Cooperative answering of queries based on hierarchical decision attributes

Zbigniew W. Ras[1,2], Agnieszka Dardzinska[3] , Xin Zhang[1]
[1] *Univ. of North Carolina, Dept. of Computer Science, Charlotte, N.C. 28223, USA*
[2] *Polish-Japanese Institute of Information Technology, 02-008 Warsaw, Poland*
[3] *Bialystok Technical Univ., Dept. of Computer Science, 15-351 Bialystok, Poland*

Abstract. This paper considers decision systems (see Pawlak, 1981) with decision attributes which are hierarchical. Atomic queries are built only from values of decision attributes. Queries are constructed from atomic queries the same way as we construct terms in logic using functors {+, *,¬ }. Negation symbol "¬" is only used on the atomic level. Queries are approximated by terms built from values of classification attributes. We only consider rule-based classifiers as the approximation tool for queries. When a user query fails, then the cooperative module of the query answering system (QAS) constructs its smallest generalization which does not fail and which is approximated by rules of the highest confidence discovered by the classifier. Two interpretations of queries are proposed: user-based and system-based. They are used to introduce the precision and recall of QAS. The implementation of QAS follows system-based interpretation. Automatic indexing of music by instruments and their types is an example of the application area for the proposed approach.

Keywords: information systems, knowledge discovery, cooperative query answering, music information retrieval.

1. INTRODUCTION

Responses to queries posed by a user of a database do not always contain the information desired. Database answers to a query, although they may be logically correct, can sometimes be misleading. Research in the area of cooperative answering for databases and deductive databases rectifies these problems. Classical approach proposed in is based on a cooperative method called relaxation for expanding a database and related to it queries [2], [3], [4], [7]. The relaxation method expands the scope of a query by relaxing the constraints implicit in the query. This allows the database to return answers related to the original query as well as the literal answers themselves. These additional answers may be of interest to the user.

Music information retrieval [5], [8], [9], [10] is one of the application areas for cooperative query answering. Multi-hierarchical decision system in [8] is a database of about 1,000,000 musical instrument sounds, each one represented as a vector of approximately 1,100 features. Each instrument sound is labeled by a corresponding instrument. These labels are used to define one of the decision attributes. There are many ways to categorize music instruments, such as by playing methods, by instrument type, or by other generalization concepts. Any categorization process can be represented as a hierarchical schema which can be used by a cooperative query answering system to handle failing queries. By definition, a cooperative system is relaxing a failing query with a goal to find its smallest generalization which will not fail. Two different hierarchical schemas [8] have been used as models of a decision attribute: Hornbostel-Sachs classification of musical instruments and classification of musical instruments by articulation.

Each hierarchical classification represents a unique decision attribute, in a database of music instrument sounds, leading to a construction of a new classifier and the same to a different system for automatic indexing of music by instruments and their types [8], [9].

Names of instruments and their generalizations (Hornbostel-Sachs classification, generalization by articulation) are used to construct atomic queries of a query language built for retrieving musical objects from MIR Database (see http://www.mir.uncc.edu). When query fails, the cooperative strategy may tries to find its lowest generalization which does not fail. Clearly, by having a variety of different hierarchical structures available for modeling decision attribute we have better chance not only to succeed but also to succeed with a possibly smallest generalization of a query.

This paper introduces a new theoretical framework for modeling a multi-hierarchical decision system S and its corresponding query language which is built from values of decision attributes in S. Standard interpretation and classifier-based interpretation of queries are introduced and used to model the quality (precision, recall) of a query answering system.

2. DECISION-HIERARCHICAL INFORMATION SYSTEM

In this section we introduce the notion of a multi-hierarchical decision system *S* and the query language built from atomic expressions reduced only to values of decision attributes. Classifier-based semantics and standard semantics of queries in *S* are proposed. The set of objects *X* in S forms the interpretation domain for both semantics. Standard semantics identifies all objects in X which should be retrieved by a query. Classifier-based semantics gives weighted set of objects which are retrieved be a query. The notion of precision and recall of the query answering system (QAS) in the proposed setting is introduced. Only rule-based classifiers are used to define the classifier-based semantics. By improving their confidence and support we improve the precision and recall of QAS.

