Distributed knowledge-based medical system for data assimilation and analysis

V. Tomenko & V. Popov
Wessex Institute of Technology, Southampton, UK

Abstract

Internet and Global Information Systems present unique opportunities to collect, assimilate and analyse data from different sources. Ontologies allow domain experts to create, share and reuse existing knowledge in a standardized way. These advantages can be exploited in medicine in order to create a unified assimilation and analysis system. An ontology-based server-side application for distributed medical system was developed. The application uses data from a medical knowledge base to answer client queries. Additionally, data may be collected and assimilated into the knowledge base to improve the quality of the analysis.
Keywords: artificial intelligence, diagnosis, distributed system, knowledge base, prediction.

1 Introduction

In recent years the development of ontologies has been moving from the realm of research centres to the desktops of domain experts. Ontologies have become common on the World-Wide Web. They range from large taxonomies categorizing Web sites to categorizations of products for sale and their features (such as on Amazon.com). Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. Medical experts, for example, have produced standardized, structured vocabularies such as SNOMED [1] and the semantic network of the Unified Medical Language System [2].

Ontology can be defined as explicit formal specifications of the terms in the domain and relations among them [3]. This includes description of classes in the specific domain (concepts), properties of each class describing various features

WIT Transactions on Information and Communication Technologies, Vol 38, © 2007 WIT Press
www.witpress.com, ISSN 1743-3517 (on-line)
doi:10.2495/DATA070271

and attributes of the concept (slots), and restrictions on slots (facets). Ontology together with a set of individual instances of classes constitutes a knowledge base (KB). Classes are the focus of most ontologies. A class can have subclasses, which represent concepts that are more specific than the superclass. Slots describe properties of classes and instances.

The reasons for creating an ontology include:

- sharing common understanding of the structure of information among people or software agents;
- enabling reuse of domain knowledge;
- making explicit domain assumptions;
- separating the domain knowledge from the operational knowledge;

Developing an ontology involves sequence of steps, repeated iteratively:

1) Defining classes in the ontology.
2) Arranging the classes in a taxonomic (subclass–superclass) hierarchy.
3) Defining slots and adding restrictions to slots values.

Often an ontology of the domain is not a goal in itself. Developing an ontology is akin to defining structure of data for other programs to use. Problem-solving methods, domain-independent applications, and software agents use ontologies and KB built from ontologies as data.

New technologies (e.g. Internet and Global Information Systems) present unprecedented opportunities to conduct surveillance and knowledge accumulation more efficiently than ever before. The aim of this study was to develop an ontology-based server-side application for a distributed system in order to collect and assimilate medical data from different sources and provide supportive information for medical organizations, doctors and patients.

2 Protégé

Among the numerous ontology-management tools Protégé is found to be one of the most widely used and promising approaches [4]. Protégé is an extensible, platform-independent environment for creating and editing ontologies and KB [5]. There are several features that distinguish Protégé from other KB editing tools, such as:

- Scalability: there is virtually no deterioration in performance between several hundred frames and thousands of frames.
- Extensible plug-in architecture: Protégé can be extended with domain and task-specific plug-ins.

Protégé can be used in two different ways. First of all it provides uniform GUI, which allows domain-experts to create ontology structure and populate KB with particular instances. It is also available to use Protégé's core in other applications.

Ontology building blocks in terms of Protégé are shown in Fig. 1. For each class a set of slots is defined describing its properties. Each slot includes associated facets, which vary according to slot type. Facet values impose restrictions on slot values. Standard facets include VALUE-TYPE (String, Float,

Instance etc.) and CARDINALITY group (defines if value is required, how many values per slot are required and allowed). Additional facets are dependent on VALUE-TYPE facet value.

