Robot Learning of Domain Specific Knowledge from Natural Language Sources

Ines Čeh, Sandi Pohorec, Marjan Mernik and Milan Zorman
University of Maribor
Slovenia

1. Introduction

The belief that problem solving systems would require only processing power was proven false. Actually almost the opposite is true: for even the smallest problems vast amounts of knowledge are necessary. So the key to systems that would aid humans or even replace them in some areas is knowledge. Humans use texts written in natural language as one of the primary knowledge sources. Natural language is by definition ambiguous and therefore less appropriate for machine learning. For machine processing and use the knowledge must be in a formal; machine readable format. Research in recent years has focused on knowledge acquisition and formalization from natural language sources (documents, web pages). The process requires several research areas in order to function and is highly complex. The necessary steps usually are: natural language processing (transformation to plain text, syntactic and semantic analysis), knowledge extraction, knowledge formalization and knowledge representation. The same is valid for learning of domain specific knowledge although the very first activity is the domain definition.

These are the areas that this chapter focuses on; the approaches, methodologies and techniques for learning from natural language sources. Since this topic covers multiple research areas and every area is extensive, we have chosen to segment this chapter into five content segments (excluding introduction, conclusion and references). In the second segment we will define the term *domain* and provide the reader with an overview of domain engineering (domain analysis, domain design and domain implementation). The third segment will present natural language processing. In this segment we provide the user with several levels of natural language analysis and show the process of knowledge acquirement from natural language (NL). Sub segment 3.1 is about theoretical background on syntactic analysis and representational structures. Sub segment 3.2 provides a short summary of semantic analysis as well as current sources for semantic analysis (WordNet, FrameNet). The fourth segment elaborates on knowledge extraction. We define important terms such as *data*, *information* and *knowledge* and discuss on approaches for knowledge acquisition and representation. Segment five is a practical real world (although on a very small scale) scenario on learning from natural language. In this scenario we limit ourselves on a small segment of *health/nutrition* domain as we acquire, process and formalize knowledge on chocolate consumption. Segment six is the conclusion and segment seven provides the references.

2. Domain engineering

Domain engineering (Czarnecki & Eisenecker, 2000) is the process of collecting, organizing and storing the experiences in domain specific system (parts of systems) development. The intent is to build reusable products or tools for the implementation of new systems within the domain. With the reusable products, new systems can be built both in shorter time and with less expense. The products of domain engineering, such as reusable components, domain specific languages (DSL) (Mernik et al., 2005), (Kosar et al., 2008) and application generators, are used in the application engineering (AE). AE is the process of building a particular domain system in which all the reusable products are used. The link between domain and application engineering, which often run in parallel, is shown on Fig. 1. The individual phases are completed in the order that domain engineering takes precedence in every phase. The outcome of every phase of domain engineering is transferred both to the next step of domain engineering and to the appropriate application engineering phase.

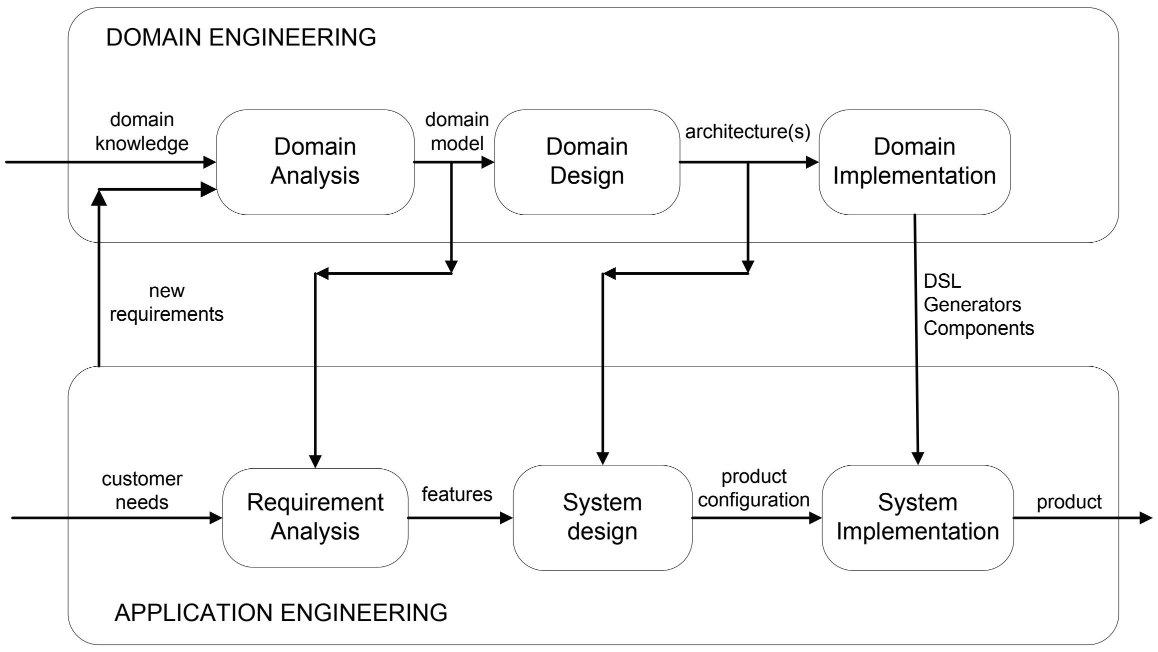


Fig. 1. Software development with domain engineering

The difference between conventional software engineering and domain engineering is quite clear; conventional software engineering focuses on the fulfilment of demands for a particular system while domain engineering develops solutions for the entire family of systems (Czarnecki & Eisenecker, 2000). Conventional software engineering is comprised of the following steps: requirements analysis, system design and the system implementation. Domain engineering steps are: domain analysis, domain design and domain implementation. The individual phases correspond with each other, requirement analysis with domain analysis, system design with domain design and system implementation with domain implementation. On one hand requirement analysis provides requirements for one system, while on the other domain analysis forms reusable configurable requirements for an entire family of systems. System design results in the design of one system while domain design results in a reusable design for a particular class of systems and a production plan. System implementation performs a single system implementation; domain implementation implements reusable components, infrastructure and the production process.

2.1 Concepts of domain engineering

This section will provide a summary of the basic concepts in domain engineering, as summarized by (Czarnecki & Eisenecker, 2000), which are: *domain*, *domain scope*, *relationships between domains*, *problem space*, *solution space* and *specialized methods* of domain engineering. In the literature one finds many definitions of the term *domain*. Czarnecki & Eisenecker defined *domain* as a knowledge area which is scoped to maximize the satisfaction of the requirements of its stakeholders, which includes a set of concepts and a terminology familiar to the stakeholders in the area and which includes the knowledge to build software system (or parts of systems) in the area.

According to the application systems in the domain two separate domain scope types are defined: horizontal (systems category) and a vertical (per system) scope. The former refers to the question how many different systems exist in the domain; the latter refers to the question which parts of these systems are within the domain. The vertical scope is increased according to the sizes of system parts within the domain. The vertical scope determines *vertical* versus *horizontal* and *encapsulated* versus *diffused* paradigms of domains. This is shown on Fig. 2, where each rectangle represents a system and the shaded areas are the system parts within the domain. While vertical domains contain entire systems, the horizontal ones contain only the system parts in the domain scope. Encapsulated domains are horizontal domains, where system parts are well localized with regard to their systems. Diffused domains are also horizontal domains but contain numerous different parts of each system in the domain scope. The scope of the domain is determined in the process of domain scoping. Domain scoping is a subprocess of domain analysis.