Definition 1.1

By a multi-hierarchical decision system we mean a triple S = $(X, A \cup \{d[1],d[2],..,d[k]\}, V)$, where X is a nonempty, finite set of objects, A is a nonempty finite set of classification attributes, $\{d[1],d[2],..,d[k]\}$ is a set of hierarchical decision attributes and

$V = \bigcup\{V_a : a \in A \cup \{d[1],d[2],..,d[k]\}\}$ is a set of their values.

We assume that:

- V_a, V_b are disjoint for any $a, b \in A \cup \{d[1],d[2],..,d[k]\}$, such that $a \neq b$,

  $a : X \rightarrow V_a$ is a partial function for every $a \in A \cup \{d[1],d[2],..,d[k]\}$.

Definition 1.2

By a set of decision queries (d-queries) for S we mean a least set T_D such that:

- $0, 1 \in T_D$,
- if $w \in \bigcup\{V_a : a \in \{d[1],d[2],..,d[k]\}\}$, then w, $\sim w \in T_D$,
- *if* $t1, t2 \in T_D$, then $(t1 + t2), (t1 * t2) \in T_D$.

Definition 1.3

Decision query t is called simple if t = t1∗t2∗…∗tn and

$(\forall j \in \{1,2,\ldots,n\})[(tj \in \bigcup\{V_a : a \in\{d[1],d[2],..,d[k]\}\}) \vee (tj = {\sim}w \;\wedge\; w\in \bigcup\{V_a : a \in\{d[1], d[2],.., d[k]\}\})]$.

Definition 1.4

By a set of classification terms (c-terms) for S we mean a least set T_C such that:

- $0, 1 \in T_C$,
- if $w \in \bigcup\{V_a : a \in A\}$, then w, ${\sim}w \in T_C$,
- *if* $t1, t2 \in T_C$, then $(t1 + t2), (t1 * t2) \in T_C$.

Definition 1.5

Classification term t is called simple if t = t1∗t2∗…∗tn and

$(\forall j \in \{1,2,\ldots,n\})[(tj \in \bigcup\{V_a : a \in A\})] \vee (tj = {\sim}w \;\wedge\; w\in \bigcup\{V_a : a \in A\})]$.

Definition 1.6

By a classification rule we mean any expression of the form $[t_1 \rightarrow t_2]$, where t_1 is a simple classification term and t_2 is a simple decision query.

Definition 1.7

Semantics M_S of c-terms in S = $(X, A\cup\{d[1],d[2],..,d[k]\}, V)$, is defined in a standard way as follows:

- $M_S(0) = 0$, $M_S(1) = X$,
- $M_S(w) = \{x \in X : w = a(x)\}$ for any $w \in V_a$, $a \in A$,
- $M_S({\sim}w) = \{x \in X : (\exists v \in V_a)[v = a(x) \;\&\; v \neq w]\}$ for any $w \in V_a$, $a \in A$,
- if $t1$, $t2$ are terms, then

  $M_S(t1 + t2) = M_S(t1) \cup M_S(t2)$,

  $M_S(t1 * t2) = M_S(t1) \cap M_S(t2)$.

Let us introduce the notation we use in this paper for values of decision attributes. Assume that $d[i]$ is a hierarchical decision attribute which is also interpreted as its first granularity level. The set $\{d[i,1], d[i,2], d[i,3],\ldots\}$ represents the values of attribute $d[i]$ at its second granularity level. The set $\{d[i,1,1], d[i,1,2],\ldots, d[i,1,n_i]\}$ represents the values of attribute d at its third granularity level, right below the node $d[i,1]$. We assume here that the value $d[i,1]$ can be refined to any value from $\{d[i,1,1], d[i,1,2],\ldots,d[i,1,n_i]\}$, if necessary. Similarly, the set $\{d[i,3,1,3,1], d[i,3,1,3,2], d[i,3,1,3,3], d[i,3,1,3,4]\}$ represents the values of attribute d at its forth granularity level which are finer than the value $d[i,3,1,3]$.

Now, let us assume that a rule-based classifier (for instance one of the modules in systems RSES or WEKA) was used to extract rules describing simple decision queries in S. We denote this classifier by ***RC***. The definition of semantics of c-terms does not depend on a classifier but the definition of semantics M_S of d-queries is a classifier dependent.

