ISSN 1407-7493 **DATORZINĀTNE** **2009-7493**
COMPUTER SCIENCE
10.2478/v10143-010-0015-9

INFORMATION TECHNOLOGY AND MANAGEMENT SCIENCE
INFORMĀCIJAS TEHNOLOĢIJA UN VADĪBAS ZINĀTNE

USAGE OF ONTOLOGIES IN SYSTEMS OF DATA EXCHANGE

ONTOLOĢIJAS PIELIETOJUMS DATU APMAIŅAS SISTĒMĀS

Pavel Osipov is a Ph.D. student in the Institute of Information Technology, at Riga Technical University. He received his master's diploma in Computer Science from Transport and Telecommunication Institute in 2007. His research interests include web data mining, machine learning and knowledge extraction.

Arkady Borisov is Professor of Computer Science in the Faculty of Computer Science and Information Technology at Riga Technical University (Latvia). He holds a Doctor of Technical Sciences degree in Control in Technical Systems and the Dr.habil.sci.comp. degree. His research interests include fuzzy sets, fuzzy logic and computational intelligence. He has 205 publications in the area.

Keywords: machine learning, ontologies, semantic WEB, OWL, RDF, software agents, attribute value taxonomies

Abstract - This paper describes the methods and techniques used to effectively extract knowledge from large volumes of heterogeneous data. Also, methods to structure the raw data by the automatic classification using ontology are discussed. In the first part of the article the basic technologies to realize the Semantic WEB are described. Much attention is paid to the ontology, as the major concepts that structure information on a very high level. The second part examines AVT-DTL algorithm proposed by Jun Zhang which allows one to automatically create classifiers according to the available raw, potentially incomplete data. The considered algorithm uses a new concept of floating levels of ontology; the results of the tests show that it outperforms the best existing algorithms for creating classifiers.

Introduction

The complexity of the structure of the modern information society is constantly growing. Due to that, the requirements for effective processing algorithms also increase. A general model of knowledge extraction from raw data is shown in Fig. 1. The most popular recent trends in this area are Data Mining (DM), Knowledge Discovery in Databases (KDD) and Machine Learning (ML). They provide theoretical and methodological framework for studying, analyzing and understanding large amounts of data.

However, these methods are not sufficient if the structure of the data is poorly suited for machine analysis, which can be observed today on the Internet. A need constantly arises to provide a convenient interface to access various data and provide an opportunity to consider them with a convenient point of view in such diverse areas as bio-informatics, e-commerce, etc.

To solve that problem, a global initiative is launched that is aimed at restructuring the Internet data in order to transform it into the Semantic Web, which provides opportunities for efficient retrieval and analysis of data, both for humans and software agents. The most important part of the Semantic Web is the use of ontologies that provide opportunities for the representation of some data from different points of view, according to the needs and skills of those who make requests.

The major drawback of the existing structure of the Internet is that it practically does not use standards for the presentation of data easier to understand the computer, and all information is intended primarily for human consumption. For example, in order to get working hours of a family doctor, a human can just go to the clinic site and find the information needed on the list of all practicing physicians. However, what is easy to make for a human is virtually impossible for the software agent in the automatic mode unless you build it, subject to a rigid structure of a specific site.

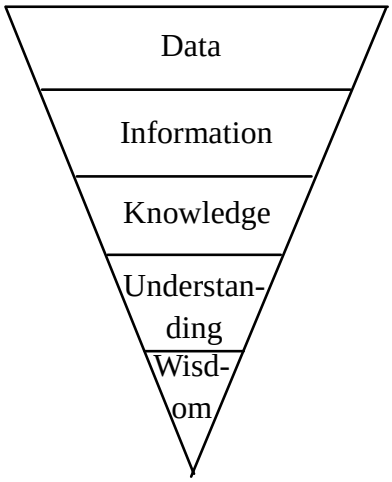

Fig. 1. Knowledge distillation process

To solve these problems, *ontology* is used that allows describing any of the subject area in understandable for the machine terms and enables effective use of *mobile agents*.

Using this approach, in addition to the visible to humans data on each page, there are also ancillary data allowing efficient use of data by software agents.

In turn, the ontology is part of a global vision of the Internet development to a new level, called the **Semantic WEB** (SW) [1].

