Data Mining Use for Learning Process Design of an Information Source Locator Agent

Christian Böhm, María Rosa Galli and Omar Chiotti
GIDSATD-UTN- Facultad Regional Santa Fe, Lavaise 610- 3000 Santa Fe- Argentina
INGAR - CONICET, Avellaneda 3657 - 3000 Santa Fe - Argentina
chiotti@ceride.gov.ar, mrgalli@ceride.gov.ar

Abstract

The aim of this work is to present a data mining application to software engineering. We describe the use of data mining in some parts of the design process of a dynamic decision support system agent-based architecture. The main function of this system is to guide information requirements from users to the domains that offer greater possibilities of answering them. For that purpose, a strategy is developed, which provides the system with capacity for analyzing an information requirement, and determining to which domains it will be directed. To learn from errors made during its operation, a learning mechanism based in CBR techniques is also proposed.

On the one hand, by using data mining techniques it is possible to define a discriminating function to classify the system domains into two groups: those that can probably provide an answer to the information requirement made to the system, and those that cannot.

On the other hand, the application of data mining to the cases base allows the specification of rules to settle relationships among the stored cases with the aim of inferring possible causes of error in the domains classification. In this way, a learning mechanism is designed to update the knowledge base and thus improve the already made classification as regards the values assigned to the discriminating function.

Keywords: Data mining, learning process, DSS, Cases base

Data Mining Use for Learning Process Design of an Information Source Locator Agent

1. Introduction

The enterprise management involves making many different decisions. An enterprise can be generally considered as being organized in several *domains*. Many of these enterprise domains, such as product design, planning, control, scheduling, forecasting and sales constitute different decision points and are generally located on different functional units geographically disperse. They use specific models and techniques to each decision type, and need to share knowledge and information. Therefore, a Decision Support System (DSS) that involves the whole organization should be designed as a set of geographically distributed subsystems (Domains), operating with the smallest level of possible joining, which we will call distributed Decision Support System (Figure 1).

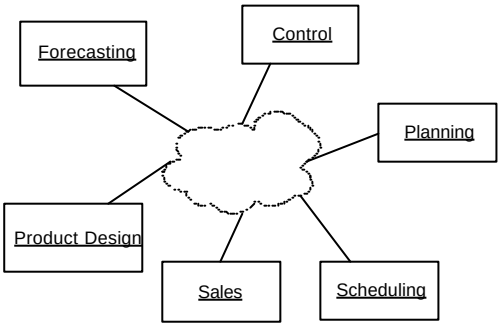

Figure 1: Distributed Decision Support System Domains

There are two kinds of information needs: the one that can be predicted on design time and the one that can not (this occurs whenever new or non-habitual decisions should be made). To bring support to the first type of information requirement, several enterprise integration systems have been designed (Shen and Norrie, 1999). To attend the second type of information requirement, we are developing a multi-agent system that we call dynamic DSS (Cabral et al., 2000). A dynamic DSS is a distributed DSS able to work in a dynamic way. That is, a *system able to look for information, analyzing where it is available or can be generated*. This system is composed by *Domain Representative Agents* (DRAs); an *Information Source Locator Agent* (ISLA) and mobile agents called *Query Coordinator Agents* (QCAs) (Figure 2).

When a user of the system needs some information, makes a query in natural language and the dynamic DSS transfers that information requirement to a domain that can satisfy it. For that purpose, the system determines the sites that offer a greater possibility of providing the required information and those are firstly targeted. Then, when it gets the answer from a domain, it takes this information to the domain that asked for it.