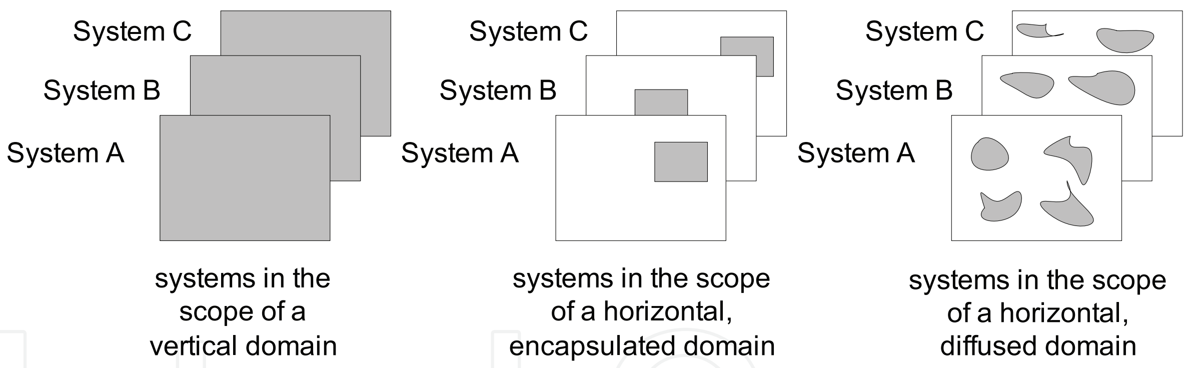

System C
System B
System A
systems in the scope of a vertical domain
System C
System B
System A
systems in the scope of a horizontal, encapsulated domain
System C
System B
System A
systems in the scope of a horizontal, diffused domain

Fig. 2. Vertical, horizontal, encapsulated and diffused domains

Relationships between domains *A* and *B* are of three major types:

- *A* is contained in *B*: All knowledge in domain *A* is also in the domain *B*. We say that *A* is a subdomain of domain *B*.
- *A* uses *B*: Knowledge in domain *A* addresses knowledge in domain *B* in a typical way. For instance it is sensible to represent aspects of domain *A* with terms from the domain *B*. We say that domain *B* is a support domain of domain *A*.
- *A* is analogous to *B*: There are many similarities between *A* and *B*; there is no necessity to express the terms from one domain with the terms from the other. We say that domain *A* is analogous to domain *B*.

A set of valid system specifications in the domain is referred to as the problem space while a set of concrete systems is the solution space. System specifications in the problem space are expressed with the use of numerous DSL, which define domain concepts. The common structure of the solution space is called the target architecture. Its purpose is the definition of a tool for integration of implementation components. One of the domain engineering goals is the production of components, generators and production processes, which automate the mapping between system specifications and concrete systems. Different system types (real-time support, distribution, high availability, tolerance deficiency) demand different (specialized) modelling techniques. This naturally follows in the fact that different domain categories demand different specialized methods of domain engineering.

2.2 Domain engineering process

The domain engineering process is comprised of three phases (Czarnecki & Eisenecker, 2000), (Harsu, 2002): *domain analysis*, *domain design* and *domain implementation*.

Domain analysis

Domain analysis is the activity that, with the use of the properties model, discovers and formalizes common and variable domain properties. The goal of domain analysis is the selection and definition of the domain and the gathering and integration of appropriate domain information to a coherent domain (Czarnecki & Eisenecker, 2000). The result of domain analysis is an explicit representation of knowledge on the domain; the domain model. The use of domain analysis provides the development of configurable requirements and architectures instead of static requirements which result from application engineering (Kang et al., 2004).

Domain analysis includes domain planning (planning of the sources for domain analysis), identification, scoping and domain modelling. These activities are summarized in greater detail in Table 1.

Domain information sources are: existing systems in the domain, user manuals, domain experts, system manuals, textbooks, prototypes, experiments, already defined systems requirements, standards, market studies and others. Regardless of these sources, the process of domain analysis is not solely concerned with acquisition of existing information. A systematic organization of existing knowledge enables and enhances information spreading in a creative manner.

Domain model is an explicit representation of common and variable systems properties in the domain and the dependencies between variable properties (Czarnecki & Eisenecker, 2000). The domain model is comprised (Czarnecki & Eisenecker, 2000) of the following activities:

- *Domain definition* defines domain scope and characterizes its content with examples from existing systems in the domain as well as provides the generic rules about the inclusion or exclusion of generic properties.
- *Domain lexicon* is a domain dictionary that contains definitions of terms related to the domain. Its purpose is to enhance the communication process between developers and impartial persons by simplifying it and making it more precise.
- *Concept models* describe concepts in the domain in an appropriate modelling formalism.
- *Feature models* define a set of reusable and configurable requirements for domain systems specifications. The requirements are called features. The feature model prescribes which property combinations are appropriate for a given domain. It represents the configurability aspect of reusable software systems.

The domain model is intended to serve as a unified source of references in the case of ambiguity, at the problem analysis phase or later during implementation of reusable components, as a data store of distributed knowledge for communication and learning and as a specification for developers of reusable components (Falbo et al., 2002).

Domain Analysis major process components	**Domain analysis activities**
Domain characterization (domain planning and scoping)	*Select domain* Perform business analysis and risk analysis in order to determine which domain meets the business objectives of the organization.
	Domain description Define the boundary and the contents of the domain.
	Data source identification Identify the sources of domain knowledge.
	Inventory preparation Create inventory of data sources.
Data collection (domain modelling)	*Abstract recovery* Recover abstraction
	Knowledge elicitation Elicit knowledge from experts
	Literature review
	Analysis of context and scenarios
Data analysis (domain modelling)	*Identification of entities, operations, and relationships*
	Modularization Use some appropriate modelling technique, e.g. object-oriented analysis or function and data decomposition. Identify design decisions.
	Analysis of similarity Analyze similarities between entities, activities, events, relationship, structures, etc.
	Analysis of variations Analyze variations between entities, activities, events, relationship, structures, etc.
	Analysis of combinations Analyze combinations suggesting typical structural or behavioural patterns.
	Trade-off analysis Analyze trade-offs that suggest possible decompositions of modules and architectures to satisfy incompatible sets of requirements found in the domain.
Taxonomic classification (domain modelling)	*Clustering* Cluster descriptions.
	Abstraction Abstract descriptions.
	Classification Classify description.
	Generalization Generalize descriptions.
	Vocabulary construction
Evaluation	*Evaluate the domain model.*

Table 1. Common Domain Analysis process by Arango (Arango, 1994)

Domain analysis can incorporate different methodologies. These differentiate by the degree of formality in the method, products or information extraction techniques. Most known methodologies are: Domain Analysis and Reuse Environment - DARE (Frakes et al., 1998), Domain-Specific Software Architecture – DSSA (Taylor et al., 1995), Family-Oriented Abstractions, Specification, and Translation - FAST (Weiss & Lai, 1999), Feature Reuse-Driven Software Engineering Business - FeatureRSEB (Griss et al., 1998), Feature-Oriented Domain Analysis - FODA (Kang et al., 1990), Feature-Oriented Reuse Method – FORM (Kang et al., 2004), Ontology-Based Domain Engineering - ODE (Falbo et al., 2002) and Organization Domain Modelling - ODM (Simons & Anthony, 1998).
FODA has proved to be the most appropriate (Alana & Rodriguez, 2007) and we will shortly examine it in the following. FODA is a method of domain analysis that was developed by the Software Engineering Institute (SEI). It is known for its models and feature modelling.
FODA process is comprised of two phases: context analysis and domain modelling. The goal of context analysis is to determine the boundaries (scope) of the analyzed domain and the goal of domain modelling is to develop a domain model (Czarnecki & Eisenecker, 2000). FODA domain modelling phase in comprised of the following steps (Czarnecki & Eisenecker, 2000):

- *Information analysis* with the main goal of retrieving domain knowledge in the form of domain entities and links between them. Modelling techniques used in this phase can be in the form of semantic networks, entity-relationship modelling or object oriented modelling. The result of information analysis is an information model that matches the conceptual model.
- *Features analysis* covers application capabilities in the domain as viewed by the project contractor and the final user. Common and variable features that apply to the family of systems are simply called features. They are contained in the features model.
- *Operational analysis* results in the operational model. It shows how the application works and covers the relations between objects in the informational model and the features in the feature model.

An important product from this phase is the domain dictionary that defines the terminology used in the domain (along with the definitions of domain concepts and properties). As we mentioned FODA methodology is known by its feature modelling. Properties can be defined as the system characteristics visible to the end user (Harsu, 2002).They are categorized into: mandatory, alternative and optional. They are visualized on a feature diagram, which is a key element of the domain model. The feature concept is further explained and presented in (Czarnecki & Eisenecker, 2000).

Domain design

Domain design takes the domain model built in the process of domain analysis and tries to create a general architecture that all the domain elements are compliant with (Czarnecki & Eisenecker, 2000). Domain design focuses on the domain space for solution planning (solution space). It takes the configuration requirements, developed in the domain analysis phase and produces a standardized solution for a family of systems that is configurable. Domain design tries to create architectural patterns that try to solve common problem for the family of systems in the domain, despite different configuration requirements (Czarnecki & Eisenecker, 2000). Beside the pattern development the engineers have to carefully determine the scope of individual pattern and the level of context at which the pattern applies. Scope limitation is crucial: too much context is reflected in a pattern that is not acceptable to many systems, too little context on the other hand is reflected in a pattern that lacks capability to an extent that it is not usable. A usable pattern has to be applicable to many systems and of high quality (Buschmann et al., 2007).

Domain implementation

Domain implementation covers the implementation of the architecture, components and tools designed in the phase of domain design.

