International Journal of Grid and Distributed Computing
Vol. 4, No. 1, March 2011

Searching the Best Collaborating Agent for Query Answering Systems

Agnieszka Dardzinska
Bialystok University of Technology
Department of Mechanics and Applied Computer Science
ul. Wiejska 4C, 15-351 Bialystok Poland
a.dardzinska@pb.edu.pl

Abstract

In this paper we assume a group of collaborating agents where each agent is defined as an Information System. The system is coupled with a Query Answering System and a knowledge base which is initially empty. Information Systems can be incomplete. A user issues a query to the local database S in search for objects in information system that match a desired description. In case when some components of the description are missing in S, the query cannot be answered. The client then has to contact other agents to get some definitions of missing parts of the query. Our goal is to find optimal agent for client, where by optimal we mean an agent of maximal precision and recall. From all the agents we have, we choose the agent with the minimal value of distance, which corresponds to the closest agent to client. The definition of distance d discovered at the closest agent to the client and stored in KB of the client will guarantee that Query Answering System connected with the client has maximal precision and recall in group of agents having attribute d.

Keywords: *Information system, knowledge base, chase, agent, query answering system*

1. Introduction

We assume that there is a group of collaborating agents where each agent is defined as an Information System $S = (X, A, V)$, where X is a set of objects, A - a set of attributes and V - a set of values of attributes. The system S is coupled with a Query Answering System (*QAS*) and a knowledge base (*KB*) which is initially empty. Information Systems can be incomplete, where by incompleteness we mean a property which allows using attribute values with corresponding to them weights, as a value of an attribute ((a_i, p_i)). We assume that the sum of these weights p_i for each object has to be equal 1. The definition of an information system of type λ given in this paper was initially proposed in [9]. The type λ was introduced with a purpose to monitor the weights assigned to values of attributes by Chase algorithm. If a weight is less than λ, then the corresponding attribute value is ruled out as a possible value and weights assigned to the remaining attribute values are equally adjusted so its sum is equal again to one. Semantic inconsistencies are due to different interpretations of attributes and their values among sites (for instance one site can interpret the concept "tall" or "happy" differently than the other one). Different interpretations are also implied by the fact that each site may differently handle null values. Null value replacement by a value suggested either by statistical or some rule-based methods is quite common before a query is answered by QAS.

Also ontologies ([5], [6], [11], [12], [13], [1], [2], [14], [4]) are widely used as a part of semantical bridge between agents built independently so they can collaborate and understand each other. In [8], the notion of the optimal rough semantics and a method of its construction was proposed. The rough semantics can be used to model and nicely handle semantic inconsistencies among sites due to different interpretations of incomplete values. As the result of collaborations among agents, a knowledge base of any agent is updated and it contains rules extracted from information systems representing other agents. Although the names of attributes can be the same among information systems, their granularity levels may differ. As the result of these differences, the knowledge base has to satisfy certain properties in order to be used by Chase. Also, the semantic differences between agents may influence the precision and recall of a query answering system. We will show that it is wise to use the knowledge obtained from agents which are semantically close to the client agent when solving a query. This way, the precision and recall is getting improved.

2. Query Processing with Data

In real life, data are often collected and stored in information systems residing at many different locations, built independently, instead of collecting and storing them at a single location. In this case we talk about distributed (autonomous) information systems or about agents. Assume that user submits a query to one of the agents (called a client), which cannot be answered because some of the attributes used in a query do not exist in the information system representing the client site.
In such case, the client has to ask other agents for definitions of these unknown attributes. All these new definitions are stored then in the knowledge base of a client and can be used for answering the queries.

Example 1.

Let us assume, we have we have three distributed information systems S, S1, and S2, as in Table 1, Table 2 and Table 3 respectively.