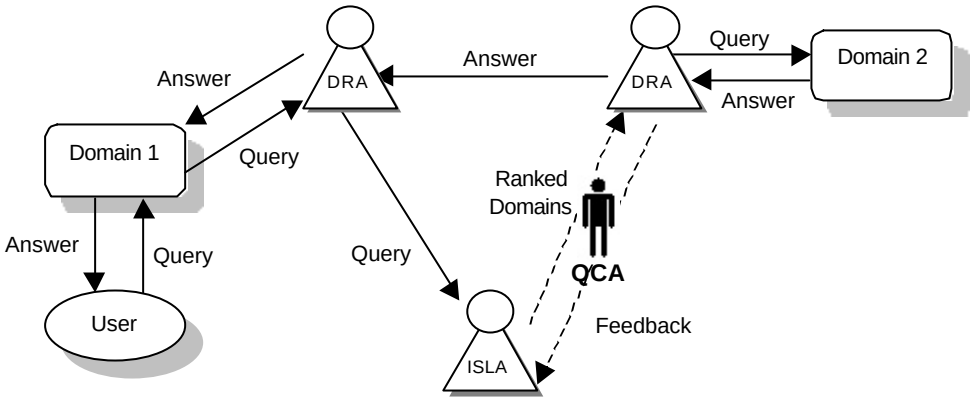

Figure 2: Multi-agent architecture for the dynamic DSS

The component in charge of doing such a determination is the ISLA. This agent has knowledge about the information that can be provided by domains and the capacity for updating that knowledge, learning from cases that result from its operation. The main function of this agent is to deliver every query to a domain capable of answering it, avoiding the overload of the system sending the queries to all the domains (Figure 3).

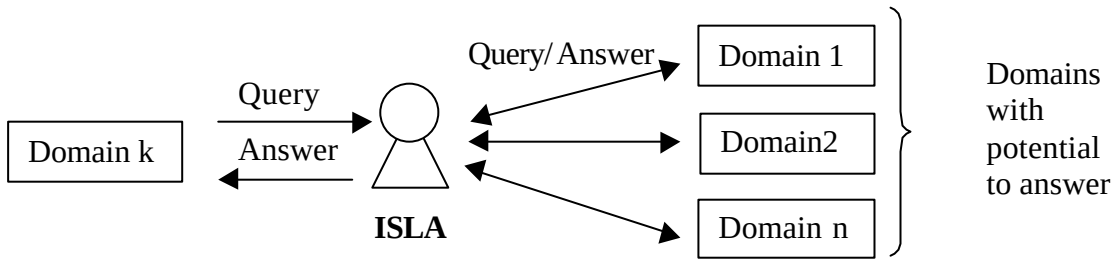

Figure 3: The main function of ISLA

Dingsoyr (1998) presents several possible uses of data mining and case-based reasoning. He classifies these uses into two groups: data mining in case-based reasoning, and case-based reasoning in data mining. He sketches the following possible uses:

Data mining search is the case. That is, the information about the search results and the whole knowledge discovery in database process might be stored in a case so that extra time will not be spent on mining the same information more than once. Rodriguez et al. (2000) presented a heterogeneous architecture for knowledge extraction from data. The architecture combines the knowledge discovery in database process with a case-based reasoning system intended to record the successful knowledge extraction experiments to be shared later on.

Case-based reasoning provides information that can be used in providing some background knowledge about features in a database (for instance the weight of features for a classifier can be learned).

Find features from cases base for a case to classify the cases in the cases base. This might speed up the computation of similarity metrics in the *retrieve* step.

Find features from a database for a case to supplement the information given in it.

Find domain knowledge by mining a database or a cases base in the form of functions, rules or causal graphs which can be later used to identify features and explain away unimportant features.

Construct artificial cases from a database that are not present in a cases base. This would require a tight integration where the data mining algorithm would search for patterns in the form of cases, which could be evaluated by a novelty function that gives high values to cases not present in the cases base.

Ruiz et al. (2001) describe a project of data mining application on software engineering. Specifically, they propose to simulate the designed system to generate cases. Such cases are mined to detect possible behavioral patterns of the design.

It is necessary to highlight that data mining and case-based reasoning applications on system design offers a still little developed potential. On the other hand, in designing an information system it is possible that several of the alternatives sketched by Dingsoyr (1998) are simultaneously used.

The aim of this work is to describe the usage of data mining to design the learning process of the ISLA. That is, to define the data structure of the system cases base and the rules of relationships among cases.

