Methods and Tools for Ontology Building, Learning and Integration – Application in the SYNAT Project*

Anna Wróblewska[1], Teresa Podsiadły-Marczykowska[2], Robert Bembenik[1], Grzegorz Protaziuk[1], and Henryk Rybiński[1]

[1] Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warszawa, Poland
{A.Wroblewska,R.Bembenik,G.Protaziuk,H.Rybinski}@ii.pw.edu.pl
[2] Institute of Biocybernetics and Bioengineering, Trojdena 4, 02-109 Warszawa, Poland
tpodsiadly@ibib.waw.pl

Abstract. One of the main goals of the SYNAT project is to equip scientific community with a knowledge-based infrastructure providing fast access to relevant scientific information. We have started building an experimental platform where different kinds of stored knowledge will be modeled with the use of ontologies, e.g. reference/system ontology, domain ontologies and auxiliary knowledge including lexical language ontology layers. In our platform we use system ontology defining "system domain" (a kind of meta knowledge) for the scientific community, covering concepts and activities related to the scientific life and domain ontologies dedicated to specific areas of science. Moreover the platform is supposed to include a wide range of tools for building and maintenance of ontologies throughout their life cycle as well as interoperation among the different introduced ontologies.

The paper makes a contribution to understanding semantically modeled knowledge and its incorporation into the SYNAT project. We present a review of ontology building, learning, and integration methods and their potential application in the project.

Keywords: Ontology building, ontology maintenance, ontology integration, semantic modeling, ontology-based systems.

1 Introduction

SYNAT is a large scientific project aiming at creating a universal hosting and scientific content storage and sharing platform for academia, education and open knowledge society. It is the national project funded by the National Centre for Research and Development. The project has started in 2010 and will be carried out

* This work is supported by the National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 by the Strategic scientific research and experimental development program: „Interdisciplinary System for Interactive Scientific and Scientific-Technical Information".

R. Bembenik et al. (Eds.): Intelligent Tools for Building a Scient. Info. Plat., SCI 390, pp. 121–151.
springerlink.com

until 2013 by the research consortium comprising major Polish science institutions. Its purpose is to develop a system being a heterogeneous repository of data from various structured and unstructured sources [1]. The system is supposed to be capable of automatic acquisition of knowledge from web sources, including universities resources, such as, *inter alia,* researchers' homepages, research projects, tutorials, conference and workshop information. The structuring of information in the system will be based on ontologies, and will be stored in the SYNAT Knowledge Base (KB).

The purpose of KB is to provide all potential SYNAT users with a fast access to relevant scientific information. SYNAT "actors" may vary significantly in their needs - they may be researchers, lecturers, reviewers, students of all levels, publishers, librarians etc. Their requirements are not static, and may vary dynamically in time. The final system should thus possess a flexible functionality being smoothly adjustable to the changing users' requirements. The functionality of KB will not be "hard coded" into the system, but instead, it will be "ontology driven".

To this end, we plan to build a system ontology and provide tools for building many needed domain ontologies. The system ontology describes kinds of information provided by the system (covering concepts and activities related to the scientific life) and relationships between the system-based concepts. The system ontology is also thought as a "semantic definition" of the system scope and possible functionality. It can help in finding meaning of queries and building a sequence of searches in KB, as well as suggesting user additional options in the searching process. In addition, a number of domain ontologies will be incorporated into KB in order to provide a semantic support within specific research domains.

The system will provide access to different scientific information resources in one place and integration with these resources. A very important aspect of the integration will be sharing. On one hand, we want our system to make Polish national resources available to other, global scientific databases. Global resources worth mentioning include Web of Science (WoS) [2], Scopus [3] and Google Scholar (GS) [4]. Each of these sources has its specifics when it comes to the scientific fields covered. WoS is the leader in classical areas such as Physics and Chemistry [5]. Scopus is efficient for fields like Health and GS takes a leader role in integrating all domains (in some domains, like computer science it is already one of the best). GS has some major advantages over the other scientific databases: it is freely available, and relatively easy to use. Because of that, we plan to open national resources to the global scientific information services, including also such free services like GS or Microsoft Academic Search [6]. On the other hand we would like to make use of free web resources for integrating scientific services with the main information providers[1].

System users should be able to access resources available in local (e.g. university databases), national, and global systems in a convenient way. As a matter of fact, in existing services, searching for documents in a given subject does not take into account specific needs of a user. For example, a student may request some basic tutorials, a Ph.D. student will probably be looking for break-through documents on a particular subject and publications presenting it in detail, whereas a matured researcher is probably interested in the latest documents on the topics from his/her domain.

[1] To the most possible extent from legal and technical point of view.

Because of that we plan to equip the system with semantically-grounded technologies. Semantics is considered to be capable of dealing with the heterogeneity, massive scale and dynamic nature of resources on the Web. Semantic technology means application of techniques that support and exploit semantics of information (as opposed to syntax and structure/schematic issues) to enhance existing information systems [7]. Accordingly the semantics used in the Web is supposed to provide well-defined meaning enabling cooperation between machines and people [8]. Currently in more practical terms, Semantic Web technology also implies the use of standards such as RDF/RDFS and OWL. The semantic standards provide a good framework to represent, share and use heterogeneous knowledge but also allow for discovering new information [7]. Nowadays ontology driven information systems are used for information retrieval, integration and analysis across many areas of applications. Such systems are utilized for example in public e-employment services [9] or to fuse knowledge automatically extracted from different media types [10]. More and more researchers make efforts towards the development of ontology-based knowledge bases including aspects from ontology design and population, using "semantic" data, to automated reasoning and semantic query answering.

In this paper we present a general idea of the SYNAT KB in the context of ontologies and their usage in the system. Also, we introduce a preliminary version of the system ontology, which is the core ontology of the KB. The rest of the paper is structured as follows: Section 2 outlines the proposed knowledge base and presents the first version of the system ontology. Sections 3, 4 and 5 give a review of methods for maintaining ontologies throughout their life-cycle, especially ontology building, ontology learning and ontology integration appropriately. Section 6 concludes the paper presenting the state of the art of ontology life-cycle tools and outlines research plans.

2 Ontologies in the SYNAT Project

The main functionality of the knowledge base can be divided into the two following categories: (i) searching and acquiring information, and (ii) building customized ontologies. An extensive usage of ontologies in the system is planned not only for better understanding users' queries and obtained information but also for driving the information acquisition process.

2.1 General Ideas Concerning the Experimental SYNAT Knowledge Base

Potential Users and Usage Scenarios

The intended end-users of the knowledge base are people involved in research and scientific activities i.e. scientists coming from various fields of science, and with various levels of expertise. The users accessing the system may play various roles in the scientific life; they may be researchers, lecturers, reviewers, research organizers, administrators, etc.

One of the main aims of the system ontology is to support end-user dialog with the knowledge base and determine the system actions for answering the end-user queries.

The system ontology is also thought as a "semantic definition" of the system scope and possible functionality. It can help in resolving the query meaning and building a sequence of searches in the knowledge base as well as suggest to a user additional options in preparing the answer.

Searching in the Knowledge Base
The experimental user interface starts with a single Google-like field to be used by the end-user for specifying a query. The system applies the system ontology in the query analysis and tries to discover the "semantics" of a user query. If the process of understanding a query is positive, the system indicates a user possible ways of querying the knowledge base, and provides relevant answers. If the query parsing brings negative results, the system can still indicate to a user the most common ways of using the knowledge base. Additionally, the user's profile will be used to help the system in recognizing the sense of the query and/or user needs.

In addition, the system starts collecting associated information from various resources, e.g. definitions from Wikipedia, information from local system resources, as well as external resources on the Internet. If there is a need, an intelligent query assistant can try to resolve query ambiguity on its own or, if necessary, interact with a user. The assistant can suggest information associated with a query or other related topics.

Scope
The SYNAT knowledge base it thought first and foremost as the source of scientific information concerning the community members, their research, documents (publications, reports, etc.), organizations (scientific, governmental, etc.), events (conferences, workshops, seminars, etc,) and useful data resources (databases, home pages of individuals or institutions, etc.) [11]. Another field of usage is connected with evaluation of this type of information.

Knowledge Domains
The experimental KB should cover the following categories of knowledge:

- Information about academic and scientific community, modeled by the system ontology.
- Research domains modeled *inter alia* by domain ontologies (describing particular details of the domains and also ontologies defining scientific domains and their new topics).
- Analytical knowledge related to evaluation measures of various objects. Such knowledge is acquired by applying various evaluation algorithms.
- Auxiliary knowledge base which includes different kinds of objects e.g. dictionaries with grammatical forms (inflection, conjugation, declension), language oriented tools (POS taggers, gazetteers, etc.), semantics analysis tools (dictionaries of synonyms, homonyms), ontological or semi-ontological lexico-linguistic layers of the above ontologies.

System Architecture Outline

A general idea of the system architecture is shown in Fig. 1. Three parts can be distinguished in the system: (1) the searching subsystem, (2) the ontology subsystem, and (3) repositories, dictionaries and thesauri subsystem.