Definition 1.8

Classifier-based semantics M_S of d-queries in $S = (X, A\cup\{d[1],d[2],..,d[k]\}, V)$, is defined as follows:

- if t is a simple d-query in S and $\{r_j = [t_j \rightarrow t]: j \in J_t\}$ is a set of all rules defining t which are extracted from S by classifier ***RC***, then

$M_S(t) = \{(x,p_x): (\exists j \in J_t)(x \in M_S(t_j)[p_x =$
$\Sigma\{conf(j)\cdot sup(j): x \in M_S(t_j) \;\&\; j \in J_t\}/\Sigma\{sup(j): x \in M_S(t_j) \;\&\; j \in J_t\}]\}$, where $conf(j)$, $sup(j)$ denote the confidence and the support of $[t_j \to t]$, correspondingly.

Definition 1.9

Attribute value $d[j_1, j_2,\ldots j_n]$ in S = (X, A∪{d[1],d[2],..,d[k]}, V) is dependent on $d[i_1, i_2,\ldots, i_k]$ in S, if one of the following conditions hold:

1) $n \le k \;\&\; (\forall m \le n)[i_m = j_m]$,
2) $n > k \;\&\; (\forall m \le k)[i_m = j_m]$.

Otherwise, $d[j_1, j_2,\ldots j_n]$ is called independent from $d[i_1, i_2,\ldots, i_k]$ in S.

Example 1.1

The attribute value $d[\mathbf{2,3,1,2}]$ is dependent on the attribute value $d[\mathbf{2,3,1,2},5,3]$. Also, $d[\mathbf{2,3,1,2,5,3},2,4]$ is dependent on $d[\mathbf{2,3,1,2,5,3}]$.

Definition 1.10

Let S = $(X, A\cup\{d[1],d[2],..,d[k]\}, V)$, $w \in V_{d[i]}$, and $IV_{d[i]}$ be the set of all attribute values in $V_{d[i]}$ which are independent from w.

Standard semantics N_S of d-queries in S is defined as follows:

- $N_S(0) = 0$, $N_S(1) = X$,
- if $w \in V_{d[i]}$, then $N_S(w) = \{x \in X : d[i](x)=w\}$, *for any* $1 \le i \le k$
- if $w \in V_{d[i]}$, then $N_S(\sim w) = \{x \in X : (\exists v \in IV_{d[i]})[d[i](x)=v]\}$, *for any* $1 \le i \le k$
- if $t1$, $t2$ are terms, then

  $N_S(t1 + t2) = N_S(t1) \cup N_S(t2)$,
  $N_S(t1 * t2) = N_S(t1) \cap N_S(t2)$.

Definition 1.11

Let $S = (X, A\cup\{d[1],d[2],..,d[k]\}, V)$, t is a d-query in S, $N_S(t)$ is its meaning under standard semantics, and $M_S(t)$ is its meaning under classifier-based semantics. Assume that $N_S(t) = X_1 \cup Y_1$, where $X_1 = \{x_i, i \in I_1\}$, $Y_1 = \{y_i , i \in I_2\}$. Assume also that $M_S(t) = \{(x_i, p_i): i \in I_1\} \cup \{(z_i, q_i): i \in I_3\}$ and $\{y_i , i \in I_2\} \cap \{z_i , i \in I_3\} = \varnothing$.

By precision of a classifier-based semantics M_S on a d-query t, we mean

$$rec(M_S, t) = [\Sigma\{p_i : i \in I_1\} + \Sigma\{(1 - q_i) : i \in I_3\}]/[card(I_1) + card(I_3)].$$

By recall of a classifier-based semantics M_S on a d-query t, we mean

$$Rec(M_S, t) = [\Sigma\{p_i : i \in I_1\}]/[card(I_1) + card(I_3)].$$

Example 1.2

Assume that $N_S(t) = \{x_1, x_2, x_3, x_4\}$, $M_S(t) = \{(x_1, p_1), (x_2, p_2), (x_5, p_5), (x_6, p_6)\}$.
Then:

$\text{Prec}(M_S, t) = [p_1 + p_2 + (1-p_5) + (1 - p_6)]/4$,
$\text{Rec}(M_S, t) = [p_1 + p_2]/4$.

Example 1.3

Assume that the decision-hierarchical information system $S =(\{x1,x2,x3,x4\}, \{a,b\}\cup\{c,d\},V)$ is represented by the table below. The set $\{a,b\}$ contains classification attributes. The set $\{c,d\}$ contains decision attributes.