This paper is structured as follows. In section 2 a view into the system functionality is presented. Then, in section 3, it is showed how the use of data mining allows designing the learning process of the ISLA. Finally, three rules obtained by the mining process, which can be used to develop the system learning process, are presented as example.

2. A brief view into the system functionality

To understand the learning process design, more knowledge about the system functionality is needed.

The query: when a decision maker needs information, he/she can make a query in natural language with the help of a user interface. This query, in order to be processed by the ISLA is filtered becoming a set of keywords without connectors, articles, prepositions, etc.

$$q=\{keyword_1, keyword_2 ... keyword_n\}$$

The Domain Knowledge Base: the ISLA has a representation of the information managed by the domain d_j.

$$Kd_j=\{(\ keyword, weight)/\ 0 \leq weight \leq 1\}$$

where *keyword* is one of the words that identify the knowledge managed by domain d_j, and *weight* is the strength that such a word represents this knowledge. The set of keywords in Kd_j defines a taxonomy of domain d_j.

This information must be updated by both increasing/decreasing weights or adding/removing keywords from the Kd_j, with the aim to adapt the ISLA to its environment changes (Stegmayer et al., 2001)

The Discriminant function (RSV): let D be the set of all domains being part of the system. Given an information requirement q_i, this function must define the subset of domains $D_P \subset D$, which have the potential for answering the information requirement.

Then, for each information requirement q_i there will be a set of domains D_P that have the potential for providing the required information, and a set of domains D_N that are not able to provide the required information. In this way, the process can be seen as a domains classification into: domains with potential for providing the required information and domains without it.

To carry out such a classification, a discriminating analysis can be employed (Kachigan, 1991). We define the discriminating function with a qualitative criterion variable defined by the classification labels [Potential domain] and [Non potential domain]

$$RSV(d_j,q_i) = w_1 p_1 + ... + w_k p_k + ... + w_n p_n$$

where RSV is the resulting domain's discriminating score.

Each domain (object) $d_j \in D$ will have a value on this discriminating function depending upon its values on the predicting variables $p_1, ... p_k, ... p_n$.

Every predicting variable is related to a keyword in Kd_j, and it is 1 if such a keyword belongs to the query, otherwise it is 0 and w_k are the weights associated to each keyword in Kd_j.

The criterion variable is qualitative, and thus it will be necessary to define a cutoff score. The cutoff score will be equal to or greater than zero. Domains with RSV equal to or greater than the cutoff score are assigned to the D_P criterion set, and domains with a RSV lower than the cutoff score are assigned to the D_N criterion group.

On the other hand, given an information requirement q_i, the RSV value of each domain classified into the D_P criterion set can be used to rank these domains. That is, the domain $d_j \in D_P$ with the greatest RSV will be ranked first, the domain $d_j \in D_P$ with the second greatest RSV will be ranked second, and so forth.

Figure 4 shows an schematic representation of the main steps carried out by the DSS to bring support to a decision maker. Note that the learning functionality is still missing.

Classification errors may affect the system efficiency and thus other domains d_j whose $RSV(d_j,q_i)$>cutoff scores are assigned to the D_P criterion set. In fact, this classification error only affects the system efficiency when $RSV(d_j, q_i) > RSV(d_R, q_i)$. In that case, domain $d_R \in D_P$ but it does not have the first place in the ranking.

Classification errors of the discriminating function may be due to two main causes:

- The value of weights w_k is not the right one.
- Not all keywords (predicting variables) characterizing the domain are included.

It is necessary to highlight that these errors can be originated by the evolution of domains that could take place as time goes by. Then, it will be necessary to update the system knowledge base to avoid these errors. This updating can be carried out by analyzing the results of all those cases in which there were classification errors. In other words, domain d_R that answered to the respective query q_i did not have the greatest $RSV(d_R, q_i)$ and therefore it was not ranked first. This process can be defined as a learning process that must be structured so that it can be automatically

developed by the system. In the following section, we describe the use of data mining for designing the learning process.