The Searching Subsystem

The searching subsystem includes the following main components:

- Query Analyzer - the basic tasks realized by this component are: parsing – splitting text into terms (also dates, complex terms, etc.), lemmatization, recognition of query language, mapping terms to ontologies (using linguistic layers, taking into consideration also recognition of language, allowing or depicting any disambiguation), query enrichment based on ontologies and their linguistic layer (synonyms). The result of query analyzing process is a structured query prepared for interpretation.
- Query Interpreter is responsible for building a query execution plan; it includes: query disambiguation (choosing the most possible interpretation; asking a user about possible interpretations in the case of a strong ambiguity), generating additional prompts for a user and requesting for additional possible contexts. Query Interpreter uses the word sense disambiguation algorithms (based on statistical and data mining methods), thesauri and ontologies including WordNet.
- Query Executor searches in and communicates between knowledge base and the many available resources. The main task of Query Executor is to choose resources relevant for the given query, and send to them properly structured queries. After receiving answers, it should remove duplicates and refine obtained answers.
- Results Evaluator assesses results of a given query in terms of quality (e.g. similarity of the returned documents to the query) and quantity. Moreover it performs appropriate text mining algorithms (e.g. in order to group the results thematically, or classify them by requested type of information), and ranks the results, based on similarity to the query.

The Ontology Subsystem

The ontology subsystem is envisaged as a component, which provides the entire functionality needed for accessing, maintaining and developing ontologies. The main parts of this subsystem are:

- Ontology Manager, which is the main component of the ontology subsystem, and enables other components' access to ontologies. The functionality of this component includes, among others, searching and retrieving entities from ontologies, adding new entities, and extracting parts of ontologies.

  Tools for ontology maintenance and enriching – it is a set of tools offering various functionalities helpful for maintaining and developing ontologies. It is envisaged that several methods for semi-automated ontology learning, and for integration of ontologies (e.g. comparison of ontologies at semantic level) will be available to the user. It covers discovering concepts or relations between concepts from text repositories, as well as language layers for the ontologies. This set also

includes tools supporting different types of ontology integrations (aligning, merging, mapping).

Besides functional components the ontology subsystem includes ontologies: the system ontology and domains ontologies.

Internal Resources

In the system we foresee using several types of internal resources. The main resources are repositories including text documents with their structured descriptions. Other types of internal resources are various kinds of dictionaries, thesauri, terminologies (e.g. dictionaries including proper names from a given domain, lemmas from a given language) which may be useful in methods for ontology integration and learning.

Descriptions of all accessible repositories (both internal and external) are stored in the repositories metadata component.

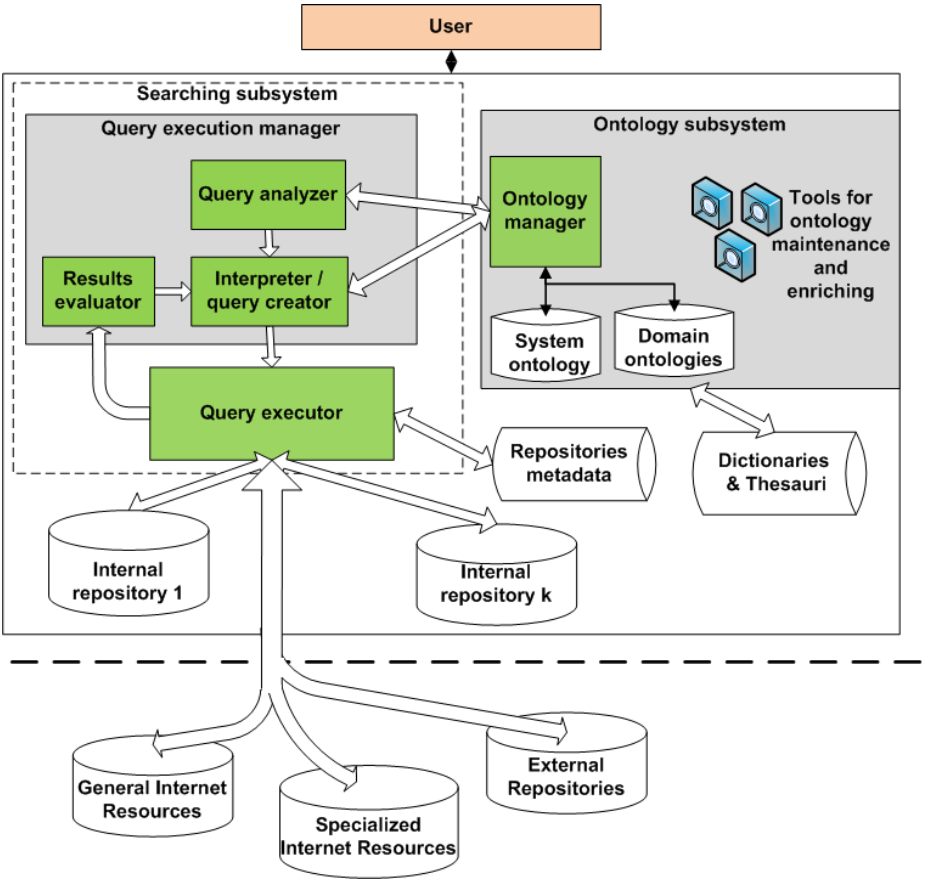

Fig. 1. A general idea of the knowledge base architecture

2.2 System Ontology

Construction of the System Ontology

The system ontology has been build using NeOn methodology [12], which decomposes the general problem of ontology construction into nine, more precisely stated, sub-problems, called scenarios [12], characterized in more detail in Section 3. Any combination of the scenarios is allowed for ontology building. All stages of the system ontology construction are a combination of four NeON scenarios, namely scenarios number 3, 4, 2 and 1.

The search for similar and potentially useful ontologies, performed using Watson plug-in from the NeON Toolkit, and searching for web resources using classical search engines (Google, Bing) revealed the existence of a large number of potentially useful ontologies (the detailed description of the chosen potentially useful ontologies is available in [11]). To this end, we decided to reuse existing ontological resources (scenario 3 in [12]).

The available ontological resources have been assessed as useful, but not exactly fitting our purpose. Therefore they were subject to modifications and reengineering (scenario 4 in [12]).

We also used non–ontological resources, in particular scientific domains from Google Scholar and Web resources characteristics elaborated by workers of Main Library of Warsaw University of Technology.

Finally, scenario 1 has been used to carry out *conceptualization*, *formalization*, and *implementation.* To formulate and validate the ontology domain and range, as well as ontology functional requirements, we formulated a set of competency questions (CQ) in the natural language. An ontology taxonomy has been constructed manually, based on the CQs. Concepts characteristics, and class definitions have been elaborated with the aim to enable CQ's answering and data control.

Ontology Formal Requirements

From the formal point of view, high-quality ontology should be: (1) consistent — all classes are consistent and can have instances; (2) semantically correct — should capture intuitions of domain experts, *inter alia*, clear and self-explanatory hierarchy of concepts and properties; (3) minimally redundant — no unintended synonyms should be present there; (4) sufficiently axiomatized — it should contain detailed descriptions, well-defined concepts (definitions with restrictions on properties, etc.), data and object properties defined with their domains and ranges [13].

Ontology Functional Requirements: Competency Questions

The core part of the system ontology is to characterize any scientist, giving general information about: personal and contact data, research interests and publications, education level, work positions, collaborators, activity in scientific events and teaching profile. The other aspects and information about academic life generally can be produced from the facts about scientists.

We collected about 50 competency questions for the ontology (the details are provided in [11]). They can be divided into the following groups:

- basic info about the *scientist* and the *research interest* (e.g. contact info, research areas);
- scientist's *education level* (e.g. achieved academic degrees, supervisors);
- *work activities* (e.g. affiliations, current and past work positions);
- *teaching profile* (e.g. supervisees, teaching activities, works reviewed, special lectures);
- participation in scientific events (e.g. taking part in scientific events, roles held at the events);
- *projects* (e.g. participation in the projects, kinds of projects, activities in projects);
- the authored *document*(s) (e.g. authored publications, patents, edited books);
- the *document profile* (e.g. publication data, publisher, referenced documents and citations);
- educational organization profile (e.g. education domains and research topics);
- additional possible CQs for *opinion* module (e.g. scientists working in a similar domain, similar projects).

The System Ontology – Current State

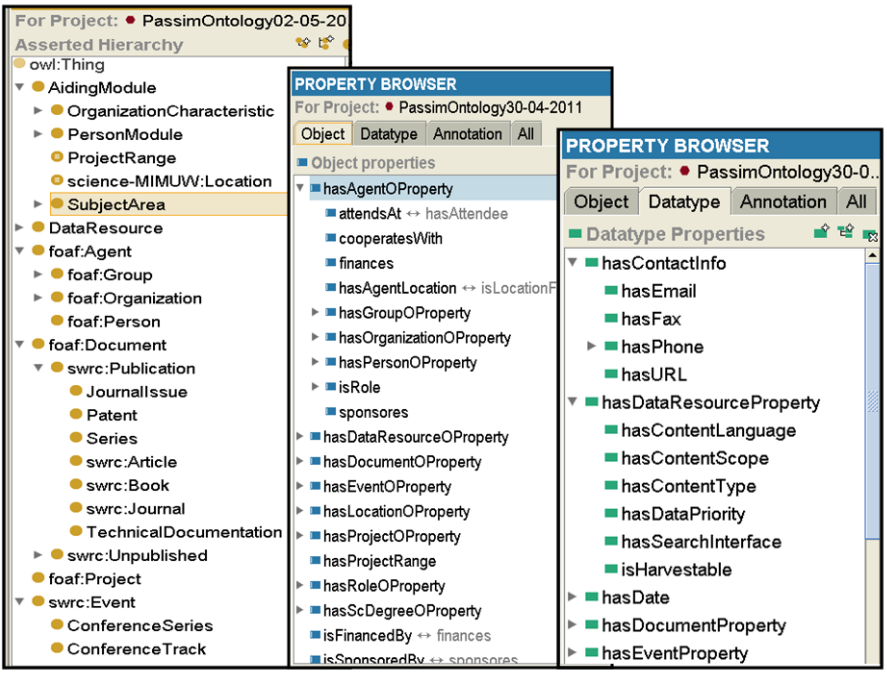

Fig. 2. The system ontology is composed of 5 main general modules: DataResource, Agent, Document, Project and Event (presented on the left panel). AidingModule contains notions used to define classes in the main ontology modules. The system core ontology counts: 175 classes (163 primitive, 12 defined) and 173 properties (objects and data types). Class definitions have been specified using universal, existential and cardinality restrictions.