The Most Important Concepts of Semantic WEB

To achieve a goal such as the global reorganization of the global network, a set of interrelated technologies is required. Figure 2 provides the overall structure of the concepts of Semantic WEB; a brief description of key technologies is given below.

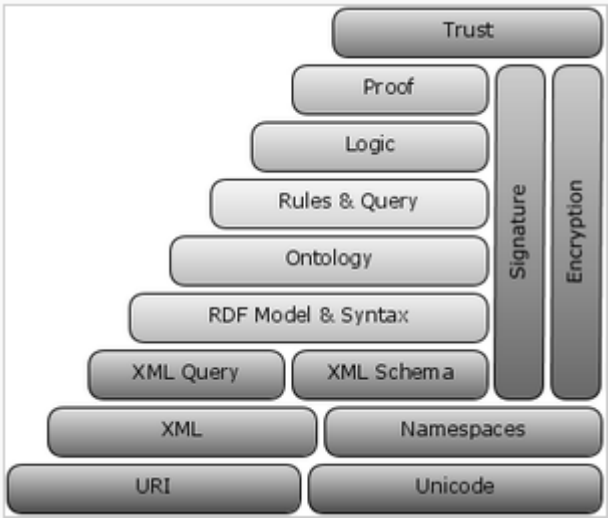

Fig. 2. Stack of semantic web concepts

Semantic WEB

The concept of the semantic web is central to the modern understanding of evolution of the Internet. It is believed that the future data network will be presented both in the usual form of pages and as metadata, in about the same proportion, which will allow machines to use them for logical conclusions realizing the benefits from the use of methods of ML. Everywhere there will be used *Uniform Resource Identifiers* (URI) and ontology.

However, there are doubts about the full realization of the semantic web. Key Points for the benefit of doubt in the possibility of creating an efficient semantic web are:

- The human factor [2]: people can lie, be lazy, add a meta description and use incomplete or simply incorrect metadata. As a solution to this problem, automated tools to create and edit metadata can be used.
- Unnecessary duplication of information, where each document must have a full description of both for the man and for the machine. This is partly solved by the introduction of *microformats* [3].

In addition to the metadata itself, the most important part of the SW is the *semantic Web services*. They are a source of data for agents of semantic web. They are originally designed to interact with the machines and have a means for advertising their opportunities.

URI (Uniform Resource Identifier)

URI is a uniform identifier of any resource. It may indicate both the virtual and the physical object. In essence, it is a unique character string, whose common structure is represented in Fig. 3. Currently the best-known URI is a URL, which is the identifier of the resource on the Internet and additionally contains information about the whereabouts of the addressed resource.

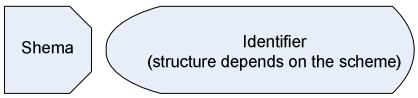

Fig. 3. Base URI format

Ontologies

For the area under the Machine Learning, ontology means a structure or conceptual framework for describing (formalizing) the values of some elements of the *subject domain* (SD). Ontology consists of a set of terms and rules describing their relationships.

Usually, ontologies are built from *instances*, *concepts of attributes* and *relationships*.

- *Instances* are elements of the lowest level. The main purpose of ontology is precisely the classification of specimens; although their presence in the ontology is not required, usually they are present.
  Example: the word, the breed of animals, stars.
- *Concepts* are abstract sets and collections of objects.
  Example: The concept of "stars", the embedded concept of "sun". What is the "sun", or the concept of nested instance (celestial body), depends on the ontology.
  The concept of "celestial body", a copy of "sun".
- *Attributes* - Each object can have an optional set of attributes allowing to store specific information.
  Example: object "sun" has attributes like these:
  - *Type:* yellow dwarf;
  - *Mass*: $1.989 \cdot 10^{30}$ kg;
  - *Radius*: 695 990 km.
- *Relationships* - allow you to specify relationships between objects of the ontology.

Since it is possible to establish points of intersection between different ontologies, the use of ontologies allows us to look at one SD from different points of view and depending on the task to use different levels

of detail of the considered SD. The concept of levels of detail is one of the key concepts of the ontology. For example, to refer to the color of traffic lights, it is sometimes sufficient to simply specify it as "*green*", while for dyeing machines even such a detailed description of colour as "*dark green, similar in tone to the needles*" may not be sufficient.

Let us consider the general structure of ontologies usages.