X	a	b	c	d
x1	a[1]	b[2]	c[1]	d[3]
x2	a[1]	b[1]	c[1]	d[3,1]
x3	a[1]	b[2]	c[2,2]	d[1]
x4	a[2]	b[2]	c[2]	d[1]

Table 1. Multi-hierarchical decision system S

Let us use LERS (Chmielewski, Grzymala-Busse, 1993) module implemented in RSES for rules extraction. We assume that the threshold for minimum support = 1, and the threshold for minimum confidence = 1/3. We get:

$r1 = [a[1] \rightarrow c[1]]$, with conf(r1)= 2/3, sup(r1)=2
$r2 = [a[2] \rightarrow c[2]]$, with conf(r2)= 1, sup(r2)=1
$r3 = [b[2] \rightarrow c[2]]$, with conf(r3)= 2/3, sup(r3)=2
$r4 = [b[1] \rightarrow c[1]]$, with conf(r4)= 1, sup(r4)=1
$r5 = [a[1] \rightarrow c[2,2]]$, with conf(r5)= 1/3, sup(r5)=1
$r6 = [b[2] \rightarrow c[2,2]]$, with conf(r6)= 1/3, sup(r6)=1
$r7= [b[2] \rightarrow c[1]]$, with conf(r7) = 1/3, sup(r7)=1
$r8 = [a[1] \cdot b[2] \rightarrow c[1]]$, with conf(r8)= 1/2, sup(r8)=1
$r9 = [a[1] \cdot b[2] \rightarrow c[2,2]]$, with conf(r9)= 1/2, sup(r9)=1

Let us notice that the rule $r10 = [a[1] \rightarrow c[2]]$ is not extracted because its confidence and support is the same as $r5$ which is a more precise rule than $r10$.

Now, we are ready to compute the classifier-based semantics of d-queries *c[1], c[2], c[2,2].* For *c[1]* and *x1* we use rules *r1*, *r8, r7* since only these three rules support *x1*. For *c[1]* and *x2* we use rules *r1*, *r4*. For *c[2]* and *x3* we use rules *r5, r6, r9*. For *c[2]* and *x4* we use *r2, r3*. For *c[2,2]* and *x3* we use *r5, r6, r9*. For $\neg c[1]$ and *x3* we use rules *r5, r6, r9*. For $\neg c[1]$ and *x4* we use rules *r2, r3*.

$$M_S(c[1]) = \{(x1, (2/3 \cdot 2 + \tfrac{1}{2} \cdot 1 + 1/3 \cdot 1)/(2 + 1 + 1)),$$
$$(x2, (2/3 \cdot 2 + 1 \cdot 1)/(2 + 1))\} = \{(x1, 13/24), (x2, 7/9)\},$$
$$M_S(c[2]) = \{(x3, (1/3 \cdot 1 + 1/3 \cdot 1 + \tfrac{1}{2} \cdot 1)/(1 + 1 + 1)),$$
$$(x4, (1 \cdot 1 + 2/3 \cdot 2)/(1 + 2))\} = \{(x3, 7/18), (x4, 7/9)\},$$
$$M_S(c[2,2]) = \{(x3, (1/3 \cdot 1 + 1/3 \cdot 1 + \tfrac{1}{2} \cdot 1)/(1 + 1 + 1))\} = \{(x3, 7/18)\}.$$

$$M_S(\neg c[1]) = M_S(c[2]),\ M_S(\neg c[2,2]) = M_S(c[1]), M_S(\neg c[2]) = M_S(c[1]).$$

Standard semantics N_S of the above d-queries will retrieve:

$$N_S(c[1]) = \{(x1,1), (x2,1)\},\ N_S(c[2]) = \{(x3,1), (x4,1)\}, N_S(c[2,2])= \{(x3,1)\}.$$
$$N_S(\neg c[1]) = \{(x3,1), (x4,1)\}, N_S(\neg c[2,2]) =\{(x1,1), (x2,1)\}= N_S(\neg c[2]).$$

Now, we compute the precision and recall of M_S on d-queries *c[1], c[2], c[2,2],* $\neg c[1]$, $\neg c[2]$, and $\neg c[2,2]$.

$$Prec(M_S, c[1]) = [13/24 + 7/9]/2 = 95/144 = 0.66,\ Rec(M_S, c[1]) = 0.66,$$
$$Prec(M_S, c[2]) = [7/18 + 7/9]/2 = 21/36 = 7/12 = 0.58, Rec(M_S, c[2]) = 0.58$$
$$Prec(M_S, c[2,2]) = 7/18,\ Rec(M_S, c[2,2]) = 7/18 = 0.39$$
$$Prec(M_S, \neg c[1]) = Prec(M_S, c[2]) = 0.58, Rec(M_S, \neg c[1])= 0.58$$