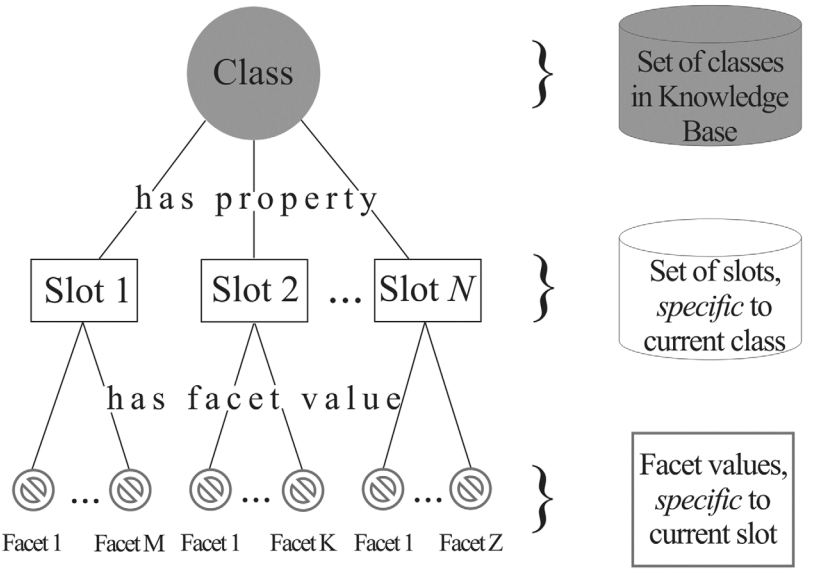


Figure 1: Ontology building blocks in terms of Protégé.

3 Data assimilation and analysis application

Server-side application uses Protégé core to manipulate KB. KB is stored on server as Protégé project file and can be accessed and modified via Protégé API. The application can be used either within Protégé GUI as plug-in or separately by utilizing project file.

The scheme of the system is shown in Fig. 2. Four main modules include:

1) data filtering and assimilation module;
2) data preprocessing module;
3) analysis model creation module;
4) analysis and query resolving module.

Typical sequence of actions is the following. Server receives query from a client and the query contains raw data, which should be processed. Data filtering and assimilation module receives raw data as input and assigns them to one or several existing classes in the KB. In assimilation mode, instances of selected classes can be created, filled with corresponding data values and added to KB. In query resolving mode filtered data (represented by pattern matrix [6]) are preprocessed and converted into internal format, appropriate for analysis. After that, analysis and query resolving module receives internal data and trained

model (created by algorithm, specific to query provided) and produces query result, which is then sent to a client. Additionally, the result of the analysis (e.g. classification) can be assimilated into the KB and the trained model can also be refined and saved. The detailed description of the main modules and their interaction is given in the foregoing sections.