3. Natural language processing

The term natural language is used to describe human languages (English, German, Chinese, …). Processing these languages includes both written and spoken language (text and speech). In general the term refers to processing written language since the speech processing has evolved into a separate field of research. According to the term this segment will focus on the written language. To implement a program to acquire knowledge from natural language requires that we transform the language to a formal language (format) that is machine readable. In order to perform such a task it is vital that we are able to understand the natural language. Major problems with language understanding are that a large amount of knowledge is required to understand the meaning of complex passages. Because of this the first programs to understand natural language were limited to a minute environment. One of the earliest was Terry Winograd's SHRDLU (Winograd, 1972) which was able to formulate conversations about a blocks world. In order to work on a larger scale where a practical application is possible language analysis is required. Generally speaking there are several levels of natural language analysis (Luger, 2005):

- *Prosody* analyses rhythm and intonation. Difficult to formalize, important for poetry, religious chants, children wordplay and babbling of infants.
- *Phonology* examines sounds that are combined to form language. Important for speech recognition and generation.
- *Morphology* examines word components (morphemes) including rules for word formation (for example: prefixes and suffixes which modify word meaning). Morphology determines the role of a word in a sentence by its tense, number and part-of-speech (POS).
- *Syntax* analysis studies the rules that are required for the forming of valid sentences.
- *Semantics* studies the meaning of words and sentences and the means of conveying the meaning.
- *Pragmatics* studies ways of language use and its effects on the listeners.

When considering means to acquire knowledge from natural language sources the analysis is a three step process: syntactic analysis, meaning analysis (semantic interpretation; generally in two phases) and the forming of the final structure that represents the meaning of the text. The process is shown on Fig. 3. In the next sections (3.1 and 3.2) we will provide an overview of the syntactic and semantic analysis while the practical overview with resources and approaches specific to the learning of domain specific knowledge will be discussed in the following sections.

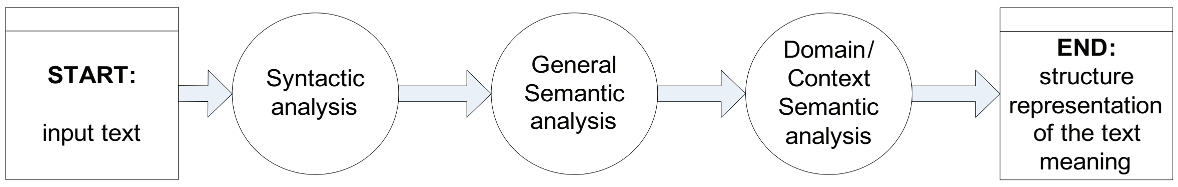
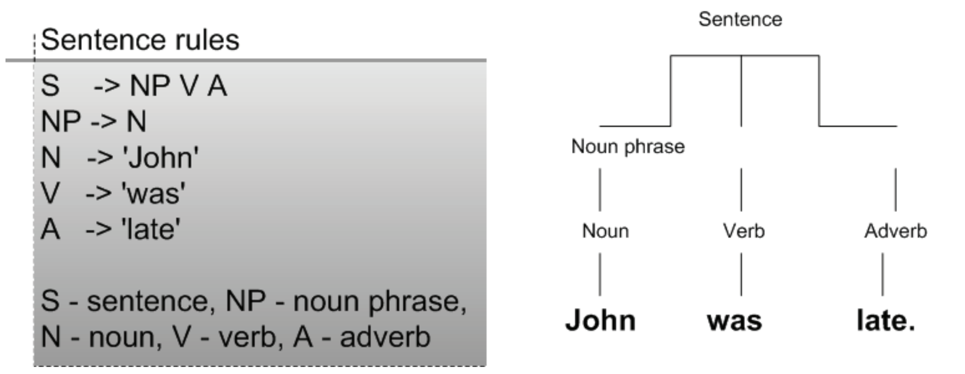

START:
input text
Syntactic analysis
General Semantic analysis
Domain/ Context Semantic analysis
END: structure representation of the text meaning

Fig. 3. Natural language analysis process

3.1 Theoretical overview of the syntactic analysis and representational structures

The goal of syntactic analysis is to produce the *parse tree*. The parse tree is a breakdown of natural language (mostly on the level of sentences) to their syntactic structure. It identifies the linguistic relations. Syntactic analysis can be achieved with context-free or context sensitive grammars. The theoretical background for context-free grammars was outlined by Partee et al., 1993. An example of a system built on context-free grammars is presented in Alshawi, 1992. Perhaps the simplest implementation of a context-free grammar is the use of production (rewrite) rules with a series of rules with terminals (words from natural language) and non terminals (linguistic concepts: noun phrase, verb, sentence...). An example of the parse three with the rewrite rules is shown on Fig. 4.
An alternative approach is in the form of transition network parsers which although they themselves are not sufficient for natural language they do form the basis for augmented transition networks (Woods, 1970).



Sentence rules

```
S   -> NP V A
NP -> N
N   -> 'John'
V   -> 'was'
A   -> 'late'

S - sentence, NP - noun phrase,
N - noun, V - verb, A - adverb
```

Fig. 4. Small set of rewrite rules and the result of syntax analysis, a parse tree

The shortcoming of context-free grammars is evident in the name itself; they lack the context that is necessary for proper sentence analysis. Although they can be extended to take context into consideration a more native approach to the problems seems to be the use of grammars that are natively context aware. The main difference between the context-free (CF) and context-sensitive (CS) grammars is that the CS grammars allow more than one symbol on the left side of a rule and enable the definition of a context in which that rule can be applied.
CS grammars are on a higher level on the Chomsky grammar hierarchy (Chomsky, 1956) presented on Fig. 5 than CF grammars and they are able to represent some language structures that the CF grammars are unable but they do have several significant shortcomings (Luger, 2005):

- they dramatically increase the number of rules and non-terminals,
- they obscure the phrase structure of the language,
- the more complicated semantic consistency checks lose the separation of syntactic and semantic components of the language and
- they do not address the problem of building a semantic representation of the text meaning.

It appears that despite their native context awareness CS grammars have proven to be too complicated and they are usable only for the validation of sentence structure. For the purpose of acquiring knowledge from natural language sources they are not usable at all since they do not address the building of a semantic representation of the text meaning.
Various researchers have focused on enhancing context-free grammars. A new class of grammars emerged; the augmented context-free (ACF) grammars. The approach replaces

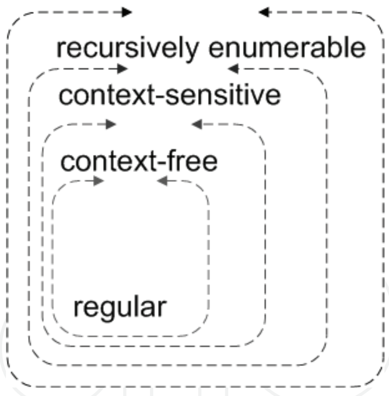


Fig. 5. Chomsky grammar hierarchy

the usage of the grammar to describe the number, tense and person. These terms become features attached to terminals and nonterminals. Most important types of ACF grammars are augmented phase structure grammars (Heidorn, 1975), augmentations of logic grammars (Allen, 1987) and augmented transition networks.

3.2 Semantic analysis

We have tried to give a broad overview of the complexity of syntactic analysis of natural language in the previous section because syntactic analysis is tightly coupled with semantic analysis. Semantic analysis tries to determine the meaning at the level of various language structures (words, sentences, passages). In other words semantic analysis is the process in which words are assigned their *sense*. Semantic analysis is a component of a large number of scientific research areas (Sheth et al., 2005): Information retrieval, Information Extraction, Computational Linguistics, Knowledge Representation, Artificial Intelligence and Data Management. Since the research areas are very different each has a very different definition of cognition, concepts and meaning (Hjorland, 1998). Sheth et al. organized the different views to three forms of semantics: *Implicit*, *Formal* and *Powerful* (Soft). Techniques based on the analysis of unstructured texts and document repositories with loosely defined and less formal structure in the fields of Information Retrieval, Information Extraction and Computational Linguistics have data sources of the *implicit* type.
Knowledge Representation, Artificial Intelligence and Data Management fields have a more formal data form. Information and knowledge is presented in the form of well defined syntactic structures (and rules by which the structures can be combined to represent the meaning of complex syntactic structures) with definite semantic interpretations associated (Sheth et al., 2005). According to the aforementioned properties these fields rely on Formal Semantics. Semantic web in the future will be annotated with knowledge from different sources so it is important that the systems would be able to deal with inconsistencies. They should also be able to increase the expressiveness of formalisms. These are the features that would require *soft* (powerful) semantics (Sheth et al.)
The datasets that contain meanings of words are called sense sets. The first sense set, “WordNet®”was created at Princeton (Miller, 1995). WordNet is a lexical database that contains nouns, verbs, adjectives and adverbs of the English language. Words are grouped into sets of cognitive synonyms (sysnets). Each sysnet expresses a concept. Interlinked sysnets form a network of related words and concepts. The network is organized in hierarchies which are defined by either a *generalization* or *specialization*. An example of a WordNet hierarchy is presented on Fig. 6.