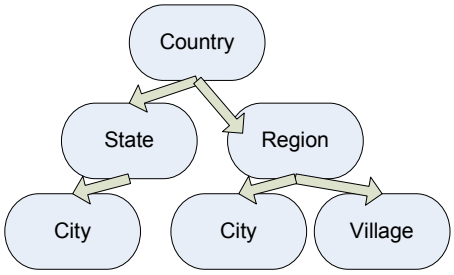

Fig. 4. Part of the possible ontologies addresses

In the case of the ontology shown in Fig. 4, in order to send a letter to an American university, it suffices to specify its name; a software agent finds its own address on the basis of standard address information from the university. If you wish to send a letter to a specific department, then the list of all the faculties will be obtained from the site and the correct one will be selected. From the site of the required faculty the address will be taken. Using the above ontology the program will then determine the address format valid in the United States.

The computer does not understand all the information in the full sense of the word, but the use of ontologies allows it to use the available data much more efficiently and intelligently.

Of course, many questions remain, for example: How can the agent find the desired site of the university in the beginning? However, for that purpose the tools are already developed, for example, an ontology language Network Services (Web Services Ontology Language, OWL-S [4]) which allows services to advertise their capabilities and services.

Fig. 5 shows a variant of the division of ontologies by formalization levels that represent the ontological accuracy of each method.

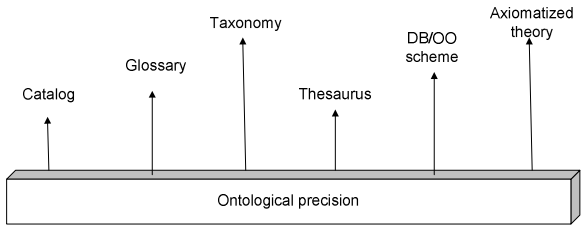

Fig. 5. Hierarchy taxonomies

Under the ontological accuracy there is understood the ability of the ontology to distinguish between shades of meaning.

- *Catalog* – contains a set of normalized terms without any hierarchical structure.
- *Glossary* – all terms are sorted in alphabetical order of natural language.
- *Taxonomy* – partially ordered collection of concepts (taxons) that divides the subject domain from general to specific concepts.
- *Axiomatised taxonomy* – taxonomies subset; it contains certain axioms in a formal language.
- *Context libraries / axiomatised ontologies* – are interrelated set of axiomatised taxonomies. They are the intersection of several contexts, or the inclusion of one context in another.

Taxonomies

Taxonomies are one of the options for implementing ontologies. With the help of taxonomy one may define the classes into which objects of a certain subject area are divided, as well as what relations exist between these classes. Unlike ontologies, taxonomies task is clearly defined within the hierarchical classification of objects.

Attribute Value Taxonomies (AVT)

An important example of the concept of taxonomy in Machine Learning is the concept of *Attribute Value Taxonomies (AVT)*.

Consider a formal definition of AVT. Suppose there is a vector of attributes $A=\{A_1, A_2, ..., A_N\}$, and $dom(A_i)$ defines a set of attribute values A_i. In this case, the formal definition of AVT is as follows:

Attribute Value Taxonomies T_i of attribute A_i is a tree-like structure containing a partially ordered hierarchy of concepts presented in the form of a set $(dom(A_i), \prec)$, where $dom(A_i)$ is a finite set containing all the attributes in A_i, and $\prec$ is the sign of a partial order defining the relationship among the values of attributes in $dom(A_i)$. Together, $T=\{T_1, T_2, ..., T_N\}$ represent an ordered set of taxonomies of attribute values connected with $A_1, A_2, ..., A_N$.

Let $Nodes(T_i)$ present a set of all values T_i, and $Root(T_i)$ point to the root element of the taxonomy T_i. The set of leaves $Leaves(T_i)$ represents

all the primitive attributes A_i. Then the tree $Nodes(T_i) - Leaves(T_i)$ will be called abstract values of attributes A_i. Every edge of the tree represents the ***is-a*** relation between the attributes of the entire taxonomy. Thus AVT defines an abstract hierarchy between the values of attributes.

Modern Languages for Describing Ontologies

RDF (Resource Description Framework) [5] is a language for describing metadata resources; its main purpose is to represent the allegations so that they are equally well perceived by both human and machine.
The atomic object of RDF is a triple which consists of subject, predicate and object. It is considered that any object can be described in the terms of simple properties and values of those properties.

