Telecon Res. Center Vienna 2008, ISSN 1613-0073, Vol. 399, pp. 33–35.

[26] Návrat, P.—Bieliková, M.—Rozijanová, V.: Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: Proc. International Conference on Computer Systems and Technologies, CompSysTech 2005, Varna, 2005, http://ecet.ecs.ru.acad.bg/cst05/Docs/cp/SIII/IIIB.7.pdf.

[27] Ontea: Pattern based Semantic Annotation Platform. SourceForge.net project, http://ontea.sourceforge.net/, 2008.

[28] Cunningham, H.—Maynard, D.—Bontcheva, K.—Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL '02), Philadelphia, 2002.

[29] 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.

[30] Krajči, S.—Novotný, R.: Lemmatization of Slovak Words by a Tool Morphonary. In TAOPIK (2), Vydavateľstvo STU, 2007, ISBN 978-80-227-2716-7, pp. 115–118.

[31] Snowball Project, http://snowball.tartarus.org/, 2008.

[32] Lucene-hadoop Wiki, HadoopMapReduce, 2008, http://wiki.apache.org/lucene-hadoop/HadoopMapReduce.

[33] Open Source Distributed Computing: Yahoo's Hadoop Support, Developer Network blog, http://developer.yahoo.net/blog/archives/2007/07/yahoo-hadoop.html, 2007.

[34] Yahoo! Launches World's Largest Hadoop Production Application. Yahoo! Developer Network, http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html, 2008.

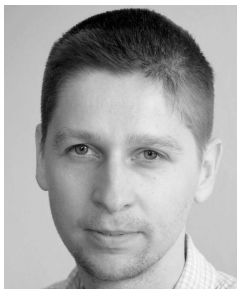
[35] The Phoenix system for MapReduce programming. http://csl.stanford.edu/christos/sw/phoenix/, 2008.

[36] Klimt, B.—Yang, Y.: Introducing the Enron Corpus. CEAS 2004, http://www.ceas.cc/papers-2004/168.pdf, http://www.cs.cmu.edu/∼enron/, 2008.

[37] Laclavík, M.—Šeleng, M.—Hluchý, L.: Supporting Collaboration by Large Scale Email Analysis. CGW 2008.

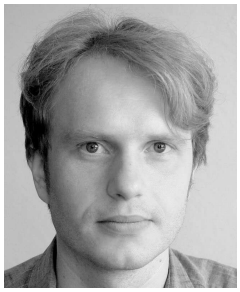
[38] Andrejko, A.—Bieliková, M.: Comparing Instances of Ontological Concepts for Personalized Recommendation in Large Information Spaces. In this issue.

[39] Barla, M.—Tvarožek, M.—Bieliková, M.: Rule-based User Characteristics Acquisition from Logs with Semantics for Personalized Web-Based Systems. In this issue.

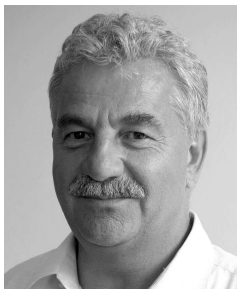


Michal Laclavík is a researcher at Institute of Informatics, Slovak Academy of Sciences. In 1999 he received his M. Sc. degree in computer science and physics. He received his Ph. D. degree in applied informatics with focus on knowledge oriented technologies in 2006. He is the author and co-author of several scientific papers, and participates in the Pellucid, K-Wf Grid and Commius European projects and in several national projects. He has strong scientific and development expertise in email based systems and semantic annotation. He also gives lectures on information retrieval at Slovak University of Technology. Finally, he is one of the initiators of the Commius project focussing on email based interoperability for SMEs. His research interests include email analysis and processing, information management and processing, and semantic annotation.

Martin Šeleng is a researcher at Institute of Informatics, Slovak Academy of Sciences. In 1999 he obtained his MSc degree in mathematics and computer science at the Faculty of Mathematics and Physics. He worked previously at Faculty of Economy and Informatics at the Economic University as a researcher and a teacher in the field of mathematics, statistics and computer science. He has strong expertise with email related system development and information system evaluation. He has been employed at the institute since 2006. He is the author and co-author of several scientific papers and participates in the K-Wf Grid and Commius European projects and in several national projects. He teaches information retrieval at the Faculty of Informatics and Information Technologies. His research interests include email communication and large scale information processing.



Marek Ciglan is a research assistant at the Institute of Informatics of Slovak Academy of Sciences. He received his M. Sc. degree in computer science in 2003. In 2008, he received his Ph. D. degree in applied informatics; with thesis focused on data replication in distributed environments. He has authored and co-authored multiple research papers and participated in several European and national research projects. His research interest is in distributed data management and distributed heterogeneous systems.



Ladislav Hluchý is the Director of the Institute of Informatics of the Slovak Academy of Sciences and also the Head of the Department of Parallel and Distributed Computing at the Institute. He received his MSc and PhD degrees, both in computer science. He is R & D Project Manager, Work-package Leader in a number of 4 FP, 5 FP, 6 FP and 7 FP projects, as well as in Slovak R & D projects (VEGA, APVT, SPVV). He is a member of IEEE, ERCIM, SRCIM, and EuroMicro consortiums, the Editor-in-Chief of the journal Computing and Informatics. He is also (co-)author of scientific books and numerous scientific papers, contributions and invited lectures at international scientific conferences and workshops. He is also a supervisor and consultant for Ph. D., master and bachelor studies.