A global repository of wordnets in languages other than English (more than fifty are available) is available on the Global WordNet Association webpage (http://www.globalwordnet.org/).
Similar project is being conducted at Berkeley University; their FrameNet (http://framenet.icsi.berkeley.edu/ is based on frame semantics. Frame semantics is a development of case grammar. Essentially to understand a word one has to know all essential knowledge that relates to that word. A semantic frame is a structure of concepts interconnected in such a way that without knowing them all one lacks knowledge of anyone in particular. A frame describes a situation or an event. Currently FrameNet contains more than 11.600 lexical units (6800 fully annotated) in more than 960 semantic frames.

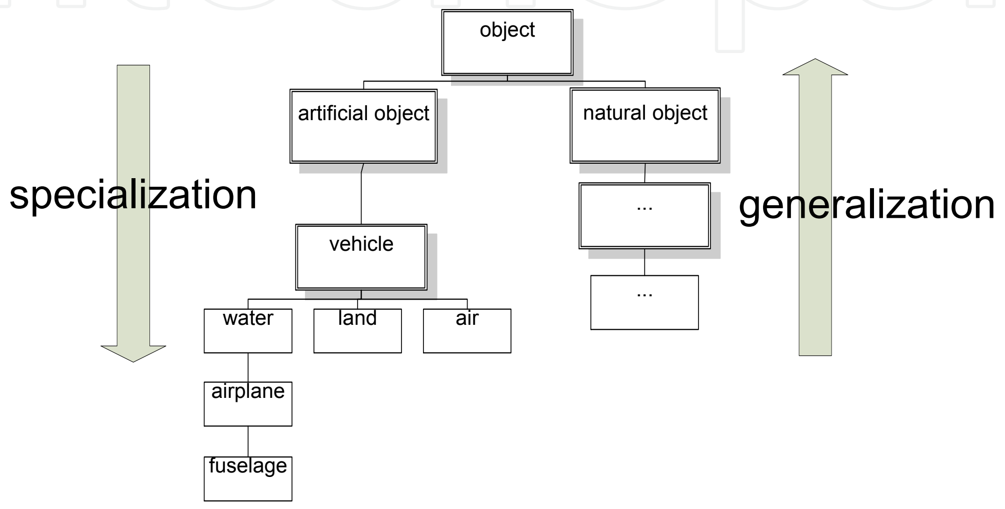


Fig. 6. Example of WordNet network of interlinked sysnets in the form of a directed acyclic graph

4. Knowledge extraction

Knowledge extraction is a long standing goal of research in the areas within artificial intelligence (Krishnamoorthy & Rajeev, 1996), (Russell & Norvig, 2003). Numerous sources, such as philosophy, psychology, linguistics, have contributed to a wealth of ideas and a solid foundation in applied science. In the early years there was a general consensus that machines will be able to solve any problem as efficiently as the world foremost experts. Scientist believed that there is the theoretical possibility of creating a machine designed for problem solving that could take on any problem with a minimum amount of information. The machine would use its enormous computing power to solve the problem. Only when the work began on building such a machine, everyone realized that the solution to problem solving lies in knowledge, not computing power. Actual machines require excessive amount of knowledge to perform even the most basic intelligent tasks. The knowledge must be in a structural form so that it can be stored and used when necessary. These machines are known as *expert systems*. Actually they are knowledge based systems (KBS) (Kendal & Creen, 2007); expert systems are just a part of knowledge based systems.
Knowledge engineers quickly realized that acquisition of quality knowledge appropriate for quality and robust systems is a time consuming activity. Consequentially knowledge acquisition was designated the bottleneck of expert system implementation. Because of this knowledge acquisition became the primary research area of knowledge engineering (Kendal

& Creen, 2007). Knowledge engineering is the process of developing knowledge systems in any field, public or private sector, sales or industry (Debenham, 1989).
In knowledge engineering it is essential that one understands these terms: *data*, *information* and *knowledge*. Their hierarchy is presented on Fig. 7.
Before we can begin to understand "knowledge" we must first understand the terms data and information. Literature provides many definitions of these terms (Kendal & Green, 2007), (Zins, 2007). Their meaning becomes clear only when we look for the differences between them. There exist no universal definition of *data* or *information* and no definition is applicable in all situations. Data becomes information when their creator supplements them with an added value. Different ways in which this can occur is presented in (Valente, 2004).
When we examine some of the definitions of knowledge, such as:"knowledge is the result of information understanding" (Hayes, 1992), "knowledge is information with context and value that make it usable" (Gandon, 2000), it becomes clear that knowledge is something that one has after he understands information.
So as information derives from knowledge, knowledge also derives from information. During the derivation one of the following transformations (Gandon, 2000) takes place: comparison (how can the information on this situation be compared to another familiar situation), consequences (what consequences does the information have on decisions and courses of action), connections (how this part of knowledge connects to other parts) and conversation (what is the people's opinion on this information). It should be clear that data, information and knowledge are not static by nature; they are the stages in a process of applying data and its transformation to knowledge. From the knowledge engineering standpoint it is positive to handle knowledge as something that is expressed as a rule or is usable for decision support. For instance: "IF it rains, THEN use an umbrella". The value of data increases as it is transformed into knowledge as knowledge enables the making of useful decisions. Knowledge can be regressed to information and data. Davenport and Prusak (Davenport & Prusak, 1998) called this process „de-knowledging". It occurs when there is too much knowledge and the knowledge worker can no longer grasp the sense of it.

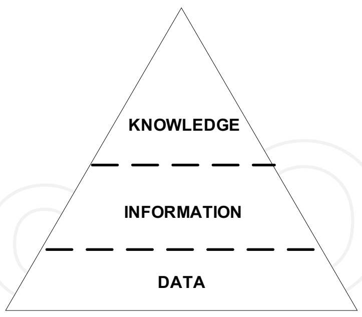


Fig. 7. Data, information and knowledge

Knowledge engineer usually works with three types of knowledge: declarative, procedural and meta-knowledge. Declarative knowledge describes the objects (facts and rules), that are in the experts systems scope, and the relations between them. Procedural knowledge provides alternative actions, which are based on the use of facts for knowledge acquirement. Meta-knowledge is knowledge about knowledge that helps us understand how experts use knowledge for their decisions.
Knowledge engineers have to be able to distinguish between these three knowledge types and to understand how to encode different knowledge types into a specific form of knowledge based systems.

Knowledge based systems (KBS) are computer programs that are intended to mimic the work of specific knowledge areas experts (Kendal & Creen 2007). They incorporate vast amounts of knowledge, rules and mechanisms in order to successfully solve problems. The main types of knowledge systems are: expert systems, neural networks, case-based reasoning, genetic algorithms, intelligent agents, data mining and intelligent tutoring systems. These types are presented in detail in (Kendal & Creen, 2007). KBS can be used on many tasks, which were once in the domain in humans. Compared to human counterparts they have some advantages as well as disadvantages. For example while human knowledge is expensive, KBS are relatively cheap. On the other side humans have a wider focus and understanding; KBS are limited to a particular problem and cannot be applied on other domains.
Since mid eighties knowledge engineers have developed several principles, methods and tools for the improvement of the knowledge acquirement process. These principles cover the use of knowledge engineering on actual world problems. Some key principles that are discussed in detail in (Shadbolt & Milton, 1999), are that different types of knowledge, experts (expertise), knowledge representation and knowledge usage exist and that structural methods should be used in order to increase efficiency.
Development of knowledge based applications is difficult for the knowledge engineers. Knowledge based projects cannot be handled with the techniques for software engineering. Life-cycle of knowledge based application and software applications are different in several aspects. With the intent to achieve impartiality in knowledge engineering the life-cycle of applications focuses on the six critical phases presented on Fig. 8.