Table 1
Sample of table with selected parameters

:me	:owns	:my_appartment
:me	:owns	:my_computer
:me	:is_in	:philadelhia
:my_appartment	:has_a	:my_computer
:my_appartment	:has_a	:my_bed
:my_appartment	:is_next_to	:my_bed

Before the colon the *Uniform Resource Identifier* (URI) should be disclosed, but in order to save traffic it is allowed to specify only the namespace.

Additionally, in order to improve the perception of human, there is the practice of representation of the RDF in the form of graphs.

Fig. 6 shows an example representation of the structure of RDF as a graph. Ribs are statements, the name at the beginning of the rib is the subject of the statement; the name at the end of the statement is its addition, but at the very name of the arc - the predicate.

OWL (Web Ontology Language) [6] is a web ontology language created to represent the value of terms and relations between these terms in dictionaries. Unlike RDF, the language uses a higher level of abstraction that allows the language along with formal semantics to use an additional glossary of terms.

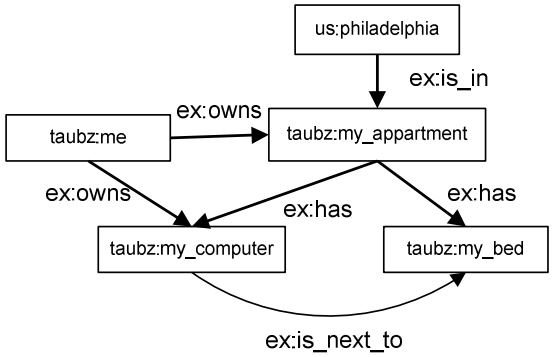

Fig. 6. Example of representation of RDF schema as a graph

An important advantage of OWL is that it is based on precise mathematical model description logic [7].

OWL place in Semantic Web Concepts stack is shown in Fig. 7.

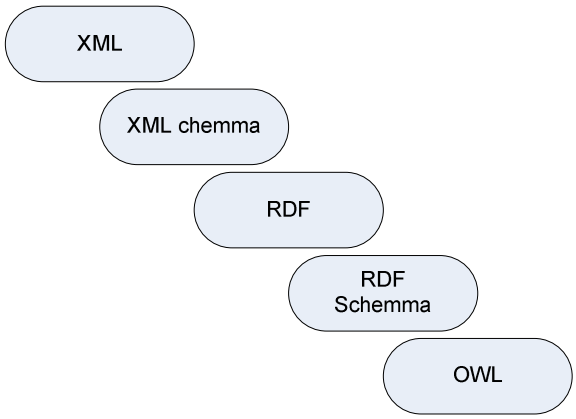

Fig. 7. W3 consortium structure of the Semantic Web vision

- XML – offers the possibility of creating structured documents, but imposes no semantic requirements on them;
- XML Schema – defines the structure of XML documents and additionally allows using specific types of data;
- RDF – provides an opportunity to describe the abstract data model, some objects and relations between them. Uses simple semantics of XML-based syntax;
- RDF Schema – allows one to describe properties and classes of RDF - resources, as well as the semantics of relations between them;
- OWL – expands the descriptive capabilities of previous technologies. Allows you to describe the relations between classes (for example Nonintersecting), cardinality (e.g. "exactly one"), symmetry, equality, enumerated types of classes.

By the degree of expressiveness there can be distinguished three dialects of OWL:

- *OWL Lite* – is a subset of the complete specification, providing minimally adequate means for describing ontologies. This dialect is designed to reduce the initial introduction of OWL and also to simplify the migration to OWL thesauri and other taxonomies. It is guaranteed that the logical conclusion to the metadata with the expressiveness of OWL Lite runs in polynomial time (complexity of the algorithm belongs to the class P).
  This dialect is based on the description logic $SHLF(D)$.
- *OWL DL* – on the one hand, it provides maximum expressiveness, computational completeness (all of them are guaranteed computed) and complete decidability (all computations are completed in a certain time). But due to that has strict restrictions, for example, on the relationship of classes, and the execution time of some queries on such data may require exponential runtime.
  The dialect is based on the description logic $SHOLN(D)$.
- *OWL Full* – provides maximum expressive freedom, but gives no decidability guarantees. All the structures developed are only justified by the feasible algorithm. It is unlikely that any reasoning software will be able to maintain full support for every feature of OWL Full. This dialect does not match any description logic as it is not resolvable in principle.