The main goal of the system ontology[2] (Fig. 2) is to define basic metadata allowing to harvest repositories storing information about science communities from Web resources by specialized data mining algorithms. This goal dictates the scope of the ontology (basic facts about science and the academic community) and most of the modeling choices in the current version of the model.

The collected ontological and non-ontological knowledge sources are overlapping, but none of them covers overall the necessary scope of the system ontology. In the current model version we included notions from namespaces defined in the following ontologies: ESWC Conference ontology, FOAF, Science Ontology and SIOC. Other models have been used as a source of meaningful vocabulary.

The ontology has been specified in OWL-DL, edited with the Protégé-2000 editor [14], ver. 3.4.4. Documentation[3] has been generated by the NeON toolkit [15]), its consistency has been verified using the Pellet reasoner.

The main classes in the system ontology correspond to the following five main general modules: DataResources, Agents (such as Person, Group or Organization), Documents resulting from scientific work, and finally scientific Projects and Events. The overall structure of the system ontology is presented on the left panel in Fig. 2. The first additional ontology module, AidingModule, groups *role* notions that characterize classes in the main ontology modules, for example educational level, scientist's achievements, or organization character and activity. Those notions are used as restrictions fillers in definitions of classes specifying formally their semantics.

The system ontology formally defines terminology for data mining algorithms and metadata for the design of the system platform. However, its current version does not contain class definitions and modeling options allowing for deeper reasoning or concept similarity assessment. The future version of the system ontology will not only increase the scope and granularity of the ontology concepts, but also change some of the preliminary, simplistic modeling choices. For example some notions such as geographical scope, countries, cities and languages will be modeled as classes, not data properties.

With almost all concepts in the designed ontology necessary conditions and extended ranges of object and data properties are associated. Personal roles are modeled in two different manners: roles with important additional facts (e.g. dates of holding them) are modeled as concepts (reified relations) and other roles with no particular additional information are designed as binary relations (e.g. participating in project, being an author of a publication).

3 Ontology Building

The ontology subsystem in the general idea of KB should provide appropriate workflows and possibility to design process sequences for ontology development and maintenance defined by specific methodologies. Additionally, the constructed system ontology for KB was carried out with the means of a suitable methodology. In this section, we introduce ideas referring to ontology building methodologies.

---

[2] Available at http://wizzar.ii.pw.edu.pl/SYNAT-ontology/, specified in OWL-DL, edited in Protégé-2000 editor ver. 3.4.5, documentation generated using NeON Toolkit ver. 2.4.2.

[3] Glossary of terms for the ontology is available directly in the ontology documentation.

Ontology building is primarily a knowledge integration process. This means that albeit in theory handcrafting an entire ontology by hand is possible, in practice the only feasible way to build a reasonably complex ontology is *via* extracting information from other sources. Basically there are two types of information sources that might be considered for acquiring knowledge for ontology building: structured data sources (other ontologies, thesauri, dictionaries, semi-structured web content, text corpora), and unstructured data sources (unstructured web content, generic text archives and document repositories, generic web contents).

As mentioned, in practice the process of building and maintaining ontologies is always semi-automatic. Approaches usually used for this purpose include *inter alia* heuristics belonging to various areas (e.g. NLP, ontology resources reuse), statistics as well as data mining. So far many ontology building methodologies have been proposed. The presence of notions such as ontology development processes, activities and knowledge resources reuse, enable the division of ontology building methodologies into three main groups. TOVE [16], METHONOLOGY [17] and NeON methodology [12] are the most representative methodologies for the aforementioned groups.

TOVE contains the guidelines for general ontology building activities such as specification, conceptualization, formalization, implementation, documentation, and maintenance. It does not however accent the need for knowledge reuse. METHONOLOGY goes a step further, and defines the notions of ontology development process, maintenance activities and stresses the weight of knowledge reuse. Its disadvantage is the lack of explicit guidelines for knowledge reuse. The NeOn methodology derives and extends the main ideas from its predecessors. Its main asset is that it formalizes the development processes by introducing ontology development scenarios and puts stress on reengineering of ontological and non-ontological knowledge resources. In particular it proposes Glossary of Processes and Activities, which identifies and defines the processes and activities carried out when ontology is collaboratively built by teams. To this end we plan to base our ontology building platform on NeOn[4].

The NeON methodology is transparent and clear for users, because it defines each process or activity precisely, states clearly its purpose, inputs and outputs, and the set of methods, techniques and tools to be used. Above -described scenarios can be combined in different ways. Any combination of the scenarios should include Scenario 1, because this scenario is made up of the core activities that have to be performed in any ontology development.

One of the key elements in the NeOn methodology is the set of nine general scenarios proposed for building ontologies and ontology networks[12]. In particular, on the one extreme NeOn considers building an ontology from scratch (in [12] it is Scenario 1). We presume that this case does not happen even when dealing with a new research domain, as they usually emerge from existing areas. Another extreme is Scenario 9 (Localizing/nationalizing ontological resources). In this case, we presume that the concept layer of scientific domain ontologies is language independent. Therefore "localizing" ontology is limited to adding a specific language layer, and integrating it with the concept layer.

[4] We used a combination of the NeON scenarios in constructing the system ontology outlined in Section 2.2.

We recall here the scenarios 4, 5, 6, and 8,which will play the crucial role in our approach.

Scenario 4: Reusing and re-engineering ontological resources
Ontological resources re-engineering process comprises the following activities: ontological resources reverse engineering, ontological resources restructuring and ontological resources forward engineering. After carrying out the ontological resources re-engineering process, its result should be incorporated into the corresponding activity: specification, conceptualization, and formalization.

Scenario 5: Reusing and merging ontological resources
This scenario is realized when several possibly overlapping ontological resources in the same domain are available. They can be merged, or only alignments among them can be established in order to create the ontology.

Scenario 6: Reusing, merging and re-engineering ontological resources
This scenario has the same activities as Scenario 5, but in this case ontology developers decide not to use the set of merged ontological resources as they are but to re-engineer it. Once the set of merged ontological resources is re-engineered, the result of such a process should be integrated with the corresponding activity of Scenario 1.

Scenario 8: Restructuring ontological resources
Ontological resources restructuring can be performed in one of the following four ways: (i) modularizing the ontology in different ontology modules, (ii) pruning the branches of the taxonomy not considered necessary, (iii) extending the ontology including new concepts and relations, (iv) specializing branches that require more granularity and including more specialized domain concepts and relations.

4 Ontology Learning

Specialized tools which support the task of ontology engineering are necessary to reduce the costs associated with the engineering and maintenance of ontologies [18]. As data in various forms (textual, structured, visual, etc.) is massively available, many researchers have developed methods aiming at supporting the engineering of ontologies by AI based techniques, (e.g. machine learning, NLP, but also text mining) which can support an ontology engineer in the task of modeling a domain. Such data-driven techniques supporting the task of engineering ontologies have become to be known as ontology learning. Ontology learning has the potential to reduce the cost of creating and, most importantly, maintaining an ontology. This is the reason why a plethora of ontology learning frameworks have been developed in the last years and integrated with standard ontology engineering tools.

There are three kinds of data to which ontology learning techniques can be applied: structured (such as databases), semi-structured (HTML or XML, for example), as well as unstructured (e.g. textual) documents. The methods applied are obviously dependent on the type of data used. While highly structured data, as found in databases, facilitate the application of pure machine learning techniques, such as

Inductive Logic Programming (ILP), semi-structured and unstructured data requires some preprocessing, which is typically performed by natural language processing methods.

Ontology learning will play an important role in the Ontology Subsystem of SYNAT Knowledge Base. Methods of ontology learning will be employed in the process of creating, expanding and updating system and domain ontologies.

4.1 Ontology Learning Architecture

In [18] a generic ontology learning architecture and its main components are presented. The architecture is given in Fig. 3.

The ontology learning architecture is composed of the following components: *ontology management*, *coordination*, *resource processing* and *algorithm library*. The individual components are described below.

Ontology Management Component
Ontology management component is used by an ontology engineer to manipulate ontologies. Ontology management tools typically facilitate the import, browsing, modification, versioning and evolution of ontologies. However, the main purpose of the ontology management component in the context of ontology learning is to provide an interface between the ontology and the learning algorithms. When learning new concepts, relations or axioms, the learning algorithms should add them into the ontology model accessing the API of the ontology management component. Thus, the ontology management API should contain methods for creating new concepts, relations, axioms, individuals, etc. Most of the available APIs fulfill this requirement.

Coordination Component
This component is used by an ontology engineer to interact with the ontology learning components for resource processing, as well as with the Algorithm Library. A comprehensive user interfaces should support the user in selecting relevant input data that are further exploited in the discovery process. Using the coordination component, the ontology engineer also chooses among a set of available resource processing methods, and a set of algorithms available in Algorithm Library. A central task of the coordination component is to sequentially arrange and apply the algorithms selected by the user, passing the results to each other. A neat way for arranging the text mining process has been implemented in TOM [19].