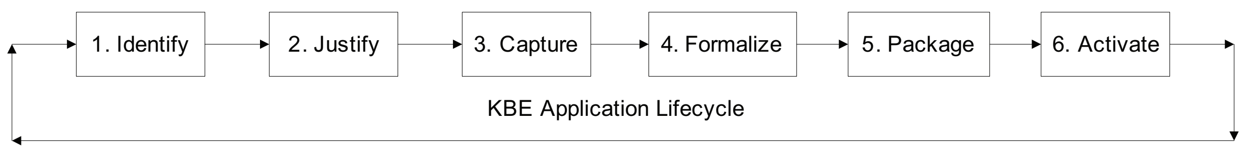


Fig. 8. Knowledge based engineering application lifecycle

According to the specifics of each principle element, numerous knowledge engineering techniques have been developed. Well known, used in many projects, techniques are: Methodology and tools Oriented to Knowledge-Based Engineering Applications - MOKA (Stokes, 2001), Structured Process Elicitation Demonstrations Environment - SPEDE (Pradorn, 2007) and Common Knowledge Acquisition and Design System - CammonKADS (Schreiber et al., 1999).
Knowledge acquisition is difficult, both for humans and machines. Phrase "knowledge acquisition" generally refers to gathering knowledge from knowledge rich sources and the appropriate classification to a knowledge base. As well as this it also refers to improving knowledge in existing knowledge bases. The process of knowledge acquisition can be manual or automatic. In the manual mode the knowledge engineer receives knowledge from one or more domain experts. In automatic mode a machine learning system is used for autonomous learning and improvement of real world knowledge.
Manual knowledge acquirement is difficult because of two reasons. First, the knowledge engineer has to maintain contact with the experts for a substantial amount of time, in some cases several years. And second because in some cases the experts cannot formally express their knowledge. These problems can be avoided with autonomous knowledge encoding with the use of machine learning. The approach is presented on Fig. 9. The database is

formed with the use of experts and various reasoning/inference systems. Machine learning uses the data in the database to infer new knowledge. This newly found knowledge is then transformed into a knowledge base.
Knowledge representation is concerned with researching knowledge formalization and machine processing. Automation inference techniques enable computer system to infer conclusions from machine readable knowledge. It is the goal of knowledge representation and inference to plan computer systems, that would, similar as humans, infer on machine interpretable representations of the real world.
An overview of state of the art technologies of the semantic web and the use cases clearly show that knowledge representation is in many different formats. Most widely used representations are semantic networks, rules and logic. We continue with a short examination of these representations, a more detailed presentation can be found in (Grimm et al., 2007).

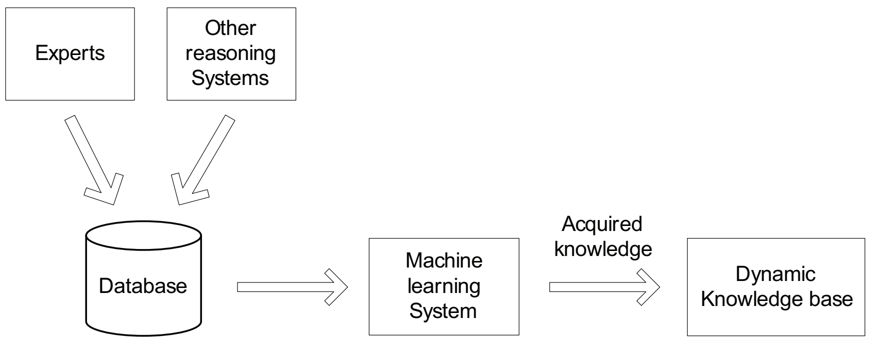


Fig. 9. Principles of automated knowledge acquisition

The term *semantic network* encompasses a family of graph-based representations which share a common set of assumptions and concerns. A visual representation is that of a graph where node connections are the relations between concepts. Nodes and connections are labelled to provide the necessary information about the concept and type of association. Fig. 10 shows an example of a semantic network for a domain of animals. The concepts are represented by ellipses and the connections are arrows. The network is a representation of this natural language passage:
„Mammals and reptiles are animals. Dogs and dolphins are mammals, snakes and crocodiles are reptiles. Snakes are reptiles without legs; crocodiles are reptiles with four legs. While dolphins live in the sea, the dogs live on land. Dogs have four legs. Labrador retriever is a medium sized dog." The nouns in this text refer to concepts; the verbs are links between them.
Newer models of a network representation language are conceptual graphs. A conceptual graph is (Luger 2005) a finite, connected, bipartite graph. The nodes in the graph are either concepts or conceptual relations. Conceptual graphs do not use labeled arcs; instead the conceptual relation nodes represent relations between concepts. Because conceptual graphs are bipartite, concepts only have arcs to relations, and vice versa. Example shown on Fig. 11 a) represents a simple proposition "Cat's color is white". A more complex graph on Fig. 11 b) represents the sentence "John bought Joan a large ice cream" indicates how conceptual graphs are used to formalize natural language.
Semantic networks are especially appropriate for the extraction of taxonomic structures or domain objects categories and for the expression of general sentences on the domain of

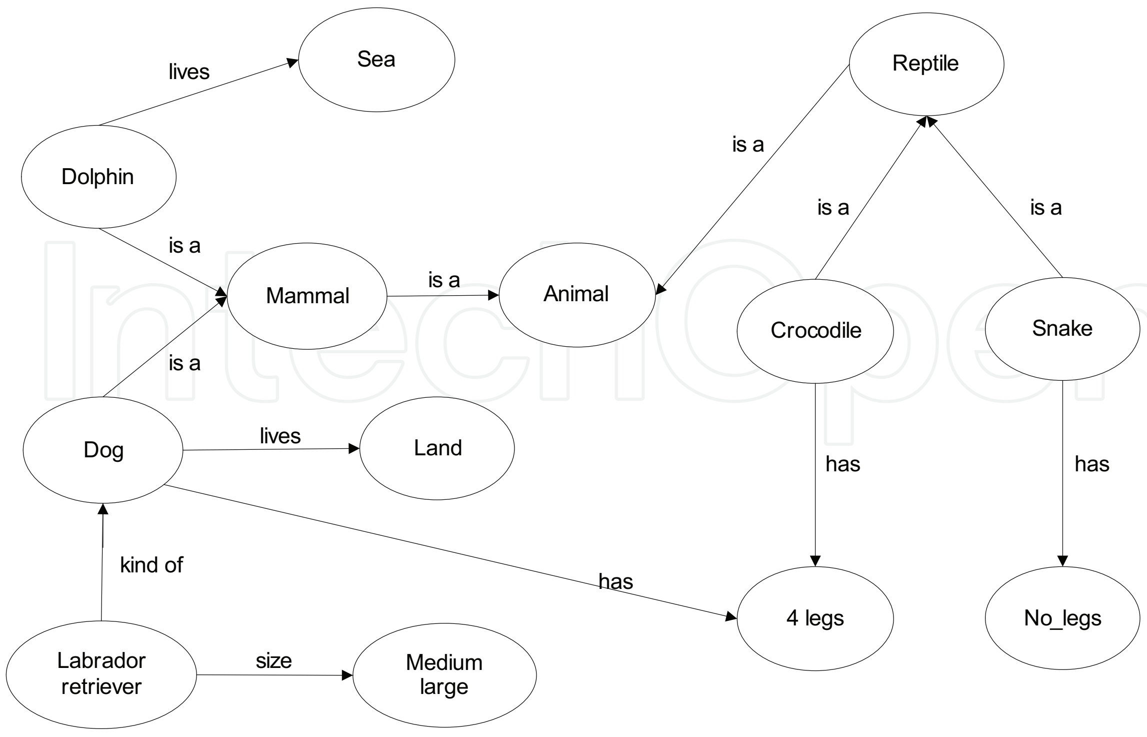


Fig. 10. Semantic net: animals

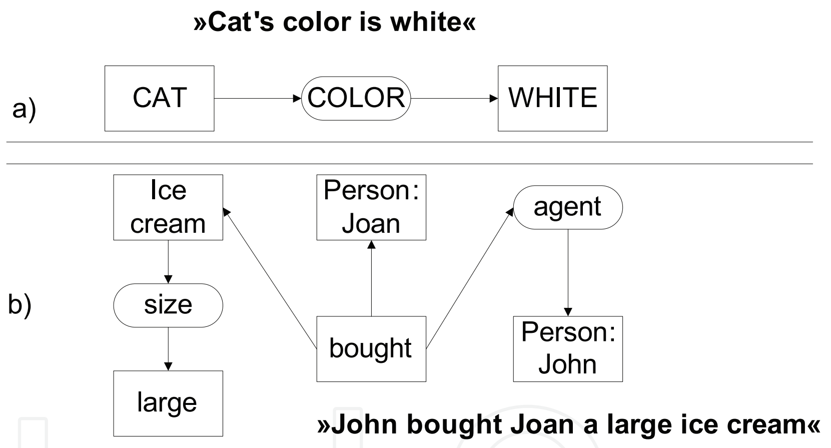


Fig. 11. Conceptual graphs of "Cat's colour is white" and "John bought Joan a large ice cream"

interest. Inheritance and other relations between these kinds of categories can be represented and inferred from the hierarchies the network naturally contains. Individual representatives or even data values like numbers or strings are not compatible with the idea of semantic networks (Grimm, 2007).