Currently OWL language is a basic tool for describing ontologies.

Software (Mobile, User) Agents

In our SD, software agents (SA) is the program acting on behalf of the user to perform independently gathering information for some, perhaps a long time while away entirely from the predictable environment. Also important feature is their ability to interact and cooperate with other agents and services to achieve the goal.

In contrast to the bots of search engines, which simply scan a range of WEB pages, the agents move from one server to another, i.e., at the sending server the agent is destroyed, and the host is created with a full set of previously collected information. This model enables the agent to use the available to servers data sources that are not accessible through the WEB interface.

It is clear that a platform must be installed at the server that enables the agent to accept and accommodate its requests. It is also important to pay attention to the security and integrity of agents. For this, the approach of selected spaces is employed in which the agent operates in a safe environment with limited rights and impact opportunities on the system.

With regard to the implementation, the agents are usually divided into two categories according to the level of their freedom:

- *Non-intelligent agents* – are inherently similarity function with a clearly defined number of parameters. It is strictly intended to perform a specific task in strictly defined conditions.
- *Intelligent agents* – are free to choose the technology to be used in each case to solve the problem. Often such programs are implemented by using neural networks, which allow the agent to constantly adapt to changing conditions.

Microformats

Microformats are an attempt to create a semantic markup entities in a variety of Web-pages that are equally perceived by both humans and machines. Information in a microformat does not require the use of additional technology or the namespace in addition to the simple (X) HTML. Microformat specification is simply an agreement on standards for naming classes of elements of registration page to enable storing in each of the classes relevant data.

To illustrate that, let us analyze the format of hCalendar. This microformat is a subset of the iCalendar format (RFC 2445) and is intended to describe the dates of future or past events to provide opportunities for their automatic aggregation of search agents.

```
<div class="vevent">
  <a class="url" href="http://www.web2con.com/">
    http://www.web2con.com/
  </a>
  <span class="summary">
    Web 2.0 Conference
  </span>:
  <abbr class="dtstart" title="2007-10-05">
    October 5
  </abbr>  -
  <abbr class="dtend" title="2007-10-20">
    19
  </abbr>
  ,at the
  <span class="location">
    Argent Hotel, San Francisco, CA
  </span>
</div>
```

This example describes the establishment of the root class of the container with the date (class = "vevent") and relating to an event a certain date in the standard ISO date format.

Currently the most common microformats are

- hAtom - newsletters format;
- hCalendar - compiling a calendar of events and description;
- hCard - description of the people, companies and places;
- hResume - format for resume description;
- hReview - reviews implementation;
- XFN - the way to specify relationships between people.

Modern Trends in Development of Ontologies

Preparation of ontologies using an expert approach is a very slow and expensive process; and although the result may be very good, the vast majority of users are employing automatic classifiers that build ontologies according to available data. This way, however, has many troublesome places, and therefore existing automatic classifiers are not of good quality and one of the options for increasing the accuracy of classification is the *Ontology Aware Classifiers* [9].

When using automatic methods of data mining (data-driven knowledge discovery) sometimes there is a need to investigate the information received from different points of view. This is especially important in scientific research, when the result of data analysis depends strongly on the used for their submission ontology. In particular, it is one of the most important moments in the procedure of automatic learning classifier data based on their ontologies. For example, if the analyzed data in different parts of the attributes are represented with different accuracy (which is very common in real applications, for example, if the data were originally collected from various sources, with different terminology), nowadays there is no algorithm that would take this into account, use different levels of ontologies for different parts of the data and work effectively with potentially inadequately specified data.

The following requirements are prerequisites for the establishment of a new generation of classifiers:

- simple, powerful and robusta classifiers are required;
- classifiers using abstract attributes often provide the most simple model of the separation of data;
- in the case of insufficient data, the score obtained on the basis of abstract attribute values is often more reliable than that obtained on the basis of primitive values.

Fig. 8 shows examples of partially and fully defined shapes. For example, partially defined are data such as {blue, ellipse}, {sky blue, triangle}, whereas fully determined are these data: {light blue, square}.