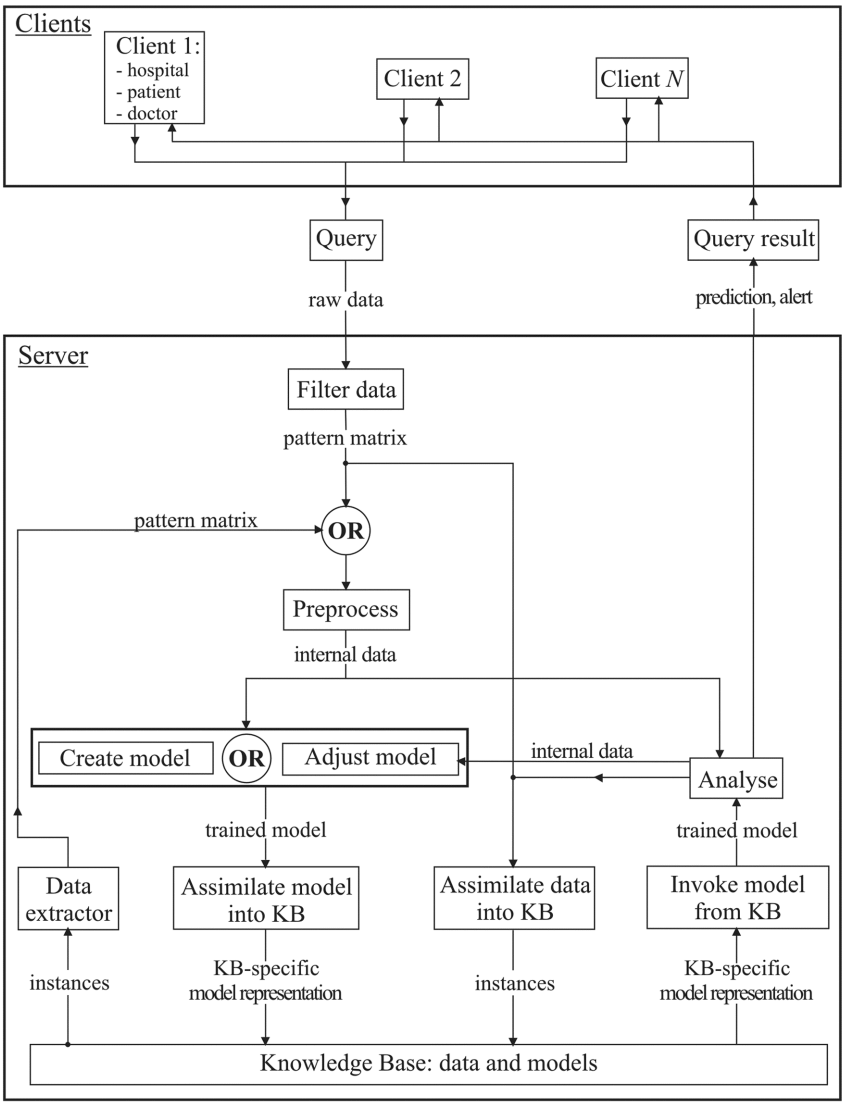


Figure 2: Scheme of the system.