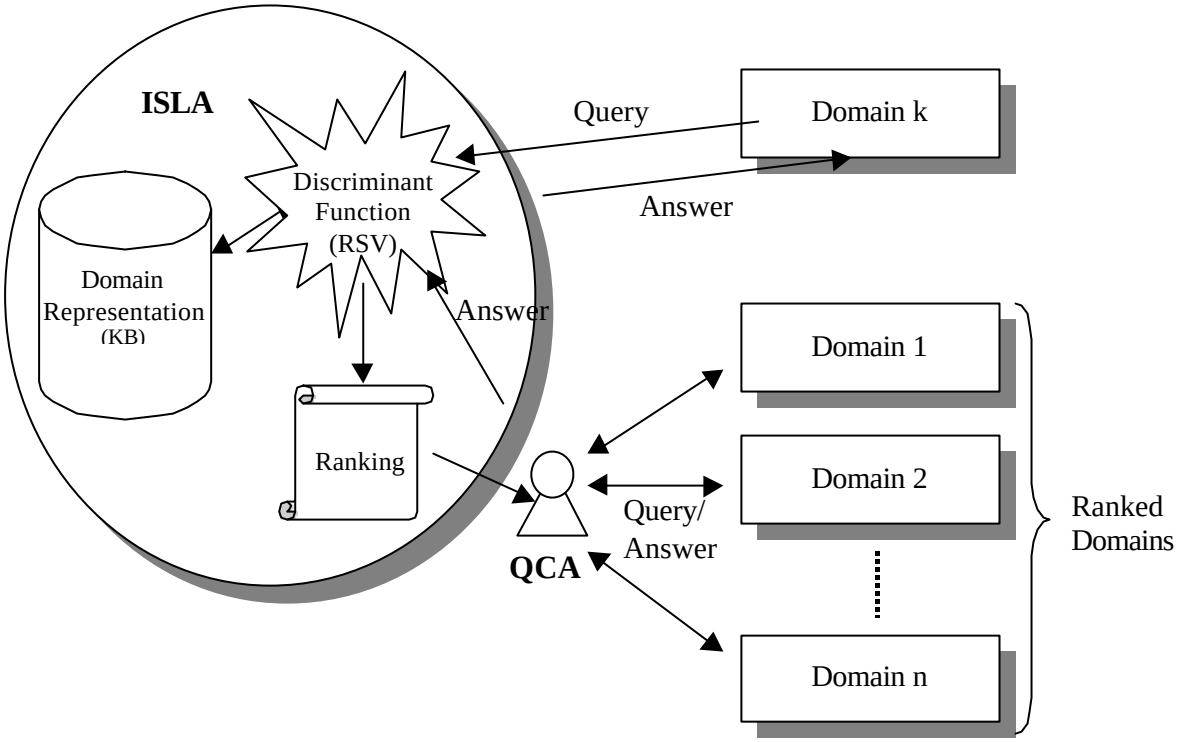

Figure 4: Main steps carried out by the DSS to bring support to a decision maker

3. Use of Data Mining in Designing the Learning Process

The aim of the learning process is to attain the agent autonomy. That is to say, that the agent be able to adapt itself to the environment changes. An agent will be autonomous when its behavior is defined by its experience (Russell and Norvig, 1995). So, the learning process of the ISLA conceptually consists of analyzing the data of stored cases (feedback from the answering domains) to apply rules that allow updating the discriminating function of the system domains, which, as it was previously indicated, conform the system knowledge base.

The first step in designing the learning process is to design a cases-base in which the data of queries and their respective results are stored and in which we would be able to use data mining techniques to infer rules for the learning process. Figure 5 shows an schematic representation of the learning process.

Conventional data mining processes are carried out on a data base whose structure was designed without taking into account such a process. For this reason, these processes at their initial stages involve tasks such as selection, preprocessing, transformation, etc. that are needed to generate a convenient data structure to be analyzed. A way of simplifying the mining process is to take into account these tasks while designing the data structure to store cases in the cases base.