Resource Processing Component
This component encapsulates techniques for discovering, importing, analyzing and transforming relevant input data. An important subcomponent is the natural language processing system. The general task of the resource processing component is to generate a pre-processed data set as input for the algorithm library component.

Resource processing strategies differ depending on the type of input data made available. Semi-structured documents, like dictionaries, may be transformed into a predefined relational structure. HTML documents can be indexed and reduced to free text. For processing free text, the system must have access to language-specific

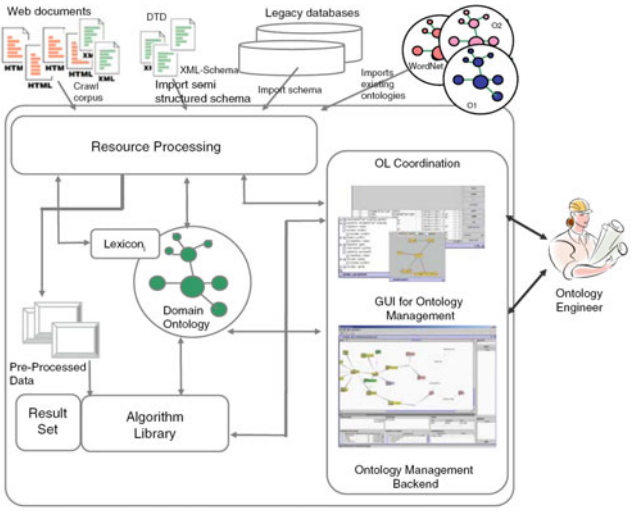

Fig. 3. Ontology learning generic architecture [18]

natural language processing systems. Some off-the-shelf frameworks, such as GATE [20], provide most of the functionality needed by ontology learning systems. The needed NLP components could be:

- A tokenizer and a sentence splitter to detect sentence and word boundaries.
- A morphological analyzer. For some languages a lemmatizer reducing words to their canonical form might suffice, whereas for languages with a richer morphology a component for structuring a word into its components (lemma, prefix, affix, etc.) is necessary. For most machine learning-based algorithms a simple stemming of the word might be sufficient.
- A part-of-speech (POS) tagger to annotate each word with its syntactic category in context, thus determining whether it is a noun, a verb, an adjective, etc.
- Regular expression matching allowing to define regular expressions and match these in the text.
- A chunker in order to identify larger syntactic constituents in a sentence. Chunkers are also called partial parsers.
- A syntactic parser determining the full syntactic structure of a sentence might be needed for some ontology learning algorithms.

Algorithm Library Component

Algorithm Library contains the algorithms applied to ontology learning. Depending on the knowledge discovery approach the library may contain machine learning algorithms (like in [18]), or text mining algorithms (like in [19]). There may also be quite efficient for some tasks algorithms looking for specific syntax patterns. In particular, typical tasks to be solved by the algorithms are as follows:

- association rule discovery – used to discover interesting associations between concepts;

- hierarchical clustering, used to cluster terms;
- classifiers, used to classify new concepts into an existing hierarchy.
- inductive logic programming – used to discover new concepts from extensional data;
- conceptual clustering – used to learn concepts and concept hierarchies.

4.2 Ontology Learning Algorithms

The tasks relevant in ontology learning have been organized in an *ontology learning layer cake* [21]. The ontology development process (manual or with automatic support) is primarily referring to the definitions of concepts and relations between them. It implies a need to acquire a linguistic knowledge about the terms that are used to refer to specific concepts in the texts, and their possible synonyms.

An important part of ontology building is discovering a taxonomy backbone (the relation *is-a*), as well as association relations (non-hierarchical ones). Finally, in order to derive facts that are not explicitly encoded by the ontology but could be derived, some rules should be defined (and if possible acquired). The layer cake presenting all the above aspects of ontology development is depicted in Fig. 4.

The layers build upon each other in the sense that results of the tasks at lower layers typically serve as input for the higher layers. For example, in order to extract relations between concepts, we should consider the underlying hierarchy to identify the right level of generalization for the domain and range of the relation. The two bottom layers correspond to the lexical level of ontology learning. The task in this part of the layer is to detect the relevant terminology as well as groups of synonymous terms. The extracted terms and synonym groups can then form the basis for the formation of concepts. Concepts differ from terms in that they are ontological entities and thus abstractions of human thought. According to the formalization, concepts are triples $c:=< i(c), [c], Ref_c >$ consisting of an intensional description $i(c)$, an extension $[c]$ and a reference function Ref_c representing how the concept is symbolically realized in a text corpus, an image, etc. At higher levels of the layer cake, we find the layers corresponding to the tasks of learning concept hierarchy, relations, a relation hierarchy, as well as, deriving arbitrary rules and axioms. The top two layers correspond to the most challenging task, as in principle there is no limit on the type and complexity of axioms and rules to be learned. In practice, however, as we commit to a specific knowledge representation language, the types of axioms allowed are usually restricted.

In the subsections to follow the learning layer cake will be presented in more detail (based on [18] and [22]).

Terms

Term extraction is a prerequisite for all aspects of ontology learning from text. Terms are linguistic realizations of domain-specific concepts and are therefore central to further, more complex tasks. There are many examples of term extraction methods that could be used as a first step in ontology learning from text. Most of them are based on information retrieval methods for term indexing, but many also take inspiration from text mining, as well as, terminology and NLP research.

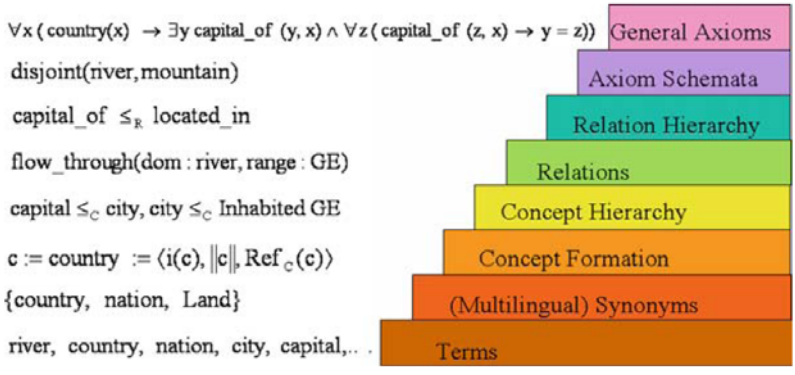

Fig. 4. Ontology learning layer cake [21]

Term extraction implies more or less advanced levels of linguistic processing, i.e. phrase analysis to identify complex noun phrases that may express terms and dependency structure analysis to identify their internal semantic structure. As such parsers are not always readily available, much of the research on this layer in ontology learning has remained rather restricted. The state-of-the-art is mostly to run a part-of-speech tagger over the domain corpus used for the ontology learning task and then to identify possible terms by manually constructing ad-hoc patterns, whereas more advanced approaches to term extraction for ontology learning build on deeper linguistic analysis. Additionally, and in order to identify only relevant term candidates, a statistical processing step may be included that compares the distribution of terms between corpora. A text mining approach (based on the T-GSP algorithm) presented in [23] has shown very good results for finding candidates for compound terms and proper names.

Synonyms

The synonym level addresses the acquisition of semantic term variants in a language, and between languages, where the latter in fact concerns the acquisition of term translations. Much of the work in this area has focused on the integration of WordNet for the acquisition of English synonyms, and EuroWordNet for bilingual and multilingual synonyms and term translations. An important aspect of this work is the identification of the appropriate (WordNet/EuroWordNet) sense of the term in question, which determines the set of synonyms that are to be extracted. This involves standard word sense disambiguation algorithms. However, specifically in the ontology learning context, researchers have exploited the fact that ambiguous terms have very specific meanings in particular domains allowing for an integrated approach to sense disambiguation and domain specific synonym extraction.

In contrast to using readily available synonym sets such as provided by WordNet and related lexical resources, researchers have also worked on algorithms for dynamic acquisition of synonyms by clustering and related techniques. On this basis much work has been done on synonym acquisition from text corpora that is based on distributional hypothesis that terms are similar in meaning to the extent in which they share syntactic contexts. Related work originates out of term indexing for information

retrieval, e.g. the family of Latent Semantic Indexing algorithms (LSI, LSA, PLSI and others). LSI and related approaches apply dimension reduction techniques to reveal inherent connections between words, thus leading to group formation. LSA/LSI-based techniques are interesting as they do not run into data sparseness problems such as approaches relying on raw data. In [24] a knowledge-poor text mining algorithm is presented. This algorithm shows interesting features and will be further upgraded.

Concepts
In [18], three paradigms of concept induction have been indicated:

- conceptual clustering;
- linguistic analysis;
- inductive methods.

Conceptual clustering approaches such as Formal Concept Analysis have been applied to form concepts and to order them hierarchically at the same time. Conceptual clustering approaches typically induce an intentional description for each concept in terms of the attributes that it shares with other concepts as well as those that distinguish it from other concepts.

Linguistic analysis techniques can be applied to derive an intentional description of a concept in the form of a natural language description. The definition of the term *knowledge management practices*: "*a kind of practice, knowledge of how something is customarily done, relating to the knowledge of management, the process of capturing value, knowledge and understanding of corporate information, using IT systems, in order to maintain, re-use and de-ploy that knowledge.*" is compositionally determined on the basis of the definitions *of knowledge management* and *practice*. For this purpose, disambiguation with respect to the different senses of a word with respect to its several meanings in a lexical database (such as WordNet) is required. Further, a set of rules is specified which drive the above compositional generation of definitions.