$Prec(M_S, \neg\, c[2]) = Prec(M_S, c[1]) = 0.66, Rec(MS, \neg\, c[2]) = 0.66$
$Prec(M_S, \neg\, c[2,2]) = Prec(M_S, c[1]) = 0.66, Rec(M_S, \neg\, c[2,2]) = 0.66$

3. COOPERATIVE QUERY ANSWERING

There are cases when classical Query Answering Systems (QAS) fail to return any answer to a submitted d-query *q* but still a satisfactory answer can be found. For instance, let us assume that in a multi-hierarchical decision system $S = (X, A \cup \{d[1],d[2],..,d[k]\}, V)$ there is no single object which description matches the query *q*. Assuming that a distance measure between objects in *S* is defined, then by generalizing *q*, we may identify objects in *S* which descriptions are nearest to the description of *q*. This problem is similar to the problem when the granularity of an attribute value used in a query *q* is finer than the granularity of the corresponding attribute used in S. By replacing such attribute values in *q* by more general values used in *S*, we retrieve objects from *S* which may satisfy *q*.

Definition 2.1

The distance δ_S between two attribute values $d[j_1, j_2,\ldots j_n]$, $d[i_1, i_2,\ldots, i_m]$ in
$S = (X, A \cup \{d[1],d[2],..,d[k]\}, V)$, where $j_1 = i_1$, $p \geq 1$, is defined as follows:

1) if $[j_1, j_2,\ldots j_p] = [i_1, i_2,\ldots, i_p]$ and $j_{p+1} \neq i_{p+1}$, then $\delta_S[d[j_1, j_2,\ldots j_n], d[i_1, i_2,\ldots, i_m]] = 1/[2^{p-1}]$
2) *if* $n \leq m$ and $[j_1, j_2,\ldots j_n] = [i_1, i_2,\ldots, i_n]$, then $\delta_S[d[j_1, j_2,\ldots j_n], d[i_1, i_2,\ldots, i_m]] = 1/[2^n]$

The second condition, in the above definition, represents the average case between the best and the worth case.

Example 2.1

Following the above definition of the distance measure, we get:

1) $\delta_S[d[2,3,2,4], d[2,3,2,5,1]] = ¼$
2) $\delta_S[d[2,3,2,4], d[2,3,2]] = 1/8$

Let us assume that $q = q(a[3,1,3,2], b[1], c[2])$ is a d-query which is submitted to *S*. The notation $q(a[3,1,3,2], b[1], c[2])$ means that *q* is built from $a[3,1,3,2]$, $b[1]$, $c[2]$ which are the atomic attribute values in *S*. Additionally, we assume that attribute *a* is not only hierarchical but also it is ordered. It basically means that the difference between the values $a[3,1,3,2]$ and $a[3,1,3,3]$ is smaller than between the values $a[3,1,3,2]$ and $a[3,1,3,4]$. Also, the difference between any two elements in $\{a[3,1,3,1], a[3,1,3,2], a[3,1,3,3], a[3,1,3,4]\}$ is smaller than between $a[3,1,3]$ and $a[3,1,2]$.

Now, we outline a possible strategy which *QAS* can follow to solve *q*. Clearly, the best solution for answering *q* is to identify objects in *S* which precisely match the d-query submitted by user. If it fails, we should try to identify objects which match d-query $q(a[3,1,3], b[1], c[2])$. If we succeed, then we try d-queries $q(a[3,1,3,1], b[1], c[2])$ and $q(a[3,1,3,3], b[1], c[2])$. If we fail, then we should succeed with $q(a[3,1,3,4], b[1], c[2])$. If we fail with $q(a[3,1,3], b[1], c[2])$, then we try $q(a[3,1], b[1], c[2])$ and so on.