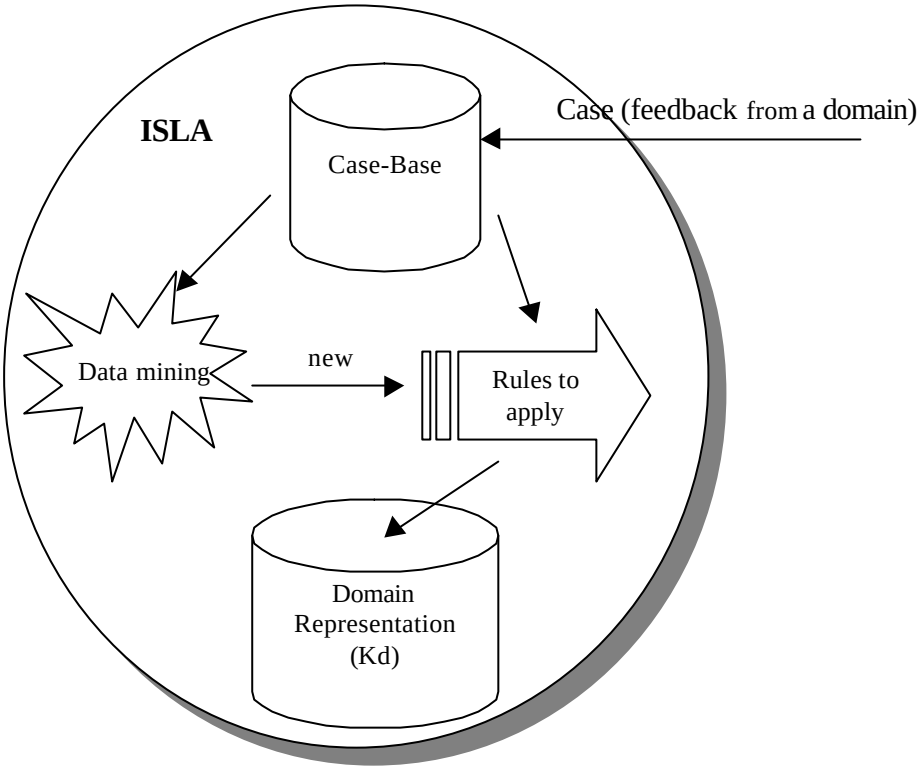

Figure 5: Schematic representation of the learning process

Table I presents a logical structure of data in the cases base that allows observing a domain's behavior before the various queries. This structure will store, for a given domain d_j, all queries q_i, for which RSV(d_j, q_i) > cutoff score. Essentially, this data structure uses binary fields to represent the relationship between a query q_i, which is defined by the set of keywords Kq_i, and the domain taxonomy Kd_j.

Table I: Logical structure of data for the domain d_j.

Query	*VA*	*OR*	P_1	...	P_k	P'_1	...	P'_m
q_i								

The first field, Query, stores the name that identifies each query q_i. The second field, called VA, stores the qualitative variable value $va(d_j, q_i)$ that can take [Positive, Negative, Null] values. The Positive value indicates that the required information q_i was provided by the consulted domain d_j. The Negative value indicates that the consulted domain d_j answered to the information query q_i in a negative way and the Null value indicates that domain d_j did not provide any answer to query q_i. According to what has been described in previous sections, in order to determine possible errors in the classification efficiency, it does not matter the RSV(d_j, q_i) value itself, but the relative position of each domain $d_j \in D_P$. Therefore, the third field (Order) stores the order that domain obtained in the ranking of domains classified as potential to answer to the query q_i. The remaining

fields are divided into two groups: In the first group, each field represents a keyword of the domain taxonomy. Each field is designated with a keyword $p_k \in Kd_j$, and represents a binary variable p_k that takes 1 as value if the keyword p_k of the domain taxonomy is in the query q_i, and takes 0 as value if p_k is not stated in that query.