Finally, given a populated knowledge base, approaches based on *inductive learning* such as Inductive Logic Programming can be applied to derive rules describing a group of instances intentionally. Such an approach can for example be used to reorganize a taxonomy or to discover gaps in conceptual definitions.

Concept Hierarchy
Different methods have been applied to learn taxonomic relations from texts. Noteworthy approaches are based on matching lexico-syntactic patterns, clustering, phrase analysis as well as classification. Some text mining methods for finding close meaning pairs provide as a side effect the taxonomic relationships (e.g. [24]).

Relations
In order to discover arbitrary relations between words, different techniques from text mining, machine learning and statistical natural language processing community have found application in ontology learning. In order to discover 'anonymous' associations between words, one can look for a strong co-occurrence between words within a certain boundary, i.e., a window of words, a sentence or a paragraph. Association rule discovery algorithm can be utilized to represent co-occurrences of words within a sentence as transactions. This representation allows to calculate the support and

confidence for binary transactions and thus to detect anonymous binary associations between words. Good text mining results can be obtained with the algorithms mining for patters. The algorithm T-GSP presented in [19], which enables searching for specific patterns expressed by regular expressions, can be efficiently applied for discovering associations relationships between concepts [25].

In the computational linguistics community, the task of discovering strong associations between words is typically called collocation discovery. The idea here is to discover words which co-occur beyond chance in a statistically significant manner. Statistical significance is typically checked using some test such as the Student's t-test or the χ^2-test. Other researchers have aimed at learning labeled relations by relying on linguistic predicate argument dependencies. Typically, verb structures are considered for this purpose. When learning relations, a crucial issue is to find the right level of abstraction with respect to the concept hierarchy for the domain and range of the relation in question.

Rules

The success of OWL that allows for modeling far more expressive axioms has led to some advances in the direction of learning complex ontologies and rules.

A research towards generating formal class descriptions from extracted natural language definitions, e.g. from online glossaries and encyclopedias has been started recently. The implementation of this approach is essentially based on a syntactic transformation of natural language definitions into OWL DL axioms in line with previous work on lexico-syntactic patterns and lexical entailment.

One of the first methods for learning disjointness axioms relies on a statistical analysis of enumerations which has been implemented as part of the Text2Onto framework [26].

Evaluation

[27] and [18] argue there are not many gold standards that could be used for evaluation of ontology learning methods. The desired result of ontology learning is not a simple list with binary classifications, but a far more complicated structure. To make it even worse, there is no clear set of knowledge-to-be-acquired, not even for very specialized domains. There are several possibilities of conceptualizations for one domain that might differ in their usefulness for different groups of people, but not in their soundness and justification. So even if the outcome of an algorithm does not compare well with a manually built ontology, how can its quality be judged?

However, several approaches at approximating the appropriateness of ontology learning methods have been done. One of them [28] tries to measure the 'corpus fit' of the ontology by considering the frequency with which the terms in the ontology appear in the corpus. AEON framework [29] aims to automatize the application of the OntoClean methodology, hence ensuring the formal consistency of an ontology. [30] proposes integration of ontology learning and evaluation. It is an approach to exploiting contextual information, or confidence and relevance values for resolving logical inconsistencies in learned ontologies, and to optimize the outcome of the ontology learning process.

4.3 Ontology Learning Tools

Since building an ontology for a huge amount of data is a difficult and time-consuming task a number of tools have been developed in order to support the user in constructing ontologies from a given set of (textual) data. Some well-known and frequently cited tools include TextToOnto [31], Text2Onto [26], OntoLT [32] and OntoLearn [33] (TextToOnto was originally integrated into the KAON ontology engineering environment, OntoLT was integrated with Protégé and Text2Onto has been recently integrated with the NeOn Toolkit).

All these tools implement various methods. OntoLearn for example integrates a word sense disambiguation component to derive intentional descriptions of complex domain-specific terms, which are assumed to denote concepts, on the basis of WordNet glosses. In this sense, OntoLearn also induces intentionally defined domain concepts and ingeniously exploits the knowledge available in general resources for a specific domain. OntoLT, which is available as a plugin to the Protégé ontology editor, allows for term extraction using various measures such as *tf.idf* and extraction of taxonomic relations relying on interpreting modifiers (nominal or adjectival) as introducing subclasses.

TextToOnto is a framework containing various tools for ontology learning. It includes standard term extraction using a number of different measures, the algorithm for mining relations based on association rules as well as hierarchical clustering algorithms based on Formal Concept Analysis. Its successor, Text2Onto, besides implementing most of the algorithms available also in TextToOnto, abstracts from a specific knowledge representation language and stores the learned ontology primitives in the form of a meta-model called Possible Ontologies Model (POM), which can then be translated to any reasonably expressive knowledge representation language, in particular to OWL and RDFS. On the other hand, it implements a framework for data-driven and incremental learning in a sense that changes in the underlying corpus are propagated to the algorithms, thus leading to explicit changes to the POM. The advantage is that these changes can be easily traced back to the original corpus changes, which gives more control to the ontology engineer.

5 Ontology Integration

In the ontology integration area one can distinguish two main problems: building one ontology from two or more existing ontologies, and expressing entities from one ontology by means of entities from another ontology. Both problems may occur during the system usage, but the first one is especially important as it is expected that several domain ontologies will be created. Below, we describe the problem of ontology integration in detail and introduce solutions which can be adopted to the system.

5.1 Problem Outline

The integration of two or more ontologies generally refers to a process of identification and establishing correspondences between elements of these ontologies. However, ontologies may differ from each other in many aspects: language types in

which they are expressed, structure, domains, etc., which makes the integration process potentially very difficult and time-consuming. The ontology integration process is heterogeneous and various characteristics of it have been proposed in the literature. One can distinguish:

- *semantic integration* (focused on intended meaning of the concepts), *structural integration*, and *syntax integration* (focused on providing uniform formalism in which integrated ontologies are expressed);
- *global-centric approach*, where concepts of the global ontology are mapped into the local ontologies and *local-centric approach*, where concepts of the local ontologies are mapped into a global ontology;
- type of integration: mapping, merging, alignment, translation, etc.

Moreover there are several types of mismatches between integrated ontologies. The mismatches can refer to: *language* – formalisms used for expressing ontologies, *terminology* (synonyms, homonyms), *encoding* (different value formats e.g. grams vs. pounds), *paradigms* – different ways of representation of concepts (subclasses vs. attributes), *content level* – differences in the scope and the level of details in which a given domain is modeled in ontologies.

5.2 Types of Integration

In the literature several different types of integration of ontologies have been introduced, the most common are: merging, mapping/matching and alignment. Unfortunately, there is no common understanding of these notions and given definitions differ considerably. Below, the review of definitions and approaches to these three types of integration is provided.

Ontology Merging

The ontology merging process refers to creating a new ontology from two or more input ontologies. The new ontology should be a unification of original ontologies and should be capable of replacing them. This definition is quite common and used in many publications e.g. [34]. In the paper [38] a bit different approach has been proposed. The ontology merging has been defined as "the minimal union of vocabularies S_1 and S_2 and axioms A_1 and A_2 of two ontologies $O_1 = (S_1, A_1)$ to $O_2 = (S_2, A_2)$ that respects their articulation. Articulation of ontologies is a result of the mapping process. The exemplary approaches to ontology merging are presented below.

FCA-merge method for ontology merging has been proposed in [39]. It is the bottom-up approach in which techniques taken from Formal Concept Analysis (FCA) theory are applied. The merged ontology is generated from two input ontologies and a repository of text documents relevant to the input ontologies. The method consists of three steps: context generation, computing a concept lattice and final ontology generation. In the *context generation* step for each ontology a formal context is created. A formal context has been defined as $K = (G, M, I)$, where G - is a set of documents; M - set of concepts and I - binary relations, for $g \in G$, $m \in M$, a relation $(g, m) \in I$ means that document m contains an instance of m. The domain-specific lexicon is used for finding instances of concepts in documents. In the *computing a concept*

lattice step the pruned concept lattice is created by merging two formal contexts obtained in the previous step. In the *final ontology generation* step the merged ontology is generated from the pruned concept lattice. Several heuristics are applied in order to generate automatically candidate entries to the final ontology. However, in this step user interaction is required as several actions cannot be fully automated, e.g. possible conflicts or duplicates resolving.

The PROMPT algorithm for ontology merging and alignment has been introduced in [40]. The method supports the user in creation of one ontology from two input ontologies. In the process the following steps can be distinguished:

1. Creation of an initial list of matches based on names of classes. The appropriate similarity measure should be provided by the user.
2. Selection of an operation to be executed by the user. Each operation has been assigned one of the following: (1) changes that can be done automatically; (2) new suggestions; and (3) conflicts that the operation may introduce. The set of available ontology-merging operations includes among others: merge classes or slots, merge bindings between a slot and a class, or perform a shallow copy of a class.
3. Execution of the selected operation, automatic execution of additional changes based on the type of operation.
4. Generation of a list of suggestions for the user based on the structure of the current version of a created ontology.
5. Determination of conflicts that the last operation introduced in the ontology and finding possible solutions for those conflicts.

The steps 2-5 are realized in a cycle.