Another natural form of knowledge expression is expression with *rules* that mimic the principle of consequence. They are in the form of IF-THEN constructs and support the expression of various complex sentences.

Rules are found in logic programming systems, such as the well known programming language Prolog (Sterling & Shapiro, 1994), deductive data bases (Minker, 1987) or business rules systems (Grimm, 2007). „IF" part of the rule is the body, while the„THEN" part is the head of the rule. An example of a rule that refers to Fig. 10 is:

IF something is a Labrador retriever **THEN** it is also a dog.

Because rules in natural language are not appropriate for machine processing these kinds of phrases are formalized with the use of predicates and object variables in the domain of interest. Formalized, the example above would be written like this:

Dogs(?t) :- Labrador retriever (?t).

In most logical programming languages the rule is read as an inverse implication, which starts with the head followed by the body. It is identified with the „:-" symbol that is a synonym for the reversed arrow (Grimm, 2007).

Afore mentioned forms, semantic networks and rules, are formalized with logic that provides the exact semantics. Without that kind of precise formalization they would be ambiguous and consequently not appropriate for machine processing. The most featured and fundamental logical formalism that is typically used for knowledge representation is the first order logic (Gašević et al., 2006). First order logic provides means to describe the domain of interest as a composition of objects and the construction of logical formulas around those objects. The objects are formed with the use of predicates, functions, variables and logic connectives.

Similar to semantic networks, most natural language sentences can be expressed with terms from logic sentences about the objects in the target domain with the appropriate choice of predicates and function symbols (Grimm, 2007).

Axiomatising parts of the semantic network on Fig. 10 will be used to demonstrate the use of logic for knowledge representation. For instance, subsumption on Fig. 12 can be directly expressed with logical implication which is formulated in (1):

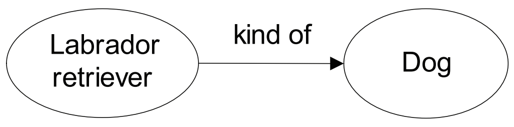


Fig. 12. Example of subsumption

$$\forall x : (Labrador\_retriever(x) \rightarrow Dogs(x)) \quad (1)$$

Logic can also be used to represent rules. IF-THEN rules can be expressed as a logical implication with universal quantity variables. For instance the basic formalization of the rule: „IF something is a Labrador retriever THEN it is also a dog" is also translated to the same logic formula (1).

5. Practical case of learning from natural language

This section will focus on a chronological sequence of learning from natural text. It will present a short example on how to use the aforementioned methodologies, approaches and techniques on a small example. We have to stress that this is only a short practical example and so the approaches chosen in it are somewhat simplified to allow for better understanding of the example. The example is not intended to be a cookbook for learning from natural language; it is merely used to present the user with a real world scenario with a chronological sequence of the steps and actions necessary for the incorporation of natural language knowledge in a formalized fashion. We will use the health/nutrition domain and we will be focusing on the consumption of chocolate and its influence on patients. The scenario will outline the entire process of learning in the proper order. The sequence of actions is the following:

- Definition of the target domain.
- Acquisition of natural language resources and pre-processing.
- Knowledge extraction and formalization.

5.1 Domain definition

The first step cannot be automated. A knowledge engineer has to determine what the boundaries of the target domain are. He accomplishes this with an extensive examination of the target domain in order to familiarize himself with the domain concepts. Almost exclusively a knowledge engineer is someone with background in computer science and with the exception that the target domain falls within his area of expertise he usually has only the most basic understanding of the target domain. To be able to successfully incorporate his skills in the entire process it is vital that he understands the target domain at least to a certain degree. This first step is concluded when the domain is firmly, formally defined in such a way that there are no ambiguities on the question what falls inside the domain and what lies on the outside. In our example a short definition of the domain would be the following: The main two entities in the domain are chocolate and patients. The domain scope will be limited to milk, sweet and dark chocolate only. The goal of the learning will be the positive and negative effect of chocolate consumption with regard to quantity. Fig. 13 shows a simplified model of the domain. Additionally the source of learning will be the news reporting on studies being done by the research community. The news selection policy will be based on top level breadth-first policy; if the title contains a keyword from the domain the news is included in the knowledge base.

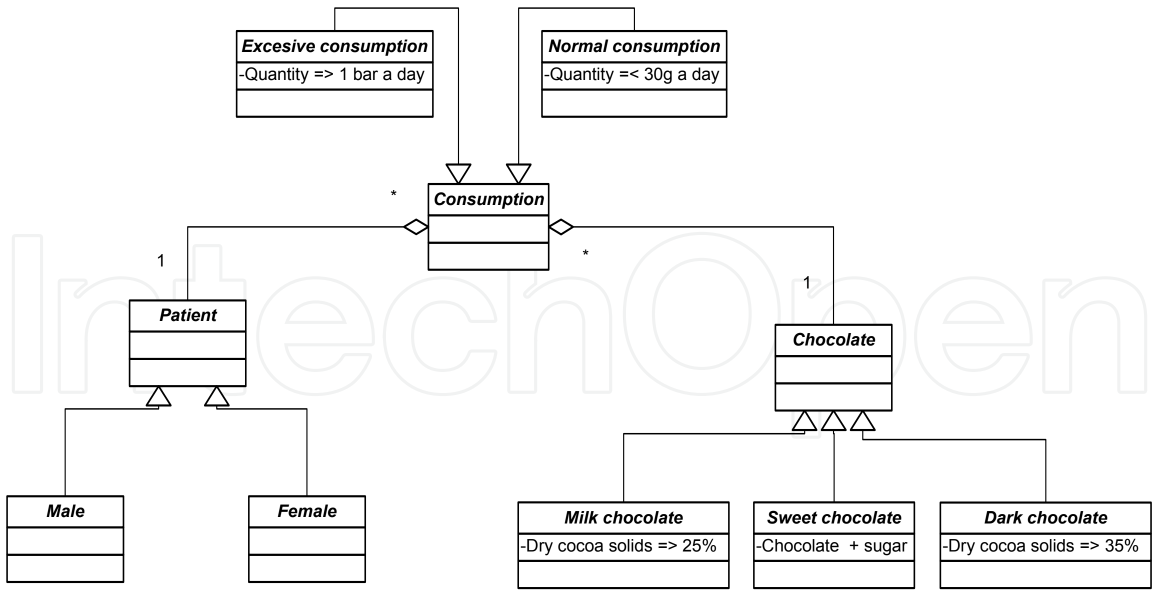


Fig. 13. Domain model for chocolate consumption

5.2 Acquisition of natural language resources and pre-processing

In this step the knowledge engineer determines which natural language sources are available to him and which can be added during the process. He has to determine the source

format (documents, web pages...) and the necessary processing for the transformation to the final format. For the example which we are providing we have chosen reports on research projects and studies as the primary source of data. The data will be acquired by RSS feeds from selected health websites. A snippet of the XML file compliant with the RSS 2.0 specification is shown on Fig. 14. News, published by CNN, titled "*Daily chocolate may keep the heart doctor away*" is selected because it contains keyword from the domain model ("chocolate"). On a similar principle other news are selected and inserted to a relational database for local storage and queued for further processing. The collection of domain documents can be enhanced with periodical download of the RSS feeds.
Pre-processing of the selected news is comprised from:

- the download of the full news,
- transformation to plaintext (stripping of HTML tags) and
- sentence level tokenization.

The download of the full news is necessary because the RSS feeds contain only short content (title, short description...) and the target web address of the news.

```
<item>
        <title>Sleep Apnea Linked to Eyelid Disorder</title>
        <link>http://news.health.com/2010/04/02/sleep-apnea-linked-eyelidisorder/</link>
        <pubDate>Sat, 03 Apr 2010 02:08:37 +0000</pubDate>
             ...
</item>
<item>
        <title>Daily chocolate may keep the heart doctor away</title>
        <link>http://rss.cnn.com/~r/rss/cnn_latest/~3/E-6aM9zPrqE/index.html/</link>
        <pubDate>Sat, 03 Apr 2010 02:08:37 +0000</pubDate>
             ...
</item>
```

Fig. 14. Snippet from the news feed

5.3 Knowledge extraction and formalization

The first step in knowledge extraction is the part-of-speech (POS) analysis. It can be performed with the use of existing POS taggers, for example the TnT (Brants, 2000) or TreeTagger (Schmid, 1994). The latter achieved 96% accuracy on the Penn-Treebank data. In the example we are following, the news has been fully downloaded, the text transformed to plaintext and tokenized to individual sentences. The sentences to be used can be classified by a simple TFIDF (Term Frequency Inverse Document Frequency) metric. For our purposes the documents in the formula are sentences. The metric is defined as follows:

$$d^{(i)} = TF(W_i, d)IDF(W_i) \tag{2}$$

The IDF is defined:

$$IDF(W_i) = \log \frac{D}{DF(W_i)} \tag{3}$$

D is the number of documents, DF(W) is the number of documents in which the word (W) occurs at least once and TF(W, d) is the number of word W occurrences in the document d.