Table 1. Information System S

X	a	d	e	f
x1	a1	d1	e1	f2
x2	a2	d1	e2	f1
x3	a1	d1	e1	f2
x4	a2	d2	e2	f2
x5	a1	d2	e2	f2
x6	a2	d2	e1	f1

Table 2. Information System S1

Y	a	b	c	d	e
y1	a1	b1	c2	d1	e1
y2	a2	b2	c1	d2	e2
y3	a2	b2	c2	d1	e1
y4	a1	b1	c2	d1	e1

y5	a1	b1	c1	d2	e2

Table 3. Information System S2

Z	B	c	d	f	g
z1	b1	c2	d2	f2	g2
z2	b1	c1	d2	f1	g1
z3	b1	c2	d1	f2	g1
z4	b2	c2	d1	f2	g3
z5	b2	c1	d1	f1	g3

Our goal is to find all objects in S satisfying the query $q = a1 * b1 * c2$.
In this example, attribute values $b1$ and $c2$ are not in the domain V of the client S. Therefore the system has to communicate with other information systems (in this case with $S1$ and $S2$) to receive the definitions for values $b1$ and $c2$ in terms of known values in system S. Systems S and $S1$ have two common attributes d and e. Systems S and $S2$ also have two common attributes d and f. In this case system S sends a request for system $S1$ to find the definition of $b1*c2$ in terms of attributes d and e, as $b1*c2$ is not known for agent S. The same system also sends a request for system $S2$ for finding a definition of $b1*c2$ in terms of attributes d and f.

For system $S1$:

$b1 = \{y1, y4, y5\}$
$c2 = \{y1, y3, y4\}$
$d1 = \{y1, y2\}$
$d2 = \{y2, y4, y5\}$
$e1 = \{y1, y3, y4\}$
$e2 = \{y2, y5\}$

None of the sets of objects describing $b1$ and $c2$ is included in sets of objects describing attributes d and e. Therefore we take a pair $b1 * c2 = \{y1, y4\}$ and this set of objects is included in set $e1$. The part of our query $b1 * c2$ can be easily replaced by $e1$, so the query in system S can have new form: $q = a1 * e1$. Objects satisfying such query are $x1$, $x3$.

For system $S2$:

$b1 = \{z1, z2, z3\}$
$c2 = \{z1, z3, z4\}$
$d1 = \{z3, z4, z5\}$
$d2 = \{z1, z2\}$
$f1 = \{z2, z5\}$
$f2 = \{z1, z3, z4\}$

None of the sets of objects describing $b1$ and $c2$ are included in sets of objects describing attributes d and f. Therefore we take into consideration a pair $b1 * c2 = \{x1, x3\}$ and this set of objects is included in set $f2$. In such case the part of the query $b1 * c2$ can be replaced by $f2$, so the query in system S can have form: $q = a1 * f2$. Objects satisfying this query are $x1$, $x3$, $x5$. The problem is: which answer is better for client, is this the answer received from agent $S1$ or agent $S2$?

3. Query Processing with Incomplete Data

In many fields, such as medical, banking, music, similar databases are kept at many sites. Some attributes may be missing in one database, while they occur in others. Missing attributes lead to problems. A user may issue a query to the local database S in search for objects in information system that match a desired description, only to realize that one component $a1$ of that description is missing in S so that the query cannot be answered. The definition of a1 may be extracted from agents at other sites e.g. $S1$, $S2$ and used to identify objects in S having property $a1$ (under the assumption that agents work with the same or similar semantic).

Example 2.

Let us assume, we have two distributed information systems S, $S1$ as in Table 4, Table 5.

Table 4. Information System S

X	B	c	d
x1	b1	c1	d2
x2	b1	c2	d2
x3	b1	c2	d1
x4	b2	c2	d1
x5	b1	c2	d1
x6	b2	c1	d2

Table 5. Information System S1

Y	A	b	c	e	f
y1	a1	b2	c2	e1	f1
y2	a2	b1	c1	e2	f1
y3	a2	b2	c2	e1	f2
y4	a1	b1	c2	e1	f1
y5	a1	b1	c2	e2	f2

Our goal is to find all objects in S satisfying query $q = q(a, b, c)$.
In this example, attribute a is non-local for a system S, so the query answering system associated with S has to contact other agents requesting a definition of a in terms of $\{b,c,d\}$. Assume that the system $S1$ is contacted. The most informative definition of a, extracted from $S1$, would involve only attributes $\{b, c, d\} \cap \{a, b, c, e, f\} = \{b, c\}$. If not, the optimal a-reduct is the one which has minimal number of elements outside {b,c}.