Ontology Mapping

An ontology mapping refers to the way in which one ontology may be expressed in entities from another ontology. The mapped ontologies are not changed, rather an additional specification (functions, set of axioms, queries) describing correspondence between entities is provided. Often mapping is done only in one direction e.g. the entities of one ontology are connected with entities of the second ontology but not vice versa. One of the main ontology mapping purposes is to allow collaboration between various systems which use various ontologies.

Various ideas of ontology mappings can be found in the literature. Here we present the definitions which seem to be the most representative. In [38] ontology mapping is defined as a morphism of ontological signatures. An ontology has been defined as a pair $O = (S, A)$ where S is the (ontological) signature describing the vocabulary, A is a set of (ontological) axioms, which indicates the intended meaning of the vocabulary. Based on this definition an ontology mapping from $O_1 = (S_1, A_1)$ to $O_2 = (S_2, A_2)$ is specified as morphism f: $S_1 \rightarrow S_2$ of ontological signatures such that $A_2 \models f(A_1)$. In [41] mapping is seen as a process of finding a semantic mapping between original ontologies. The authors concentrate here on finding one-to-one mapping between taxonomies. In [35] the following interpretation of mappings between two ontologies has been introduced: "for each entity (concept, relation, attribute, etc.) in one ontology we try to find a corresponding entity in the second ontology, with the same or the closest intended meaning; usually this correspondence is expressed by 1 to 1 functions." In [36] mapping is defined as "a (declarative) specification of the semantic

overlap between two ontologies". Such representation of mapping indicates correspondences between entities from two ontologies and is not incorporated in definitions of these ontologies. According to the authors such correspondences are typically expressed as a set of axioms in a specific mapping language. The exemplary approaches to ontology mapping are presented below.

In [42] the problem of mapping between a global ontology and a local ontology is discussed. The authors stated that it is better to express mapping concepts from one ontology to concepts from another one as a view or query over the second ontology, rather than a certain match function. Such approach requires defining a query language, which should be included in the specification of a given ontology. An ontology has been defined as a pair $O = (L, A)$, where L is a theory in a logic and A stands for the alphabet of terms used in the ontology. The authors propose a formal framework for ontology integration systems. The framework is defined as a triple $(G, S, M_{G,S})$ where:

- $G = (L_G, A_G)$ – is a global ontology,
- S is a set of local ontologies – it consists of n ontologies denoted by $S_1, \ldots, S_n$, $S_i = (L_{Si}, A_{Si})$;
- $M_{G,S}$ is a mapping between global ontology and the local ontologies.

The conceptual framework for ontology mapping process has been proposed in [43]. The framework consists of five horizontal modules reflecting the distinct phases in a mapping process. These modules are *Lift & Normalization*, S*imilarity*, *Semantic Bridging*, *Execution* and *Post-processing*. The mapping between ontologies is represented by so-called semantic bridges. The semantic bridges are specified in the five following dimensions:

1. *Entity dimension* – indicates correspondences between ontology entities i.e. between concepts, relations, and attributes.
2. *Cardinality dimension* indicates the number of ontology entities at both sides of the semantic bridge (1:1, 1: n, n:m).
3. Structural dimension indicates the way how elementary bridges may be combined into more complex bridges. The following relations have been specified between bridges: specialization, alternatives, composition, and abstraction.
4. *Constraint dimension* is used for defining constraints concerning instances. The fulfillment of these constrains is required for executing transformation rule associated with a given bridge.
5. *Transformation dimension*: describes how instances from the source ontology are transformed during the mapping process.

The GLUE system has been described in [41]. In this system learning techniques are used for creating semi-automatically semantic mapping between ontologies, which are expressed in the same way i.e. the concepts are specified in a comparable manner. The system consists of three main components, namely: *Distribution Estimator*, *Similarity Estimator* and *Relaxation Labeler*. The *Distribution* Estimator module takes as input two ontologies O_1 and O_2 (also instances of concepts are included into input data) and for every pair of concepts $< a, b >$ such that $a \in O_1$, $b \in O_2$, computes joint probability distribution. In the *Similarity* Estimator module the result obtained from Distributed Estimator component is used for calculating a similarity function.

The output of the module is a similarity matrix between the concepts in the two taxonomies. In the *Relaxation Labeler* module, based on the similarity matrix and additional to ontologies domain constraints and heuristic knowledge, generates mapping configuration which best satisfies the domain constraints.

In [44] the ontology mapping process has been defined based on information-flow theory proposed by Barwise and Seligman [45]. Authors analyzed the problem of mapping two ontologies: a reference ontology without instances and a local ontology populated with instances associated to concepts. The proposed method allows carrying out mapping between ontologies expressed in Horn logic. The mapping is defined in terms of logical infomorphis between local logics. A local logic has been specified as quadruple $L = (I,T,\models,\perp)$ where I is a set of instances, T is a set of types, $\models$ is a classification relation between elements of I and T, and $\perp$ stands for a consequence relation between subsets of T. In the process every considered ontology has a local logic associated with it.

The H-match algorithm for ontology matching has been introduced in [46]. In the algorithm an ontology is seen as a set of concepts, properties, and semantic relations. In the method two sets of features are distinguished, namely linguistic and contextual. *Linguistic features* refer to names of ontology elements and their meaning. The meaning of names for ontology matching is determined based on a thesaurus of terms and weighted terminological relationships among them. In H-Match the thesaurus is automatically derived from the lexical system WordNet. *Contextual features* of a concept c refer both to the properties and to the concepts directly related to c through a semantic relation in an ontology. The context $Ctx(c)$ of a concept c defined as: $Ctx(c) = P(c) \cup C(c)$ where $P(c)$ is a set of properties of c and $C(c)$ is a set of concepts that participate in a semantic relation with c. In H-match the four following models of matching have been defined:

- *surface* - matching based only on names of ontology elements;
- *shallow* - matching based on names and concept properties;
- *deep* - matching based on names, concepts properties and semantic relations;
- *intensive* - matching based on names, concepts properties, semantic relations, and property values.

Ontology Alignment

The alignment of one ontology with another means that for each entity (concept, relations, instance, etc.) from the first ontology the corresponding object is found from the second one. The matched entities should have the same meaning. The equality alignment, one-to-one matching, is the most investigated type of alignment.

Here we provide selected definitions of ontology alignment; some others can be found in the literature. [38] defines ontology alignment as “the task of establishing a collection of binary relations between the vocabularies of two ontologies”. The authors introduce a common intermediate source ontology O_0 - articulation of two ontologies, and decomposed a binary relation into a pair of mappings from the intermediate ontology to aligned ontologies. In [35] ontology alignment is defined as “the process of bringing two or more ontologies into mutual agreement, making them consistent and coherent with one and another”. In such a process analyzed ontologies may be affected, e.g. contradictory relations may be removed. In [36] ontology

alignment is defined as a process of discovering similarities between two input ontologies, the result of which is a specification of similarities between these ontologies. In [37] the formal alignment function *align* has been defined for the *equal* relation, the appropriate formula is given below.

$$align: E \times O \times O \rightarrow E, \tag{1}$$

where: E - a set of all entities, O – a set of possible ontologies.

In practice objects may be aligned by using other than equal relations e.g. subsumption. The formal general function *gen_align* for alignment has been defined in [37] as:

$$gen\_align: E \times O \times O \rightarrow E \times M, \tag{2}$$

where: E is a vocabulary, all terms $e \in E$; O is the set of possible ontologies, M is the set of possible alignment relations.

In general, the alignment of ontologies can be defined as a set of quadruples $< x, y, r, l>$, where x and y stand for aligned entities, r is a relation between matched entities, l expresses a level of confidence of a given alignment (it is an additional element). The exemplary approaches for ontology alignment are presented below.

In [36], [34] the general process for ontology alignment has been proposed. The process is iterative and consists of the five following steps:

- In the *Feature Engineering* step the parts of ontologies definition are chosen for the needs of the description of entities to be aligned. The selected features which characterize entities depend on the structures and languages, on which the integrated ontologies are described.
- In the Search Step Selection the pairs of entities for comparison are selected. The typical methods for candidate pairs selection are: (1) Cartesian product of sets: E_1 – consisting of all entities from O_1 and E_2 – consisting of all entities from O_2; (2) all pairs of entities of the same type (concept, relation, instance).
- In the *Similarity Computation* step the similarity between entities in candidate pairs are determined. The similarity is calculated based on selected features which exist in both ontologies and an adequate similarity measure. More than one measure may be applied for calculating similarity between entities in one pair (e.g. one measure for one feature) and also similarity measures may vary for different types of compared entities.
- In the *Similarity Aggregation* step the single value indicating similarity between entities is calculated. Typically, the value is an average value for similarity measures used in the previous step or the weighted average value. The weight may be assigned manually or may be determined, e.g. by using data mining or machine learning algorithms on a training set.
- In the *Interpretation* step a decision whether two entities should be aligned is made based on calculated similarity measures. In the literature several methods for this purpose have been introduced. In the simplest methods, a threshold is used i.e. the alignment is made if the aggregate similarity measure is greater than a given threshold θ.

In [47] Integrated Learning In Alignment of Data and Schema (ILIADS) algorithm has been introduced. The method allows for aligning two ontologies written in OWL-lite language by means of sets of entities classes, properties, individuals, data values, triples and axioms. The proposed algorithm finds a set of axioms and facts that link entities in aligned ontologies. In the algorithm the following general steps can be distinguished:

1. Creation of an ontology O from input ontologies O_1,O_2; $O = O_1 \cup O_2$;
2. Finding candidate groups of similar entities in ontologies by using a hierarchical agglomerative clustering method;
3. Determination of an equivalence or subsumption relationships between the sets of entities;
4. Computing the logical consequences and logical inference similarity of the new relationships (a relationship models cluster created by merging two groups) based on the current state of O;
5. Addition of the axiom of the highest logical similarity to a set of alignment and update ontology O.