To present this cooperative strategy in a more precise way, we use an example and start with a very simple dataset. Namely, we assume that *S* has 4 decision attributes which belong to the set $\{a, b, c, d\}$. System *S* contains only four objects listed below

X	e	f	g	.....	.....	a	b	c	d
x1	e[1]	f[1]	......	.....	.....	a[1]	b[2]	c[1,1]	d[3]
x2	e[2]	f[1]	......	.....	.....	a[1,1]	b[2,1]	c[1,1,1]	d[3,1,2]
x3	e[2]	f[1]	......	.....	.....	a[1,1,1]	b[2,2,1]	c[2,2]	d[1]
x4	e[1]	f[2]	......	.....	.....	a[2]	b[2,2]	c[1,1]	d[1,1]

Table 2. Multi-hierarchical decision system S

Now, we assume that d-query $q = a[1,2]*b[2]*c[1,1] *d[3,1,1]$ is submitted to the decision system S (see Table 2). Clearly, q fails in S.

Jointly with q, also a threshold value for a minimum support can be supplied as a part of a d-query. This threshold gives the minimal number of objects that need to be returned as an answer to q. When the query answering system (QAS) fails to answer q, the nearest objects satisfying q have to be identified.

The algorithm for finding these objects follows the following steps:

If *QAS* fails to identify sufficient number of objects satisfying q in S, then the generalization process starts. We can generalize either attribute a or d. Since the value $d[3,1,2]$ has lower granularity level than $a[1,1]$, then we generalize $d[3,1,2]$ getting a new query $q1 = a[1,2]*b[2]*c[1,1] *d[3,1]$. But $q1$ still fails in S. Now, we generalize $a[1,1]$ getting a new query $q2 = a[1]*b[2]*c[1,1] *d[3,1]$. Objects $x1$, $x2$ are the only objects in S which support $q2$.

If the user is only interested in one object satisfying the query q, then we need to identify which object in $\{x1, x2\}$ has a distance closer to q.

Clearly,
$\delta_S[q, x1] = \delta_S[[a[1,2], b[2], c[1,1], d[3,1,1]], [a[1], b[2],c[1,1], d[3]]] =$
¼+0+0+1/4=1/2,
$\delta_S[q, x2] = \delta_S[[a[1,2], b[2], c[1,1], d[3,1,1]], [a[1,1],b[2,1],c[1,1,1],d[3,1,2]]] =$
1/4+1/4+1/8+1/8 = ¾, which means $x1$ is the winning object.

Let us notice that the cooperative strategy only identifies objects satisfying d-queries and the same objects to be returned by the query answering system to the user. The confidence assigned to these objects depends on the classifier and it is calculated following the strategy described in Section 1. The next section shows how to evaluate and chose the best classifier for a multi-hierarchical decision system.

4. COMPARISON OF CLASSIFIERS FOR MULTI-HIERARCHICAL DECISION SYSTEMS

Let us assume that $S = (X, A \cup \{d\}, V)$ is a hierarchical decision system, where d is a hierarchical attribute. For the simplicity of this presentation, we consider information systems with only one decision attribute. Additionally, we assume that $d_{[i1,\ \ldots,\ ik]}$ (where $1 \le i_j \le m_j, j = 1,\ldots k$) is a child of $d_{[i1,\ \ldots,\ ik-1]}$ for any $1 \le i_k \le m_k$. Clearly, attribute d has

$\sum\{m_1 \cdot m_2 \cdot ... \cdot m_j : 1 \le j \le k\}$ values, where $m_1 \cdot m_2 \cdot ... \cdot m_j$ shows the upper bound for the number of values at the level j of d. By $p([i_1,\ldots, i_k])$ we denote a path $(d, d_{[i1]}, d_{[i1,i2]}, d_{[i1,i2,i3]},\ldots, d_{[i1,\ldots,ik-1]}, d[_{i1,\ldots,ik]})$ leading from the root of the hierarchical attribute d to its descendant $d_{[i1,\ldots,ik]}$.

Let us assume that R_j is a set of classification rules extracted from S, representing a part of a rule-based classifier $R = \cup\{R_j : 1 \le j \le k\}$, and describing all values of d at level j. The quality of a classifier at level j of attribute d can be checked by calculating $Q(R_j) = \dfrac{\sum\{\sup(r) \cdot conf(r) : r \in R_j\}}{\sum\{\sup(r : r \in R_j)\}}$, where $sup(r)$ is the support of the rule r in S and $conf(r)$ is its confidence. Then, the quality of the rule-based classifier R can be checked by calculating $Q(\cup\{R_j : 1 \le j \le k\}) = \dfrac{\sum\{Q(R_j) : 1 \le j \le k\}}{k}$.