$$p_k = \begin{cases} 1 & if \quad p_k \in Kd_j \wedge p_k \in Kq_i \\ 0 & if \quad p_k \in Kd_j \wedge p_k \notin Kq_i \end{cases}$$

In the second group, each field represents a keyword stated in query q_i that does not belong to the domain taxonomy. Each of these fields is designated with a keyword p'_k and represents a binary variable p'_k that takes 1 as value if the keyword p'_k is stated in the query q_i but does not belong to the domain taxonomy d_j, and takes 0 as value if p'_k is not in that query.

$$p'_k = \begin{cases} 1 & if \quad p'_k \notin Kd_j \wedge p'_k \in Kq_i \\ 0 & if \quad p'_k \notin Kd_j \wedge p'_k \notin Kq_i \end{cases}$$

We have a structure for each system domain. Each structure stores the way in which the domain behaved for the different queries for which it has been classified as potential.

In the following section, we will see how this logical data structure meets the posed learning needs.

3.1. Inferring New Rules with Data Mining

Once the logical data structure is designed and the cases are stored, the latter must be analyzed using data mining. The object is to analyze the data kept in the cases base so as to identify relationships among the data from which possible behavior patterns of cases can be defined. Such patterns are used to define rules for updating the discriminating function of each domain, which is stored in the knowledge base. This requires working with the cases data associated with the involved domain. For this purpose, the logical data structure presented in Table I of the previous section will be used. According to what has been discussed, the updating can be performed in two ways: either updating the weights of the keywords (predicting variables) that define the domain taxonomy or modifying the domain taxonomy by adding new keywords.

3.2. Patterns Discovery

Once a significant number of cases q_i are stored, we can perform a mining of these data to look for patterns. For that purpose, we define Q_{dj} as the set of stored cases q_i associated to d_j. That is, Q_{dj}= {q_i / RSV(d_j, q_i) > cutoff score}.

To start with, cases $q_i \in Q_{dj}$ are classified into four groups. A first group formed by cases in which no classification error occurred. A second group of cases in which the domain provided a positive answer but it was not the first one in the ranking of potential domains. These are cases in which the classification error affected the system efficiency. A third group of cases, in which the domain provided a negative answer, and finally a fourth group of cases in which the query was not answered.

To carry out this classification, $va(d_j,q_i)$ and $or(d_j,q_i)$ are defined as predicting variables.

Group of efficient cases Q^{+}_{dj}: integrated by those cases that present $va(d_j,q_i)$ = positive and $or(d_j,q_i)$ =1

$$Q^{+}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{positive} \wedge or(d_j,q_i) = 1\}$$

Group of non-efficient cases Q^{*}_{dj}: integrated by those cases that present $va(d_j,q_i)$ = positive but $or(d_j,q_i)$ >1

$$Q^{*}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{positive} \wedge or(d_j,q_i) > 1\}$$

Group of negative cases Q^{-}_{dj}: integrated by those cases that were answered in a negative way.

$$Q^{-}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{negative}\}$$

Group of Null cases Q^{0}_{dj}: integrated by those cases in which an answer was not provided.

$$Q^{0}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{null}\}$$

Once $q_i \in Q_{dj}$ cases are classified into one of the four defined groups, the purpose is to infer rules to update the discriminating function of each domain, which is stored in the knowledge base. The action of this rules will be:

- Modifying the cases belonging to Q^{+}_{dj} to the lowest extent.
- Determining the weights w_{pk} of the keywords (variables) of the domain taxonomy. These weights must be increased in order to correct the classification error produced in the cases of group Q^{*}_{dj}. These rules will operate on keywords $p_k \in Kd_j$ that are frequently present in queries $q_i \in Q^{*}_{dj}$, since it can be inferred that these predicting variables are more important to classify domains than what their associated weights really reflect. In other words, the current weight factors w_{pk} are low.
- Encouraging the incoming of new keywords into the domain taxonomy. This means including new predicting variables in the discriminating function of domain d_j. These rules will operate on keywords $p'_k \notin Kd_j$ that are frequently present in queries $q_i \in Q^{*}_{dj}$. In other words, it is inferred that these predicting variables are important to classify domains. However, if those words are also frequently present in queries answered by most of the remaining domains, these keywords would not be useful to distinguish among domains and thus they should not be incorporated.