5.3 Similarity Measures

Similarity is the key issue in many ontology integration methods. Typically comparison of ontologies is performed by comparison of their entities. In the comprehensive methods not only lexical or syntax level of entities should be taken into consideration but also their meaning and usage. In order to compare two ontologies various measures should be applied, adequate for given types of compared objects. The examples of measures for simple data type, set of entities and concepts are presented below.

Measures for Data Types

Equality
In some situations it is reasonable to assume that two entities are similar only if their data values are equal. In the simplest case a similarity function returns 1 if compared values are equal and 0 in other situations. In the context of ontology objects equality can be determined by using existing logical assertions. These assertions may be included in the definition of an ontology or may be determined manually or by using automated or semi-automated methods. The appropriate similarity function is presented below.

$$sim_{eq\_\text{object}}(o_1, o_2) = \begin{cases} 1, \text{ if assert}(o_1, o_2) \\ 0 \text{ otherwise} \end{cases} \quad (3)$$

String Values
The syntactic similarity measure for string values is defined by using equation 4. In this measure the edit distance of two strings is used.

$$sim_{syntactic}(v_1, v_2) = \max(0, \frac{\min(|v_1|, |v_2|) - ed(v_1, v_2)}{\min(|v_1|, |v_2|)}) \quad (4)$$

where: *ed*(*a, b*) is the edit distance between two strings; v_1. v_2 are string values; |v| is the number of characters in *v*.

Number from given limited range

$$sim_{diff}(n_1, n_2) = 1 - (\frac{|n_1 - n_2|)}{mdiff})^{\gamma} \quad (5)$$

where: n_1, n_2 – numbers from a given range; *mdiff* – difference between the maximal and minimal value for considered data type; and γ - a number, $\gamma > 0$.

Functions for Sets

Dice Coefficient

$$sim_{dice}(A, B) = \frac{2*|A \cap B|}{|A| + |B|} \quad (6)$$

where A and B are set of individuals.

In order to use the Dice coefficient for comparing two sets of entities it is necessary that entities from these two sets are identified by the same method, which makes it possible to determine whether a given entity belongs to one or both of these sets. In case of comparing ontologies this requirement may be difficult to fulfill.

Methods Based on Similarity between Individuals

Similarity between two sets of objects may be calculated by using similarities between individuals belonging to these sets. The methods of this kind rely on other measures which can be applied for calculating similarity between single entities. Three of them are presented below:

1. *Min linkage* - the minimum of similarity between individuals is used:

$$sim_{link\_\min}(A, B) = \min_{(a,b)|a \in A, b \in B}(sim(a, b)) \quad (7)$$

2. *Max linkage* - the maximum of similarity between individuals is used:

$$\textit{Max linkage:} \qquad sim_{link\_\max}(A, B) = \max_{(a,b)|a \in A, b \in B}(sim(a, b)) \quad (8)$$

3. *Average linkage* - the average similarity between individuals is applied:

$$\textit{Average Linkage:} \quad sim_{link\_avg}(A, B) = \frac{\sum_{\forall (a,b)|a \in A, b \in B} sim(a, b)}{|A| * |B|} \quad (9)$$

Similarity Measures for Entities

Label Similarity of Concepts

The basic property of concepts in ontologies is label. Labels are names of entities very often given by people and somehow indicating the meanings of concepts. For measuring the similarity of concepts based on their labels the syntactic similarity function may be used:

$$sim_{label}(c_1, c_2) = sim_{syntactic}(label(c_1), label(c_2)) \tag{10}$$

where: c_1, c_2 – concepts; label(c) – label of concept c.

Taxonomic Similarity for Concepts
One of the most known generic measures of similarity of concepts in one concept hierarchy has been presented in [48]. The formula of this function is shown below.

$$sim_{taxonomic}(c_1, c_2) = \begin{cases} e^{-\alpha l} * \dfrac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, \text{ if } c_1 \neq c_2 . \\ 1, \text{ otherwise} . \end{cases} \tag{11}$$

where: $\alpha \geq 0$, $\beta \geq 0$ are parameters used for scaling impact of respectively the shortest path of length l; and the depth h in the taxonomy hierarchy.

Similarities Based on Instance
Several different similarity measures of concepts have been defined based on joint probability distribution between them. Joint probability is calculated with reference to the sets of instances of considered concepts. Joint probability between two concepts A and B consists of the four following probabilities:

- $P(c_1, c_2)$ – the probability that a randomly chosen instance belongs both to a concept c_1 and a concept c_2.
- $P(c_1, \sim c_2)$ - the probability that a randomly chosen instance belongs to a concept c_1 but not to a concept c_2.
- $P(\sim c_1, c_2)$ - the probability that a randomly chosen instance belongs to a concept c_2 but not to a concept c_1.
- $P(\sim c_1, \sim c_2)$ - the probability that a randomly chosen instance does not belong to a concept c_1 nor to a concept c_2.

Jaccard Coefficient for Concepts
In [41] Jaccard coefficient is used as "exact" similarity measure:

$$sim_{concept\_jaccard}(c_1, c_2) = \frac{P(c_1, c_2)}{P(c_1, c_2) + P(c_1, \sim c_2) + P(\sim c_1, c_2)} \tag{12}$$

where c_1 and c_2 are concepts

The lowest value for this measure is *0* when concepts c_1 and c_2 are disjoint. The highest value is 1, when c_1 and c_2 are the same concept.

Concept Similarity of Instances
In [37] a similarity measure for instances of two concepts has been proposed. The measure is based on similarity of concepts, the proposed function is given below.

$$sim_{parent}(i_1, i_2) = sim_{object}(c_1, c_2) \tag{13}$$

where i_1, i_2 are instances and $i_1 \in instances(c_1)$, $i_2 \in instances(c_2)$; $instances(c)$ is the set of instances of a concept c.

Similarity Based on Context

If the context of usage of two concepts from an ontology is available the similarity between these concepts may be expressed with respect to the context. In [37] the following function has been proposed:

$$sim_{use}(e_1, e_2) = sim_{diff}(usage(e_1, cnxt), usage(e_2, cnxt)) \tag{14}$$

where: e_1, e_2 are entities from the ontologies (e.g. concepts or instances of concept); $usage(e, cnxt)$ returns the frequency of usage of entity e in the context $cnxt$; sim_{diff} is the function defined in equation 5.

In [49] the Wordnet synsets have been used for measuring similarity of concepts. The proposed definitions are based on definitions of similarities between two coherent intensional meanings of concepts introduced in [50]. Assuming that: each sense in a Wordnet syntset indicates a concept; O_1, O_2 – ontologies; c_i, c_j – concepts, $c_i \in O_1$, $c_j \in O_1$, $S[c_n]$ – region of a synset including concept c_n the four following levels of similarity have been defined:

- *Disjoint*: concepts c_i, c_j are disjoint if the conjunction of their synsets definition implies false: $S[c_i] \wedge S[c_j] \equiv$ false.
- *Specialized*: concept c_i is specialization of concept c_j if synset of c_j is an implication of the synset of c_i: $S[c_i] \wedge S[c_j] \equiv S[c_i] \Rightarrow c_i \leq c_j$.
- *Overlapping*: if the conjunction of synsets for c_i and c_j constitutes region for another concept c_k: $S[c_i] \wedge S[c_j] \equiv S[c_k]$.

6 Summary

In the paper we have presented a general idea of the role of ontologies in an experimental knowledge-base in the SYNAT project. In our system ontologies are considered to be pervasive, and utilized for modeling almost all aspects of the information stored in the Knowledge Base. Furthermore, we outlined details of the system ontology that is planned to be used as a reference for meta knowledge in the platform. We shortly introduced processes, methods, and tools for ontology maintenance, learning and integration which are assumed to be applied or adapted in our system especially in the ontology subsystem. It allows us to provide system which fulfills all users' requirements related to building, maintaining and integration of ontologies.

The planned research refers to the domain of ontology engineering, including constructing, and widely understood maintenance. It can be divided into the following three main topics: ontology learning, ontology constructing and applications, and ontology integration.

6.1 Ontology Learning

Research in the ontology learning area will be concentrated on automatic or semi-automatic methods that can be used for producing candidate entries for a given

ontology. Typically, such methods consist of two general phases: discovering terms and relations between terms and generation of candidate entries for a given ontology. In the former phase various techniques from natural language processing, statistics, and text mining areas are applied for exploration of text-document repositories in order to discover interesting terms and relations between terms. Here, the most common tasks include:

- compound term discovery;
- discovery of synonyms and homonyms;
- taxonomy discovery and/or validation;
- discovery of semantic relationship between terms for ontology building support;
- discovering relation signatures.

In the latter phase candidate entries are generated based on the results from the first phase. The entries may concern both lexical and concept layers (also instances), e.g.: a concept, a semantic relation between concepts (concept layer), and synonyms for a given term (lexical layer).

The goal of the planned research is to develop methods for discovering entries for a given ontology from text repositories. These methods will be based on text- and data-mining techniques, mostly with shallow grammatical analysis of texts and methods in which ontologies will be incorporated in algorithms used in both discovering and candidate generation phases. In particular, we plan to upgrade some methods worked out earlier by the team ([19], [23]) and to test new text mining-based approaches.