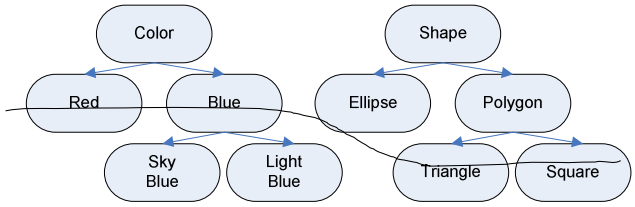

Fig. 8. Part of ontologies for colours and descriptions of figures

There is a method for creating a learning classifier templates based on Attribute Value Taxonomies (AVT) in the presence of potentially insufficient data. The algorithm uses a decision tree and is called (**AVT-DTL** Attribute Value Taxonomies - Decision Tree Learning algorithm).

There is an ontology of colours (see Fig. 8), and in the analyzed data some attributes are described as blue, some like red, and some like sky-blue. In this case, the algorithm producing a classification based on multiple levels of the ontology would be more accurate.

The classical method for constructing the classification tree at each step selects the most informative attribute, and declares it the node at the current iteration. In contrast, AVT-DTL for each iteration chooses not simply an attribute but the level of taxonomy, in which it is represented. This level is represented as a vector of numbers of all levels involved in the classification taxonomies. Then, on the basis of all created this way vectors the lower bound is created on the minimum values of the vectors for each taxonomy. Finally, checks on the adequacy of informativeness of the obtained constraints are made.

Let us define some relevant terms.

- AVT(*A*) – taxonomy of attribute *A*;
- Leaves(AVT(*A*)) – a set of all possible values of *A*;
- all internal, intermediate nodes called abstract;
- a cut is called an imaginary line abstraction taxonomy, for example (red, blue), (red, sky blue, turquoise).

To achieve each leaf, there is one path T, which can be represented as a set of paths to all the parent nodes T = $\{T_1, T_2,..T_i\}$, where T_i = AVT(A_i).
Also, in addition, the following operations on the taxonomy of T_i are used that are associated with the attribute A_i:

- *prim* (*Ti*)=*Leaves*(AVT(*Ai*)); number of nodes in the i-th taxonomy;

- *depth*(*Ti* , *v*(*Ai*)); length of the path from the root to the value of the i-th attribute taxonomy;
- *leaf*(*Ti* , *v*(*Ai*)); boolean value, it is true if the i-th attribute in the taxonomy of the *Ti* is a leaf, i.e., *v*(*Ai*) ∈ *Leaves*(*Ti*).

It is believed that the attribute is completely defined if it has the value of one of the leafs of taxonomy $A \in Leaves(\mathrm{AVT}(A))$, otherwise the attribute is defined partially.

The data are declared completely defined if each attribute has the most complete description: $\forall i\ v_i^{(j)} \in prim(T_i)$.

The considered algorithm is a search from top to bottom over the space of hypotheses for constructing a decision tree. The algorithm gives preference to the division by possibly more abstract (close to the root) attributes. This allows for each attribute to choose its level of abstraction in the taxonomy.

In the description of the algorithm the following definitions are used:

- $A=\{A_1, A_2, \ldots, A_n\}$ ordered set of attributes.
- $T=\{T_1, T_2, \ldots, T_n\}$ taxonomies set.
- $C=\{C_1, C_2, \ldots, C_m\}$ set of target mutually disjoint classes.
- $\psi(v, T_i)$ – set of descendants of the node corresponding to the value v in the taxonomy T_i.
- *Children* (v, T_i) - set of descendants of the node v in the taxonomy T_i.
- $\Lambda(v, T_i)$ – list of ancestors, including the root, for *v* in T_i.
- $\sigma_i(v, S)$ – number of occurrences of value *v* of attribute A_i in a training set S.
- *Counts*(T_i) – tree whose nodes contain numerical values of attributes in the taxonomy T_i.
- $P_s = \{p_1^{(s)}, p_2^{(s)}, .., p_n^{(s)}\}$ - vector of values (also called the boundary AVT), consisting of the values of nodes of S, where node $P_i^{(s)}$ indicates the node value T_i of attribute A_i.

$\Phi(P_s) =$ true, only if $\forall p_i^{(S)} \in P_s$, and $\Psi(p_i^{(S)}, T_i) = \{\}$.