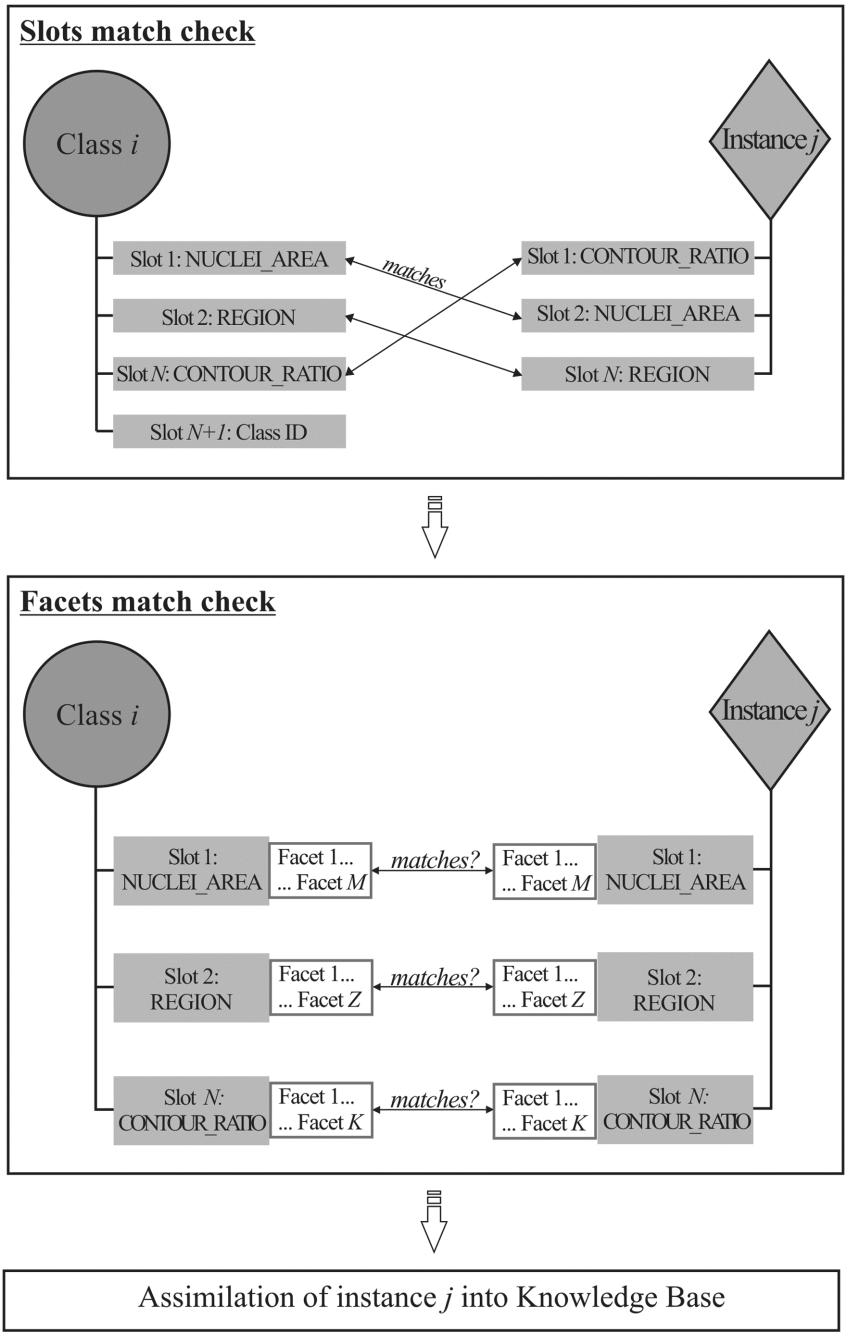


Figure 3: Data filtering and assimilation module.

3.1 Data filtering and assimilation

The input to the module is raw data from the client, which should be assigned to class/classes in the KB. For this assignment, two steps are performed (Fig. 3). After filtering data variables' names are compared to classes' slots names. Then, the set of presumably allowed classes is checked for facets correspondence. If all the restrictions are satisfied, instances are added to the allowed class/classes.

3.2 Data preprocessing

Data preprocessing module converts data pattern matrix into internal format suitable for algorithm execution. Preprocessing includes selection, normalization and transformation of variables. For example, prior to classification variables can be normalized as follows [6]:

$$\hat{x} = \frac{x - \min_j}{\max_j - \min_j} \tag{1}$$

where $\min_j$ and $\max_j$ are minimal and maximal values of the variable in column j respectively, or

$$\hat{x} = \frac{x - m_j}{\sigma_j} \tag{2}$$

where m_j and σ_j are mean and standard deviation of the variable in column j respectively. Additionally, if the dimensionality of input patterns is high, it can be reduced via linear or nonlinear dimensionality reduction technique [7].

3.3 Analysis model creation

Given internal data, this module either creates analysis model as the outcome of algorithm execution or adjusts existing model. If the created/adjusted model has satisfactory performance, it can be assimilated into the KB and later used by analysis and query resolving module.

For example, if the task is to predict target variable value (e.g. benign/malignant, disease class etc.) given input parameters, classification algorithm can be selected and corresponding model can be created. For this purpose Learning Vector Quantization (LVQ) algorithm can be utilized, which is the extension of Self-Organizing Map (SOM) [8].

LVQ learning process includes self-organizing (unsupervised) phase followed by supervised phase [9]. In its turn, self-organizing phase is comprised of ordering stage and fine-tuning stage. Self-organizing phase is comprised of two steps [8]. When input pattern x is presented, unit (neuron) with the nearest reference vector (vector of weights) is defined:

$$s(x) = \arg\min_{i \in A} \|x - w_i\| \tag{3}$$

Then, weights of the nearest unit and its topological neighbours are updated according to:

$$w_i(t) = w_i(t-1) + \eta(t) G_i(t)(x(t) - w_i(t-1)) \tag{4}$$

where $\eta(t)$ is the learning-rate factor on iteration t, $G_i(t)$ is the neighbouring function value. These two steps are repeated $t_{\max}$ iterations. At the ordering

stage, the neighbourhood radius and learning-rate factor are chosen big enough to allow all units change their reference vectors when different input patterns are presented. The neighbourhood radius significantly decreases with iterations. At the end of this stage the ordering of units in input and output spaces resembles each other.

During fine-tuning stage the neighbourhood radius continues to decrease monotonically, but much slower than during first stage. Together with the small values of learning-rate factor these changes allow adjusting reference vectors more precisely.

At supervised phase input patterns are iteratively picked at random. For each iteration, if the class labels of input pattern x and unit defined by equation (3) agree, the reference vector of the unit is shifted towards x:

$$w_i(t) = w_i(t-1) + \alpha(t)(x(t) - w_i(t-1)) \tag{5}$$

Otherwise, it is moved away from x:

$$w_i(t) = w_i(t-1) - \alpha(t)(x(t) - w_i(t-1)) \tag{6}$$

where $\alpha(t)$ is the learning-rate factor for supervised phase.