6.2 Ontology Constructing and Applications

The research concerning ontology building area will concern methods which may be applied for extending ontologies, especially the system ontology, and ways of application of it in the searching systems. The former task is concentrated on ways of extending and modeling notions on the basis of the SYNAT ontologies. It involves, among others, research on linking concepts between the system ontology and domain ontologies. The latter task is focused on applying ontologies for the evaluation of results in searching systems.

6.3 Ontology Integration

In the ontology integration area the planned research concerns both mentioned in the paper problems: building one ontology from two or more existing ontologies and expressing entities from one ontology by means of entities from another ontology. Particularly it will be focused on:

- methods in which the context associated with entries of ontologies is explored in order to obtain better correspondence between entries coming from merging ontologies;
- complex methods for measuring similarity between entries of ontologies based on all available layers of ontologies (e.g. concept and lexical layers).

References

[1] Gawrysiak, P., Ryżko, D.: Acquisition of scientific information from the Internet: The PASSIM Project Concept. In: Proc. of ICAART 2011, Rome (2011)

[2] Web of Science, http://thomsonreuters.com/products_services/science/science_products/a-z/web_of_science/

[3] Scopus, http://www.scopus.com/home.url

[4] Google Scholar, http://scholar.google.com

[5] Etxebarria, G., Gomez-Uranga, M.: Use of Scopus and Google Scholar to measure social science production in four major Spanish universities. Scientometrics 82, 333–349 (2010)

[6] Microsoft Academic Search, http://academic.research.microsoft.com/

[7] [SR03] Sheth, A., Ramakrishnan, C.: Semantic (Web) Technology In Action: Ontology Driven Information Systems for Search, Integration and Analysis. Bulletin of the Technical Committee on Data Engineering 26(4), 40–48 (2003)

[8] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American (May 2001)

[9] Della Valle, E., Cerizza, D., Celino, I., Estublier, J., Vega, G., Kerrigan, M., Ramírez, J., Villazon, B., Guarrera, P., Zhao, G., Monteleone, G.: SEEMP: An Semantic Interoperability Infrastructure for e-Government Services in the Employment Sector. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 220–234. Springer, Heidelberg (2007)

[10] Project Boemie, financed by sixth framework programme UE (2006-2009), http://www.boemie.org/

[11] State of the Art on Ontology And Vocabulary Building & Maintenance Research And Applications Solutions for System and Domain Ontologies. Technical Report B12 in SYNAT project, Institute of Computer Science, WUT (March 2011)

[12] Suárez-Figueroa, M.C., et al.: Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks. NeOn Deliverable D5.4.3, NeOn Project (January 2010), http://www.neon-project.org

[13] Ye, J., Coyle, L., Dobson, S., Nixon, P.: Ontology-based models in pervasive computing systems. The Knowledge Engineering Review 22(04), 315–347 (2007)

[14] Protégé Editor, http://protege.stanford.edu

[15] NeOn Toolkit, http://neon-toolkit.org

[16] Gruninger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies. In: Proc. Int. Joint Conf. AI Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal (1995)

[17] Fernández-López, M., Gómez-Pérez, A., Jurysto, N.: METHONOLOGY: From Ontological Art Towards Ontological Engineering. In: Spring Symposium on Ontological Engineering of AAAI, Stanford University, California, pp. 33–40 (1997)

[18] Cimiano, P., Maedche, A., Staab, S., Voelker, J.: Ontology learning, Handbook on ontologies. Springer, Heidelberg (2009)

[19] Gawrysiak, P., Protaziuk, G., Rybiński, H., Delteil, A.: Text onto miner – A semi automated ontology building system. In: An, A., Matwin, S., Raś, Z.W., Ślęzak, D. (eds.) Foundations of Intelligent Systems. LNCS (LNAI), vol. 4994, pp. 563–573. Springer, Heidelberg (2008)

[20] Cunningham, H., Humphreys, K., Gaizauskas, R.J., Wilks, Y.: GATE – A general architecture for text engineering. In: Proceedings of Applied Natural Language Processing (ANLP), pp. 29–30 (1997)
[21] Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer, Heidelberg (2006)
[22] Buitelaar, P., Cimiano, G., Magnini, B.: Ontology Learning from Text: An Overview. In: Ontology learning from text: methods, evaluation and applications. IOS Press (2005)
[23] Protaziuk, G., Kryszkiewicz, M., Rybiński, H., Delteil, A.: Discovering compound and proper nouns. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 505–515. Springer, Heidelberg (2007)
[24] Rybiński, H., Kryszkiewicz, M., Protaziuk, G., Jakubowski, A., Delteil, A.: Discovering Synonyms Based on Frequent Termsets. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 516–525. Springer, Heidelberg (2007)
[25] Rybinski, H., et al.: Text Mining Approach in Ontology Building and Maintenance Methodology – Project for FT, Final Report and Follow Up, WUT, ICS Rep. Warsaw (2007)
[26] Cimiano, P., Völker, J.: Text2Onto. In: Montoyo, A., Muńoz, R., Métais, E. (eds.) NLDB 2005. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005)
[27] Biemann, C.: Ontology Learning from Text – a Survey of Methods. In: LDV-Forum, vol. 20(2) (2005)
[28] Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y.: Data-driven ontology evaluation. In: Proc. of the 4th International Conference on Language Resources and Evaluation, Lisbon (2004)
[29] Völker, J., Vrandečić, D., Sure, Y.: Automatic evaluation of ontologies (AEON). In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 716–731. Springer, Heidelberg (2005)
[30] Haase, P., Völker, J.: Ontology learning and reasoning — dealing with uncertainty and inconsistency. In: da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) URSW 2005 - 2007. LNCS (LNAI), vol. 5327, pp. 366–384. Springer, Heidelberg (2008)
[31] Maedche, A., Volz, R.: The Text-To-Onto ontology extraction and maintenance system. In: Workshop on Integrating Data Mining and Knowledge Management, collocated with the 1st International Conference on Data Mining (2001)
[32] Buitelaar, P., Olejnik, D., Sintek, M.: A Protégé plug-in for ontology extraction from text based on linguistic analysis. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 31–44. Springer, Heidelberg (2004)
[33] Velardi, P., Navigli, R., Cuchiarelli, A., Neri, F.: Evaluation of OntoLearn, a methodology for automatic population of domain ontologies. In: Buitelaar, P., Cimiano, P., Magnini, B. (eds.) Ontology Learning from Text: Methods, Applications and Evaluation, Frontiers in Artificial Intelligence and Applications, vol. 123, pp. 92–106. IOS Press (2005)
[34] Pinto, H.S., Gomez-Perez, A., Martins, J.P.: Some issues on ontology integration. In: Proceedings of the IJCAI-1999 Workshop on Ontologies and Problem-Solving methods (KRR5), Stockholm, Sweden (1999)
[35] INTEROP, Ontology Interoperability, State of the Art Report. WP8ST3 Deliverable (2004)
[36] de Bruijn, J., Ehrig, M., Feier, C., Martin-Recuerda, F., Scharffe, F., Weiten, M.: Ontology Mediation, Merging, and Aligning. John Wiley & Sons, Ltd (2006)

[37] Ehrig, M.: Ontology Alignment Bridging the Semantic Gap. Springer, Heidelberg (2007)
[38] Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. The Knowledge Engineering Review Journal (KER) 18(1), 1–31 (2003)
[39] Stumme, G., Maedche, A.: Ontology Merging for Federated Ontologies on the Semantic Web. In: Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001), Viterbo, Italy (September 2001)
[40] Noy, N.F., Musen, M.: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000), Austin, TX, USA (2000)
[41] Doan, A., Madhaven, J., Domingos, P., Halevy, A.: Ontology matching: A machine learning approach. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies in Information Systems. Springer, Heidelberg (2004)
[42] Calvanese, D., De Giacomo, G., Lenzerini, M.: A framework for ontology integration. In: Proceedings of the 1st Internationally Semantic Web Working Symposium (SWWS), Stanford, CA, USA (2001)
[43] Maedche, A., Motik, B., Silva, N., Volz, R.: MAFRA – A mApping fRAmework for distributed ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 235–250. Springer, Heidelberg (2002)
[44] Kalfoglou, Y., Schorlemmer, M.: IF-Map: an ontology mapping method based on information flow theory. Journal on Data Semantics 1(1) (October 2003)
[45] Barwise, J., Seligman, J.: Information Flow: The Logic of Distributed Systems. Cambridge University Press (1997)
[46] Castano, S., Ferrara, A., Montanelli, S.: Matching Ontologies in Open Networked Systems: Techniques and Applications. In: Spaccapietra, S., Atzeni, P., Chu, W.W., Catarci, T., Sycara, K. (eds.) Journal on Data Semantics V. LNCS, vol. 3870, pp. 25–63. Springer, Heidelberg (2006)
[47] Udrea, O., Getoor, L., Miller, R.J.: Leveraging data and structure in ontology integration. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of data, SIGMOD 2007 (2007)
[48] Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man and Cybernetics (1989)
[49] Cho, M., Kim, H., Kim, P.: A new method for ontology merging based on concept using wordnet. In: The 8th International Conference on Advanced Communication Technology, ICACT 2006, vol. 3 (2006)
[50] Hakimpour, F., Geppert, A.: Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach. In: Proc. of the 2nd Intl. Conf. on Formal Ontology in Information Systems. ACM Press, New York (2001)