For system $S1$:

$b1 = \{y2, y4, y5\}$
$b2 = \{y1, y3\}$
$c1 = \{y2\}$
$c2 = \{y1, y3, y4, y5\}$
$a1 = \{y1, y4, y5\}$
$a2 = \{y2, y3\}$

Set of objects corresponding to $c1$ is included in set of objects corresponding to a2. Therefore we can obtain a rule $c1 \rightarrow a2$. From remaining sets we build pairs, triples etc. of values of attributes up to the fixed point, which are:

$b1 * c2 = \{y4, y5\}$
$b2 * c2 = \{y1, y3\} = b2$

The set of objects $\{y4, y5\}$ are included in set described by $a1$, so in such case we have next rule $b1 * c2 \rightarrow a1$.

All these new definitions are stored in the knowledge base of a client and then used to chase the missing attributes. But, before any chase algorithm, called rule-based chase, can be applied, semantic inconsistencies among sites have to be somehow resolved. Assuming the same semantics at the client S and agent $S1$ sites, query q will undergo the transformation process, and the resulting query depends only on attributes b and c so it can be processed by local QAS for client site. If semantic differ at all client and agents sites, then the new query has no clear meaning. Task ontologies with rough approach to query transformation should be applied to handle this problem.

Definition 1:

We say that $S(A) = (X; A; V)$ is an incomplete information system of type λ, if $S(A)$ is an incomplete information system introduced by Pawlak in [7] and the following two conditions hold:

$$a_S(x) = \{(a_i, p_i): 1 \leq i \leq m\} \text{ for any } x \in X, a \in A$$

$$\sum_{i=1}^{m} p_i = 1, \text{ and } (\forall i)(p_i \geq \lambda)$$

Now, let us assume that $S1(A)$, $S2(A)$ are incomplete information systems, both of type λ. The same set X of objects is stored in both systems and the same set A of attributes is used to describe them. The meaning and granularity of values of attributes from A in both systems $S1$ and $S2$ is also the same. Additionally we assume that::

$$a_{S_1}(x) = \{(a_{1i}, p_{1i}): 1 \leq m_1\} \text{ and } a_{S_2}(x) = \{(a_{2i}, p_{2i}): 1 \leq m_2\}.$$

We say that containment relation ψ holds between $S1$ and $S2$, if the following two conditions hold:
$(\forall x \in X)(\forall a \in A)[card(a_{S_1(x)}) \geq card(a_{S_2(x)})]$

$$(\forall x \in X)(\forall a \in A)[card(a_{S_1(x)}) = card(a_{S_2(x)})] \rightarrow [\sum_{i \neq j} |p_{2i} - p_{2j}| \geq \sum_{i \neq j} |p_{1i} - p_{1j}|]$$

Instead of saying that containment relation holds between $S1$ and $S2$, we can equivalently say that $S1$ was transformed into $S2$ by containment mapping ψ. This fact can be presented as a statement

$$\psi(S1) = S2 \text{ or } (\forall x \in X)(\forall a \in A)[\psi(a_{S_1}(x)) = \psi(a_{S_2}(x))]$$

Similarly, we can either say that $a_{S_1}(x)$ was transformed into $a_{S_2}(x)$ by ψ or that containment relation ψ holds between $a_{S_1}(x)$ and $a_{S_2}(x)$.

So, if containment mapping ψ converts an information system $S1$ to $S2$, then $S2$ is more complete than $S1$. Saying another words, for a minimum one pair $(a, x) \in A \times X$, either ψ has to decrease the number of attribute values in $a_{S_1}(x)$ or the average difference between confidences assigned to attribute values in $a_{S_2}(x)$ has to be increased by ψ.
To give an example of a containment mapping ψ, let us take two information systems $S1$, $S2$ both of the type λ, represented as Table 6 and Table 7.

Table 6. Information System S_1

X	A	B	c	d	e
x1	$\{(a_1, \frac{1}{3}), (a_2, \frac{2}{3})\}$	$\{(b_1, \frac{2}{3}), (b_2, \frac{1}{3})\}$	c_1	d_1	$\{(e_1, \frac{1}{2}), (e_2, \frac{1}{2})\}$
x2	$\{(a_2, \frac{1}{4}), (a_3, \frac{3}{4})\}$	$\{(b_1, \frac{1}{3}), (b_2, \frac{2}{3})\}$		d_2	e_1
x3		b_2	$\{(c_1, \frac{1}{2}), (c_3, \frac{1}{2})\}$	d_2	e_2
x4	a_3		c_2	d_1	$\{(e_1, \frac{2}{3}), (e_2, \frac{1}{3})\}$
x5	$\{(a_1, \frac{2}{3}), (a_2, \frac{1}{3})\}$	b_1	c_2		e_1
x6	a_2	b_2	c_3	d_2	$\{(e_2, \frac{1}{3}), (e_3, \frac{2}{3})\}$
x7	a_2	$\{(b_1, \frac{1}{4}), (b_2, \frac{3}{4})\}$	$\{(c_1, \frac{1}{3}), (c_2, \frac{2}{3})\}$	d_2	e_2
x8		b_2	c_1	d_1	e_2