The outcome of LVQ learning process is the set of labelled units together with corresponding reference vectors. If the performance is adequate, they can be regarded as trained model and saved in the KB.

This module is easily extendable by task-specific algorithms and models, which can be stored in the KB. Once the task is specified, the module's API can be used to add new algorithms and model descriptions.

3.4 Analysis and query resolving

Analysis and query resolving module receives preprocessed query data and trained model from the KB as input. The outcome of the analysis is the report, which is sent to the client.

For example, client sends test results, which are specific to the disease or group of diseases of interest and expects to receive indication of presence or probability of having disease. Analysis module processes the data provided using corresponding trained model from the KB and assigns target variable values to the test results. Then the target values are sent to the client.

Additionally, query data can be assimilated to extend the KB. In this case data are sent to filtering and assimilation module, which creates instances, initializes them with data values and saves in the KB.

Finally, if the trained model requires adjustment, internal data are sent to analysis model creation module, which adjusts the model and assimilates it into the KB.

4 Experimental results

For evaluation purposes, the system was applied to predict particular disease occurrence. The task can be specified as follows: given input parameters from the patient or doctor (e.g. test results) system should prove able to classify them

to one of several predefined classes with adequate precision. In order to test the performance of the algorithms two datasets were chosen:

1) Breast Cancer dataset [10];
2) Stomach Disease dataset [11, 12].

A dataset summary is given in Table 1. For both datasets, Cancer ontology [13] was extended and utilized for knowledge assimilation. KB was populated with data from selected datasets using filtering and assimilation module and then normalized using preprocessing module. Analysis model creation module was utilized to invoke LVQ algorithm, create trained model for prediction and assimilate it into the KB. Finally, analysis and query resolving model was used to predict target variable values for incoming test results of unknown class. Results of the evaluation are discussed in the foregoing sections.

Table 1: Summary of evaluation.

	Breast Cancer	**Stomach Disease**
Dataset properties:		
Number of patterns	683	13300
Number of attributes	9	27
Type of attributes	ID number (1-10)	Continuous
Class distribution	Benign: 65.5 % Malignant: 34.5%	Cancer: 21,95% Gastritis: 23,68% Inflammatory Displasia: 2,33% True Displasia: 2,78% Ulcer: 49,26%
LVQ parameters:		
Number of units	100	625
Number of iterations	5000	15000
$\eta(t)_{initial} - \eta(t)_{final}$	0,5 – 0.1	0,5 – 0.1
$\sigma(t)_{initial} - \sigma(t)_{final}$	6,0 – 0,2	25,0 – 0,2
$\alpha(t)_{initial} - \alpha(t)_{final}$	0,1 – 0,005	0,1 – 0,005
Test Set size	10%	20%
Model performance:		
Learning Set	98,8%	82,14%
Test Set	97,5%	79,21%

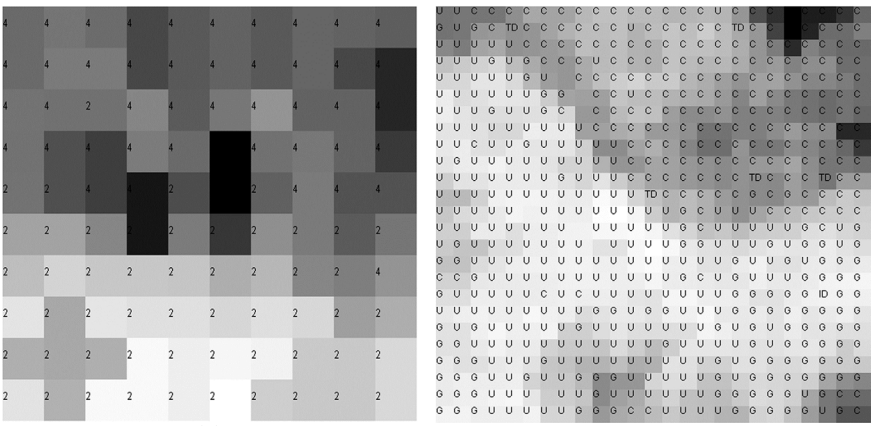
4.1 Breast Cancer occurrence prediction

The task was to assign class value (benign or malignant) to patterns. Initially, SOM model was trained to explore cluster structure, and the resulting distance image [14] is shown in Fig. 4(a). 2D visualization suggests that classes are well separated.