The quality of a tree-based classifier can be given by calculating its quality for every node of a hierarchical decision attribute d. Let us take a node $d_{[i1,\ldots,ik]}$ and the path $p([i_1,\ldots, i_k])$ leading to that node from the root of d. There is a set of classification rules $R_{[i1,\ldots,im]}$, uniquely defined by the tree-based classifier, assigned to a node $d_{[i1,\ldots,im]}$ of a path $p([i_1,\ldots, i_k])$, for every $1 \le m \le k$. Now, we define $Q(R_{[i1,\ldots,im]})$ as $\dfrac{\sum\{\sup(r) \cdot conf(r) : r \in R_j\}}{\sum\{\sup(r : r \in R_j)\}}$. Then, the quality of a tree-based classifier for a node $d_{[i1,\ldots,im]}$ of the decision attribute d can be checked by calculating $Q(d_{[i1,\ldots,im]}) = \prod\{Q(R_{[i1,\ldots,ij]}) : 1 \le j \le m\}$. Learning values of a decision attribute at different generalization levels is extremely important in the process of handling failing queries.

5. CONCLUSION

We have introduced the notion of a system-based semantics and user-based semantics of queries. User-based semantics is associated with the indexing of objects done by a user which is time consuming and unrealistic for very large sets of data. System-based semantics is associated with automatic indexing of objects in *X* which strictly depends on the support and confidence of classifiers and depends on the precision and recall of a query answering system. The quality of classifiers can be improved by a proper enlargement of the set *X* and the set of describing them features which differentiate the real-life objects from the same semantic domain as *X* in a better way [8], [9], [10]. The quality of a query answering system (QAS) can be improved by its cooperativeness. Both precision and recall of QAS is getting increased if no-answer queries are replaced by generalized queries which are answered by QAS on a higher granularity level than the initial level of queries submitted by users.

6. ACKNOWLEDGEMENTS

This research was partially supported by the National Science Foundation under grant IIS-0414815.

REFERENCES

[1] Chmielewski, M.R., Grzymala-Busse, J.W., Peterson, N.W., *The rule induction system LERS - a version for personal computers*, in Foundations of Computing and Decision Sciences, Vol. 18, No. 3-4, Institute of Computing Science, Technical University of Poznan, Poland, 1993, 181-212

[2] Chu, W., Yang, H., Chiang, K., Minock, M., Chow, G., Larson, C., *Cobase: A scalable and extensible cooperative information system*, in Journal of Intelligent Information Systems, Vol. 6, No. 2/3, 1996, 223-259

[3] Gaasterland, T., *Cooperative answering through controlled query relaxation*, in IEEE Expert, Vol. 12, No. 5, 1997, 48-59

[4] Godfrey, P., *Minimization in cooperative response to failing database queries*, in International Journal of Cooperative Information Systems, Vol. 6, No. 2, 1993, 95-149

[5] Lewis, R., Zhang, X., Ras, Z.W., *Knowledge Discovery Based Identification of Musical Pitches and Instruments in Polyphonic Sounds*, in the Special Issue on “Soft Computing Applications”, Journal of Engineering Applications of Artificial Intelligence, Elsevier, Vol. 20, No. 5, 2007, 637-645

[6] Pawlak, Z., *Information systems - theoretical foundations*, in Information Systems Journal, Vol. 6, 1981, 205-218

[7] Ras, Z.W., Dardzinska, A., *Solving Failing Queries through Cooperation and Collaboration*, Special Issue on Web Resources Access, (Editor: M.-S. Hacid), in World Wide Web Journal, Springer, Vol. 9, No. 2, 2006, 173-186

[8] Ras, Z.W., Zhang, X., Lewis, R., *MIRAI: Multi-hierarchical, FS-tree based Music Information Retrieval System*, (Invited Paper), Proceedings of RSEISP 2007, M. Kryszkiewicz et al. (Eds), LNAI, Vol. 4585, Springer, 2007, 80-89

[9] Zhang, X., Ras, Z.W., *Analysis of Sound Features for Music Timbre Recognition*, (Invited Paper), in Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE 2007), IEEE Computer Society, April 26-28, 2007, in Seoul, South Korea, 3-8

[10] Zhang, X., Ras, Z.W., *Isolation by Harmonic Peak Partition for Music Instrument Recognition,* in the Special Issue on Knowledge Discovery, Fundamenta Informaticae Journal, IOS Press, Vol. 78, No. 4, 2007, 613-628