Another possibility is that a domain presents many cases in which it answered in a negative form although appearing as better positioned in the ranking than the domain that actually provided a positive answer. This means that this domain taxonomy has words whose weights are too high when compared to their importance in the domain. Therefore, there should be a rule that diminishes the weights of these words.

With the aim of interpreting the relationships among variables, as example, we present three rules obtained by the mining process, which can be used to develop the system learning process:

Given $\mathbf{Q_{dj}}$, **to classify:**

$$Q^{*}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{positive} \wedge or(d_j,q_i) > 1\}$$
$$Q^{-}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{negative}\}.$$

Set

$p \in Kd_j$

n_P^*: frequence of p in queries of Q^*_{dj}.
n_P^-: frequence of p in queries of Q^-_{dj}.
α_1 and α_2: system parameters

IF $\#Q_{dj} > \alpha_1 \wedge \frac{n_P^*}{\#Q_{dj}} > \alpha_2 \wedge n_P^* >> n_P^-$

THEN p is a candidate for increasing its weight w_p

In the condition of this rule, we are saying that a word belonging to the domain taxonomy is a candidate for increasing its weight if:

a) more than α_1 cases are stored in Q_{dj} and

b) the number of times in which p is stated in queries of Q^*_{dj} is greater than α_2 and

c) n_p^* is much higher than n_p^-

Now, we present the rule for the incoming of new words into the domain taxonomy.

Given $\mathbf{Q_{dj}}$, **to classify**:

$Q^*_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{positive} \wedge or(d_j,q_i) > 1\}$
$Q^-_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{negative}\}$

Set
$p \notin Kd_j$
n_p^*: frequence of p in queries of Q^*_{dj}.
n_p^-: frequence of p in queries of Q^-_{dj}.
dn_p: number of domains in which p is stated or is a candidate for entering.
$D=\{d_1,d_2,...,d_n\}$ set of system domains

IF $\#Q_{dj} > \alpha_3 \wedge \frac{n_P^*}{\#Q_{dj}} > \alpha_4 \wedge n_p^* >> n_p^-$

THEN p is a candidate for entering Kd_j

p can enter Kd_j if it is not a candidate for entering the taxonomy of several domains and/or it does not belong to the taxonomy of several domains

$\frac{dn_p}{\#D} \cong 0$ Þ p can be used to discriminate the domains, to which it is a candidate, from the remaining domains. Then, p must enter the taxonomy of domains to which it is a candidate.

$\frac{dn_p}{\#D} \cong 1$ Þ p can not be used to discriminate, then it can be included as stopword.

In the first condition of this rule, we say that a word is a candidate for entering a Kd_j if:

a) more than α_3 cases are stored in Q_{dj} and

b) the proportion between the number of times in which p is stated in queries of Q^*_{dj} with respect to the number of cases stored in Q_{dj} is greater than α_4.

c) n_p^* much higher than n_p^-.

A word can enter Kd_j if the amount of domains in the system is much higher than the quantity of domains in which p is stated or is a candidate.

Set
d_R the domain that positively answer the query,
$Q^{+}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{positive} \wedge or(d_j,q_i) = 1\}$
$Q^{*}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{positive} \wedge or(d_j,q_i) > 1\}$
$Q^{-}_{dj} = \{q_i \in Q_{dj} / va(d_j,q_i) = \text{negative}\}$
$Q^{\wedge}_{dj} = \{q_i \in Q^{-}_{dj} / or(d_j,q_i) < or(d_R,q_i)\}$
$Q^{\wedge}_{dj}$ set of cases of domains that bring a negative answer to the query q_i and are best ranked than d_R
Set
$p \in Kd_j$.
n_P^{+}: frequence of p in queries of Q^{+}_{dj}.
n_P^{*}: frequence of p in queries of Q^{*}_{dj}.
$n_P^{\wedge}$: frequence of p in queries of $Q^{\wedge}_{dj}$.