The AVT-DTL algorithm works in the direction of AVT from the root of each tree to the leaves of trees and builds solutions that use the most abstract attributes which are sufficiently informative to classify the training set, containing a potentially insufficient data. In its work, the algorithm, based on existing AVT, conducts a search algorithm, hill-climbing (in which each next node is selected by increasing its proximity to the decision) over the space of hypotheses about the options for constructing a decision tree.

The algorithm execution consists of two main steps:

1. Calculation of weights on the basis of AVT.
2. Building a decision tree, based on the weights obtained.

Let us consider these steps in more detail.

1. Calculation of weights on the basis of AVT consists in the following:
   a. Creating a root element, setting a training set S, completing the set $P_S=\{A_1, A_2, \ldots, A_n\}$ so that each of the elements of the vector uses the value of the corresponding taxonomy $T_1, T_2, \ldots, T_n$.
   b. Initializing the calculation of weights based on the analysis of training data;
      i. Accumulation of weights associated with each value of each attribute of the training set. Thus each $T_i \in T$ is calulated $\sigma_i(v, S)$ for each taxonomy.
      ii. For each $T_i \in T$ the weights of all elements of the parent are updated if at the previous step for the i-th node the nonzero value was received. The result is a tree *Counts*(T_i).
      iii. For each $T_i \in T$ and for all incomplete attributes value *v* in each case $Ij \in S$, recursively calculates the weight values for all descendants *v* in taxonomy T_i on the basis of *Counts*(T_i) and rebuilds *Counts*(T_i). Thus, for each d from $\psi(v, T_i)$, rebuilds $\sigma_i(d, S)$ using

$$\sigma_i(d,S) \leftarrow \sigma_i(d,S)\left(1+\frac{1}{\sum_{d \in Children(v,T_i)} \sigma_i(d,S)}\right) \quad (1)$$

this formula:

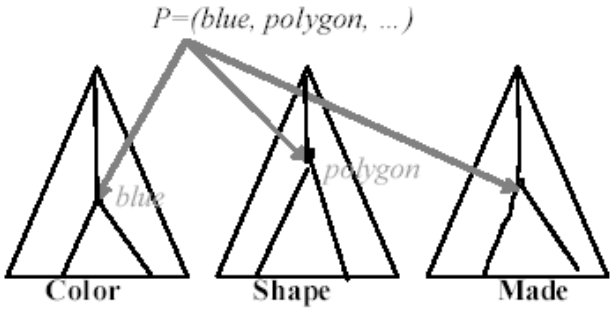

Fig. 9. Sample of the vector pointer p

As an illustration, Fig. 9 represents a vector pointing to the two attribute values *blue* and *polygon* for two taxonomies used for the approximately following rule: If (*Color=blue & Shape=polygon &*…) then *Class* =+.

2. Building a decision tree based on the obtained weights consists of the following operations ($Tree-Build(S,P_s)$):

   a. If every element of the training set S is marked as C_i, then C_i is returned, otherwise if $\Phi(P_s)=true$ then there is a need to create the final (leaf) element (Stopping Criterion).

   b. For each pointer $P_i^{(S)}$ in vector P_s do:

      i. Check each part of a null value with a partially specified attribute *v* to attribute A_i with pointer $P_i^{(S)}$. If *depth*(*Ti* , $P_i^{(S)}$) < *depth*(*Ti* , *v*), a partially unspecified value is treated as fully qualified and is sent to the appropriate level of decision tree with root $P_i^{(S)}$. Otherwise replace *v* with probability *d* (element *Children (v, T_i))* in accordance with the distribution of weights associated with the *Children (v, T_i)* by the following rule:

$$\Pr(d)=\frac{\sigma(d,T_i)}{\sum_{e\in Children(v,T_i)}\sigma(e,T_i)} \qquad (2)$$

      ii. Calculate the entropy of S on the basis of its separation with regard to *Children ($p_i^{(S)}$, T_i).*

   c. Selecting the best pointer $P_a^{(S)}$ in P_S so that this division of S would provide the maximum information gain.

   d. Separation of set S into subsets $S_1,S_2,...,S_{|Children(P_a^{(S)},T_i)|}$ using data from *Children ($p_a^{(S)}$, T_i).*

   e. Extension of P_S to obtain a new pointing vector of the subset $S_1,S_2,...,S_{|Children(P_a^{(S)},T_i)|}$ by changing the pointer $Pa^{(S)}$ to value of attribute A_i from corresponding subset S_j $(1\le j\le|Children(p_a^{(S)},T_i)|)$.

   f. For each $(1\le j\le|Children(p_a^{(S)},T_i)|)$ do $Tree-Build(S_j,P_{s_j})$.