Additionally the metric can be normalized so that the TFIDF of individual words is divided by the square root of the sum of all TFIDF word frequencies as follows:

$$nTFIDF = \frac{TFIDF_{i,j}}{\sqrt{\sum_i TFIDF_{i,j}^2}} \tag{4}$$

The very first sentence provides useful knowledge and we will follow the example on this sentence. It is stated as:
"Eating as little as a quarter of an ounce of chocolate each day may lower your risk of experiencing heart attack or stroke!". The POS analysis provides the tags listed in.
This is processed by semantic interpretation that uses existing domain knowledge (defined in the domain definition phase) to produce a representation of the meaning. Fig. 15 shows an internal representation in the form of a conceptual graph. Semantic interpretation uses both the knowledge about word meanings (within the domain) and linguistic structure.

Word	Tag	Word	Tag	Word	Tag
Eating	VBG	*as*	RB	*little*	JJ
as	IN	*a*	DT	*quarter*	NN
of	IN	*an*	DT	*ounce*	NN
of	IN	*chocolate*	NN	*each*	DT
day	NN	*may*	MD	*lower*	VB
your	PRP$	*risk*	NN	*of*	IN
experiencing	VBG	*a*	DT	*heart*	NN
attack	NN	*or*	CC	*stroke*	VB
Legend: IN - Preposition or subordinating conjunction, JJ - Adjective, MD - Modal, NN - Noun, singular or mass, PRP$ - Possessive pronoun, RB - Adverb, VB - Verb, base form, VBG - Verb, gerund or present participle					

Table 2. POS tags of a news sentence

The sentence is separated into two distinct categories: cause (IF) and effect (THEN). Both are associated with the object. In the figure the application used knowledge that *ounce* is a unit of amount, *day* is a unit of time and that a person normally eats chocolate not the other way around. So combining this knowledge produced the resulting representation of knowledge in the sentence. The agent (the one that influences) is *chocolate*, the object (the recipient of the action) is the word *your* and the action (agent to object) is *eating*. Combining that *to eat* is associated with the domain concept of amount and that *ounce* is a unit of amount the application can effectively reason that the meaning of the cause part (Fig. 15 segment A) of the sentence is: object that *eats* a 0.25 *ounce* of *chocolate* in a period of *one day.* The effect side (Fig. 15 segment C) has the meaning of: the object experiences the influence of *reduced* possibility of a disease of type *heart attack/stroke*. This internal representation is then generalized with the addition of known concepts. The object *yours* is a possessive pronoun and therefore is mapped to a person which is marked as *"patient„* in the domain.
The amount of quarter of an ounce is mapped to the primary unit for amount in the domain, (grams) with the use of a conversion factor. So ¼ of an ounce becomes 7.08738078 grams. The resulting semantic net with the additional information is the final interpretation of the domain specific world knowledge learned from this sentence. These representations can

transform to a rule base which can then be automatically evaluated and used by the final application (the one that uses the knowledge learned).
For the example we have been following a rule would be in the following form:

```
RULE    chocolate consumption influence
IF      typeof (object) IS patient
AND     typeof (action) IS eat
AND     action::target IS chocolate
AND     quantityof (action) IS 7g
AND     timespan (action) IS 24h
THEN    typeof(consequence) IS influence
AND     consequence::target IS disease
AND     typeof(disease) IS heart attack/stroke
AND     relationship (consequence, consequence::target) IS reduced risk
```

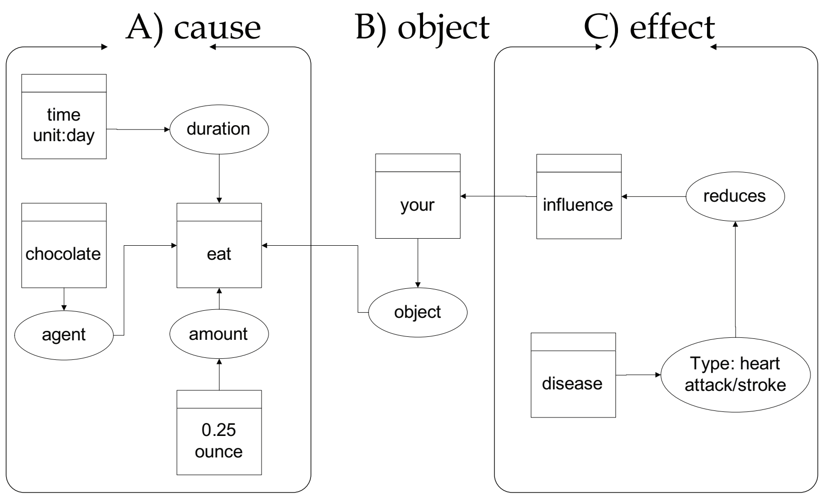


Fig. 15. Internal representation of the meaning of the sentence

This is the final formalization of acquired knowledge. In this form the knowledge is fully machine readable, providing there are inferring rules that define how to evaluate the value entities (typeof, quantityof,…). This format can be stored and used as need arises.

6. Conclusion

We have presented the major research areas that are vital to learning domain specific knowledge. The practical example shows the chronological sequence of the learning process. We have shown that it is vital to formally define the target domain. Also in order for the knowledge engineers to effectively determine the domain and evaluate the progress of the project they have to have a more than superficial knowledge of the domain. Incorporation of existing knowledge (dictionaries, semantic annotations etc.) is very important since every task is very time consuming and repeating existing work is not efficient.
Knowledge extraction should have a much higher success rate if it is done on smaller documents. It is for this reason that the practical example uses news feeds. Their content is already summarized in the form of the short description. The full text of the news can then be used to provide facts that show how and why the summary is correct. So in the example we are counting on the title and short description to provide the new facts while the news

body is used as the supporting information. This provides an efficient example of knowledge learning from natural language.

7. References

Alana, E. & Rodriguez, A. I. (2007). Domain Engineering Methodologies Survey, available on *http://www.pnp-software.com/cordet/download/pdf/GMV-CORDET-RP-001_Iss1.pdf*

Allen, J. (1994). *Natural Language Understanding (2nd Edition),* Addison Wesley, ISBN-10: 0805303340

Alshawi, H. (1992). *The Core Language Engine,* The MIT Press, ISBN-10: 0262011263, USA

Arango, G. (1994). Domain Analysis Methods. In *Software Reusability,* Schafer, W.; Prieto-Díaz, R. & M. Matsumoto (Ed.), page numbers (17-49), Ellis Horwood

Brants, T. (2000). TnT – A Statistical Part-of-Speech Tagger, *Proceedings of the sixth conference on Applied Natural Language Processing,* pp. 224-231, ISBN-10: 1558607048, Seattle, Washington, April – May 2000, Morristown, NJ, USA

Buschmann, F.; Henney, K. & Schmidt, D. C. (2007). *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages*, John Wiley & Sons, ISBN-10: 0471486480, England

Chomsky, N. (1956). Three Models for the Description of Language, *IRE Transactions on Information Theory*, Vol. 2, page numbers (113-124

Czarnecki, K. & Eisenecker, U. (2000). *Generative Programming: Methods, Tools andApplications*, ACM Press/Addison-Wesley Publishing Co., ISBN-10: 0201309777, New York, NY, USA

Davenport, T. H. & Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know,* Harvard Bussiness Press, ISBN-10: 0875846556, United States of America

Debenham, J. K. (1989). *Knowledge systems design*, Prentice Hall, ISBN-10: 0135164281

Falbo, R; Guizzardi, G & Duarte, K. C. (2002). An ontological approach to domain engineering, *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pp. 351-358, ISBN-10: 1581135564, Ischia, Italy, July 2002, ACM, New York, NY, USA

Frakes, W.; Prieto-Diaz, R. & Fox, C. (1998). DARE: Domain analysis and reuse environmrent, *Annals of Software Engineering*, Vol. 5, No. 1, (January 1998) page numbers (125-141), ISSN: 1573-7489

Gandon, F. (2000). Distributed Artifical Intelligence and Knowledge Management: Ontologies and Multi-Agent systems for a Corporate Semantic Web. *Scientific Philosopher Doctorate Thesis in Informatics*, INRIA and University of Nice.