IF $\#Q_{dj} > \alpha_5 \wedge \dfrac{n_P^{\wedge}}{\#Q_{dj}} > \alpha_6 \wedge n_P^{\wedge} >> n_P^{+} + n_P^{*}$

THEN p is a candidate for diminishing its weight

In the condition of this rule, we are saying that a word p is a candidate for diminishing its weight if:

a) more than α_5 cases are stored in Q_{dj} and

b) the proportion of the number of times in which p is stored in queries of $Q^{\wedge}_{dj}$ in respect to the number of stored cases Q_{dj} is greater than α_6 and

c) $n_p^{\wedge}$ is much greater than the number of times in which p is stated in queries with positive answer ($n_p^{+} + n_p^{*}$)

4. Conclusions

A dynamic DSS that works efficiently, i.e., that does not constantly interrupt users with information requirements they cannot satisfy, must be able to identify the relationship among the characteristics of consults and domains. By using data mining techniques it was possible to define a discriminating function to classify the system domains into two groups: those that can probably provide an answer to the information requirement made to the system, and those that cannot.

The system needs to learn from the errors it could make during its operation so that it can tend to diminish the number of consulted domains in each information requirement presented to the system. The use of data mining allowed to define the data structure that is convenient for analyzing the system operation results and according to that, designing a cases base to store the information associated with the quality of each performed search.

Moreover, the application of data mining to the cases base allowed the specification of rules to settle relationships among the stored cases with the aim of inferring possible causes of error in the domains classification. In this way, a learning mechanism was designed to update the knowledge base and thus improve the already made classification as regards the values assigned to the discriminating function.

References

Cabral, L., Caliusco, M., Nissio, H., Villarreal, M., Galli, P., Taverna,M. & Chiotti, O. (2000). "Use of Agents Technology to Support Distributed Decision Processes", in Proceedings of the *First World Conference on Production and Operations Management*, POM Sevilla 2000. Sevilla, Spain, August 2000.

Dingsoyr, T. (1998), "Integration of Data Mining and Case-Based Reasoning". Accessible online at: <http://www.idi.ntnu.no/~dingsoyr/diploma>

Jennings, N. (2001). "An Agent-Based Approach for Building Complex Software Systems". *Communications of the ACM*, 44 (4), 35-42.

Kachigan, S. K. (1991). "*Multivariable Statistical Analysis. A conceptual introduction*". New York: Radius Press.

Rodriguez, F., Ramos, C., & Henriques, P. (2000). "A case based reasoning framework to extract knowledge from data", in *Data Mining II*, N. Ebecken & C.A. Brebbia (Eds.) Southampton, Boston, WITpress.

Ruiz, C., Fisler, K. & Cole, Ch., "KDDRG – Data Mining for Software Engineering". Accessible online at: <http://www.cs.wpi.edu/~ruiz/KDDRG/dm_for_se.html >

Rus, D., Gray, R. & Kots, D. (1998). "Transportable Information Agents". In *Reading in Agents* ,Michael Huhns & Munidar Singh book,.

Russell, S.J. and Norvig, P. (1995). "*Artificial Intelligence. A Modern Approach*". Prentice Hall, Englewood Cliffs, NJ.

Shen, W.,& Norrie, D.H. (1999). "Agent-Based Systems for Intelligent Manufacturing: A Sate-of-the-Art Survey". *Knowledge and Information Systems*, 1 (2), 129-156.

Stegmayer, G., Taverna, M.L., Chiotti, O., & Galli, M.R. (2001). "The Agent Routering Process of Dynamic Distributed Decision Support System". *Journal of Computer Science & Technology*, 2 (5), 30-43. Accessible online at: <http://journal.info.unlp.edu.ar>