Algorithm Evaluation

To confirm the effectiveness of this algorithm, a comparative test on the same sample with another, the most popular at the moment algorithm C 4.5 was made. The results proved the effectiveness of the algorithm, particularly important was the marked increase in the quality of the final classifier constructed from the sample with a high percentage of missing data. The final payoff is within a 5% -30% increase in the efficiency of classification. It has also been noted that classifier trees constructed by the AVT-DTL algorithm contain fewer nodes and are more compact, which allows a gain in the amount of resources required for their use.

Conclusions

At the moment being it is difficult to predict what the systems of data exchange will be in the future, however, the directions of development are being laid right now and their effectiveness depends on many things.

This paper describes the basic technologies that allow you to create complex systems that can be used effectively by both humans and machines. In the stack of Semantic WEB concepts at various levels a large number of different technologies are included, but only their joint application would lead to synergies.

Since the effectiveness of technologies for each level of the stack affects the whole system, it can be concluded that the considered **AVT-DTL** algorithm by improving the quality of automatic classifiers will in the long run increase the effectiveness of the Semantic Web.

References

1. Main page of W3 consortium, Semantic Web ideology creator http://www.w3.org/2001/sw/.
2. Cory Doctorow Metacrap: Putting the torch to seven straw-men of the meta-utopia http://www.well.com/~doctorow/metacrap.htm.
3. Main microformats ideology site http://microformats.org/.
4. Main page of W3 consortium, Web Ontology Language for Services http://www.w3.org/Submission/2004/07/.
5. Main page of W3 consortium, Resource Description Framework http://www.w3.org/Submission/2004/07/.
6. Main page of W3 consortium, Web Ontology Language (OWL) http://www.w3.org/2004/OWL/.
7. Baader F., Horrocks I., Sattler U., Description Logics as Ontology Languages for the Semantic

Web // Berlin: Springer ISSN 0302-9743, February 2005. - P. 228-248.
8. Main page of microformats community http://microformats.org/.
9. Zhang J., Learning ontology aware classifiers // Dissertation of the requirements for the degree of DOCTOR OF PHILOSOPHY / Iowa State University 2005

Pāvels Osipovs, Arkādijs Borisovs. Ontoloģijas pielietojums datu apmaiņas sistēmās

Rakstā aprakstītas metodes un paņēmieni, ko izmanto, lai efektīvi izgūtu zināšanas no liela apjoma jaukta tipa datiem. Tāpat apskatītas metodes izejas datu strukturizēšanai, izmantojot automātisko klasifikāciju un izmantojot ontoloģijas. Raksta pirmajā daļā apskatītas pamata tehnoloģijas, kas ļauj realizēt Semantisko WEB. Liela uzmanība tiek veltīta ontoloģijām kā svarīgākajiem elementiem, kuri nodrošina informācijas strukturizēšanu ļoti augstā līmenī. Otrajā daļā apskatīts AVT-DTL algoritms, kuru piedāvā Jun Zhang. Tas ļauj automātiski izveidot klasifikatorus, izmantojot pieejamus, iespējams nepilnīgus, datus. Apskatītais algoritms izmanto jaunu koncepciju - peldošā līmeņa ontoloģijas, un, kā parāda testu rezultāti, darbojas labāk nekā esošie algoritmi klasifikatoru radīšanai.

Павел Осипов, Аркадий Борисов. Использование онтологий в системах обмена данных

В статье рассмотрены методы и технологии, используемые для эффективного извлечения знания из больших массивов разнородных данных. Также рассмотрены методы структуризации сырых данных путём автоматической классификации с использованием онтологий. В первой части статьи рассмотрены основные технологии, позволяющие реализовать Semantic WEB. Большое внимание уделено онтологиям как важнейшему понятию, позволяющему структурировать информацию на очень высоком уровне. Вторая часть рассматривает алгоритм AVT-DTL, предложенный Jun Zhang, который позволяет автоматически создавать классификаторы по имеющимся сырым, потенциально неполным данным. Рассмотренный алгоритм использует новое понятие нефиксированных уровней онтологии и, по результатам тестов, превосходит лучшие существующие алгоритмы создания классификаторов.