LVQ was used to create classification model. LVQ parameters and trained model performance are summarized in Table 1. Confusion matrix is given in Table 2. Based on obtained results, the application of LVQ for breast cancer occurrence prediction is found to be promising.

Table 2: Confusion matrix for Breast Cancer dataset.

		Experimental	
		Malignant	Benign
Predicted	Malignant	22	0
	Benign	2	44



(a) (b)

Figure 4: Distance images for datasets.

4.2 Stomach Disease prediction

The second task was stomach disease prediction, which was performed by assignment of gastric cells to one of the five categories (cancer, gastritis, inflammatory displasia, true displasia and ulcer). Distance image (Fig. 4(b)) suggests that classes are not well separated, which is supported by the nature of the problem [12] and classification results. LVQ parameters and trained model performance are summarized in Table 1. Confusion matrix is given in Table 3. The overall performance for given task is acceptable for supportive information, except for inflammatory displasia recognition (this class is very close to ulcer class).

Rescaling the model by adjusting more units may slightly improve classification accuracy but this will require additional computational resources.

5 Conclusions

Ontologies provide efficient way to share, reuse and extend domain-specific knowledge, which is particularly useful in such areas as medicine. With the help of uniform knowledge representation provided by medical ontologies data can be assimilated and analyzed more effectively. The implemented knowledge-based system combines ontology-based data representation and distributed sources of data (e.g. hospitals) for client-specific query resolving. The system is extendable and can be used to solve such tasks as prediction, outbreak alerts, medical advice etc. Server-side application allows creating centralized knowledge pool and analysis engine.

Table 3: Confusion matrix for Stomach Disease dataset.

		Experimental				
		Cancer	Gastritis	Infl. displasia	True displasia	Ulcer
Predicted	Cancer	510	3	15	27	6
	Gastritis	5	423	12	8	157
	Inf. displasia	0	1	2	0	0
	True displasia	18	15	2	41	4
	Ulcer	26	213	38	1	1133

References

[1] Price, C. and Spackman, K. (2000). SNOMED clinical terms. BJHC&IM-British Journal of Healthcare Computing & Information Management 17(3): 27-31.

[2] Humphreys, B.L. and Lindberg, D.A.B. (1993). The UMLS project: making the conceptual connection between users and the information they need. Bulletin of the Medical Library Association 81(2): 170.

[3] Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5: 199-220.

[4] OntoWeb. D1.3. A survey on ontology tools IST-2001-29243.

[5] Protégé Project. http://protege.stanford.edu

[6] Jain, A. K., Dubes, R. C., Algorithms for Clustering Data, Prentice Hall: Englewood Cliffs, New Jersey, 1988.

[7] Tomenko, V., and Popov, V. Nonlinear dimensionality reduction of large datasets for data exploration. Data Mining VII, 2006.

[8] Kohonen T. Self-Organizing Maps. Berlin: Springer, 2001.

[9] S. Haykin, Neural Networks: a Comprehensive Foundation, 2nd Ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[10] UCI ML Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html

[11] Stomach Disease Dataset: http://cgi.di.uoa.gr/~makis/projects/CCS/CCS.html

[12] K. Koutroumbas, G. Paliouras, V. Karkaletsis and C.D. Spyropoulos, "Comparison of Computational Learning Methods on a Diagnostic Cytological Application". Proceedings of the EUNITE, pp. 500-508, Tenerife, Spain, 2001.

[13] DAML Ontology Library, Cancer Ontology: http://www.mindswap.org/2003/CancerOntology/nciOncology.owl

[14] J. Mao, and A. K. Jain, 1995. Artificial neural networks for feature extraction and multivariate data projection, IEEE Trans. Neural Networks, vol. 6, pp. 296-317.