Gašević, D.; Djurić, D. & Devedžić, V. (2006). *Model Driven Architecture and Ontology Development*, Springer-Verlag Berlin Heidelberg, ISBN-10: 3540321802, Germany

Grimm, S.; Hitzler, P. & Abecker, A. (2007). Knowledge Representation and Ontologies, In: *Semantic Web Services*, Studer, R.; Grimm, S. & Abecker, A., (Ed.), page numbers (51-105), Springer Berlin Heidelberg New Vork, ISBN-13: 9783540708940, Online edition

Griss, M.L.; Favaro, J. & d' Alessandro M. (1998). Integrating Feature Modeling with the RSEB, *Proceedings of the 5th Internarional Conference on Software Reuse*, pp. 76, ISBN-10: 0818683775, Victoria, British Columbia, Canada, IEEE Computer Society Washington, DC, USA

Harsu, M. (2002). A survey on domain engineering. *Report 31*, Institute of Software Systems, Tempere University of Technology

Hayes, R. (1992). Measurement of information, *Information Processing & Management*, Vol. 29, No. 1, (January-February 1993), page numbers (1-11), ISSN: 0306-4573

Heidorn, G. E. (1975). Augmented phrase structure grammars, *Proceedings of the 1975 Workshop on Theoretical issues in natural language processing,* pp. 1-5, Cambridge, Massachusetts, June 1975, Association for Computational Linguistics, Morristown, NJ, USA

Hjorland, B. (1998). Information retrieval, text composition and semantics, *Knowledge Organization,* Vol. 25, No. 1-2, (1998), page numbers (16-31), ISSN: 0943-7444

Kang, K.; Cohen, S.; Hess, J.; Novak, W. & Peterson, S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study, *Technical CMU/SEI-90-TR-21*, Software Engineering Institute, Carnegie Mellon University

Kang, K. C.; Kim, S.; Lee, J.; Kim, K.; Shin, E. & Huh, M. (2004). FORM: A feature-oriented reuse method with domain-specific reference architectures, *Annals of Software Engineering*, Vol. 5, No. 1, (January 1998) page numbers (143-168), ISSN: 1573-7489

Kendal, S. & Creen, M. (2007). *An Introduction to Knowledge Engineering,* Springer, ISBN: 1846284759, United States of America

Kosar, T.; Martinez Lopez, P. E.; Barrientos, P. A. & Mernik, M. (2008), A preliminary study on various implementation approaches of domain-specific language, *Information and Software Technology*, Vol. 50, No. 5, (April 2008) page numbers (390-405), ISSN: 0950-5849

Krishnamoorthy, C. S. & Rajeev, S. (1996). *Artifical Intelligence and Experts Systems for Engineers*, CRC-Press, ISBN-10: 0849391253, USA

Luger, G. F. (2005). *Artificial intelligence, Structure and Strategies for Complex Problem Solving (Fifith Edition)*, Pearson Education Limited, ISBN-10: 0321263189, USA

Mernik, M.; Heering, J. & Sloane, A. M. (2005). When and how to develop domain-specific languages, *ACM Computing Surveys (CSUR)*, Vol. 37, No. 4, (December 2005), page numbers (316-344), ISSN: 0360-0300

Miller, G. A. (1995). WordNet: A Lexical Database for English, *Communications of the ACM,* Vol. 38, No. 11, (November 1995) page numbers (39-41), ISSN: 0001-0782

Minker, J. (1987). *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Pub, ISBN: 0934613400, United States

Partee, B. H.; ter Meulen, A. & Wall, R.E. (1993). *Mathematical methods in linguistics*, Kluwer Academic Publishers, ISBN-10: 9027722454, Netherlands

Pradorn, S.; Nopasit, C.; Yacine, O.; Gilles, N. & Abdelaziz, B. (2007). Knowledge Engineering Technique for Cluster Development, Springer, ISBN-13: 9783540767183

Russell, S. & Norvig, P. (2003). *Artifical Intelligence A Modern Approach*, Prentice Hall, ISBN: 0131038052, United States of America

Schmid, G. (1994). TreeTagger - a language independent part-of-speech tagger, available on *http://www.ims.uni-stuttgart.de/Tools/DecisionTreeTagger.html*

Schreiber, G.; Akkermans, H.; Anjewierden, A.; de Hoog, R.; Ahadbolt, N.; Van de Velde, W. & Wielinga, B. (1999). *Knowledge Engineering and Management: The CommonKADS Methodology*, The MIT Press, ISBN: 0262193000, USA

Shadbolt, N. & Milton, N. (1999). From Knowledge Engineering to Knowledge Management, B*ritish Journal of Management*, Vol. 10, No. 4, (December 1999), page numbers (309-322), IISN: 1045-3172

Sheth, A.; Ramakrishnan, C. & Thomas, C. (2005). Semantics for the Semantic Web: The Implicit, the Formal and the Powerful, *International Journal on Semantic Web and Information Systems,* Vol. 1, No. 1, (January-March 2005), page numbers (1-18), ISSN: 1552-6283

Simons, M. & Anthony, J. (1998). Weaving the Model Web: A Multi-Modeling Approach to Concepts and Features in Domain Engineering, *Proceedings of the 5th International Conference on Software Reuse*, pp. 94-102, ISBN: 0818683775, Victoria, DC, Canada, June 1998, IEEE Computer Society Washington, DC, USA

Sterling, L. & Shapiro, E. (1994). *The Art of Prolog, Second Edition: Advanced Programming Tecniques (Logic Programming)*, The MIT Press, ISBN: 0262193388, USA

Stokes, M. (2001). *Managing Engineering Knowledge. MOKA - Methodology for Knowledge-Based Engineering Applications*, John Wiley & Sons Australia, Limited, ISBN: 1860582958

Taylor, N. R.; Tracz, W. & Coglianse, L. (1995). Software development using domain-specific software architectures, *ACM SIGSOFT Software Engineering Notes*, Vol. 20, No. 5, (December 1995) page numbers (27-38), ISSN: 0163-5948

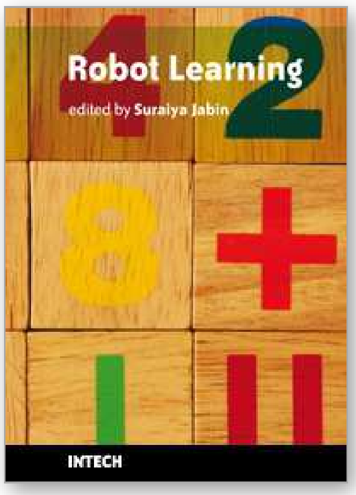
Valente, G. (2004). Artifical Intelligence methods in Operational Knowledge Management, Ph.D. Dissertation, University of Turin

Winograd, T. (1972). *Understanding natural language,* Academic Pr, ISBN-10: 0127597506, Orlando, Florida, USA

Weiss, D. M. & Lai, C. T. R. (1999). *Software Product-Line Engineering: A Family-Based Software Development Process*, Addison-Wesley Professional, ISBN: 0201694387

Woods, W.A. (1970). Transition network grammars for natural language analysis, *Communciations of the ACM,* Vol. 13, No. 10, (October 1970), page numbers 591-606, ISSN: 0001-0782

Zins, C. (2007). Conceptual approaches for defining data, information and knowledge, *Journal of the American Society for Information Science and Technology*, Vol. 58, No. 4, (February 2007), page numbers (479-493), ISSN: 1532-2882



Robot Learning
Edited by Suraiya Jabin

ISBN 978-953-307-104-6
Hard cover, 150 pages
Publisher Sciyo
Published online 12, August, 2010
Published in print edition August, 2010

Robot Learning is intended for one term advanced Machine Learning courses taken by students from different computer science research disciplines. This text has all the features of a renowned best selling text. It gives a focused introduction to the primary themes in a Robot learning course and demonstrates the relevance and practicality of various Machine Learning algorithms to a wide variety of real-world applications from evolutionary techniques to reinforcement learning, classification, control, uncertainty and many other important fields. Salient features: - Comprehensive coverage of Evolutionary Techniques, Reinforcement Learning and Uncertainty. - Precise mathematical language used without excessive formalism and abstraction. - Included applications demonstrate the utility of the subject in terms of real-world problems. - A separate chapter on Anticipatory-mechanisms-of-human-sensory-motor-coordination and biped locomotion. - Collection of most recent research on Robot Learning.