It can be easily checked that the values assigned to $e(x1)$, $b(x2)$, $c(x2)$, $a(x3)$, $e(x4)$, $a(x5)$, $c(x7)$, and $a(x8)$ in $S1$ are different than the corresponding values in $S2$. In each of these eight cases, an attribute value assigned to an object in $S2$ is less general than the value assigned to the same object in $S1$. It means that $\psi(S1) = S2$.
From now on, an agent will be denoted by AG(S;KB), where S is an incomplete information system of type λ and KB is a knowledge base containing rules extracted from information systems of other agents collaborating with AG(S;KB).

Table 7. Information System S2

X	a	b	c	d	E
x1	$\{(a_1, \frac{1}{3}), (a_2, \frac{2}{3})\}$	$\{(b_1, \frac{2}{3}), (b_2, \frac{1}{3})\}$	c_1	d_1	$\{(e_1, \frac{1}{3}), (e_2, \frac{2}{3})\}$

x2	$\{(a_2,\frac{1}{4}),(a_3,\frac{3}{4})\}$	b_1	$\{(c_1,\frac{1}{3}),(c_2,\frac{2}{3})\}$	d_2	e_1
x3	a_1	b_2	$\{(c_1,\frac{1}{2}),(c_3,\frac{1}{2})\}$	d_2	e_3
x4	a_3		c_2	d_1	e_2
x5	$\{(a_1,\frac{3}{4}),(a_2,\frac{1}{4})\}$	b_1	c_2		e_1
x6	a_2	b_2	c_3	d_2	$\{(e_2,\frac{1}{3}),(e_3,\frac{2}{3})\}$
x7	a_2	$\{(b_1,\frac{1}{4}),(b_2,\frac{3}{4})\}$	c_1	d_2	e_2
x8	$\{(a_1,\frac{2}{3}),(a_2,\frac{1}{3})\}$	b_2	c_1	d_1	e_3

4. Query Processing based on Collaboration and Chase

Assume that we have a group G of collaborating agents and user submits a query $q(B)$ to an agent AG($S(A)$;KB) from that group, where:

- $S(A) = (X; A; V)$, $KB = \emptyset$,
- B are the attributes used in $q(B)$,
- $A \cap B \neq \emptyset$.

All attributes in $B \setminus [A \cap B]$ are called foreign for AG($S(A)$;KB). Since AG($S(A)$;KB) can collaborate with other agents in G, definitions of foreign attributes for AG(S(A);KB) can be extracted from information systems associated with agents in G. In [8], it was shown that agent AG($S(A)$;KB) can answer the query q(B) assuming that definitions of all values of attributes from $B \setminus [A \cap B]$ can be extracted at the remote sites for S and used to answer $q(B)$.

Foreign attributes for S, can be seen as attributes with only null values assigned to all objects in S. Assume now that we have three collaborating agents:
AG(S;KB), AG(S_1;KB_1), AG(S_2;KB_2), where S=(X;A;V), S_1=(X_1;A_1;V_1), S_2=(X_2;A_2;V_2), and $KB = KB1 = KB2 = \emptyset$. If the consensus between AG(S;KB) and AG(S_1;KB_1) on the knowledge extracted from $S(A \cap A1)$ and $S_1(A \cap A_1)$ is closer than the consensus between AG(S;KB) and AG(S_2;KB_2) on the knowledge extracted from $S(A \cap A_2)$ and $S_2(A \cap A_2)$, then AG(S_1;KB_1) is chosen by AG(S;KB) as the agent to be asked for help in solving user queries. Rules defining foreign attribute values for S are extracted at S_1 and stored in KB.

Assuming that systems S_1, S_2 store the same sets of objects and use the same attributes to describe them, system S_1 is more complete than system S_2, if $\psi(S_2) = S_1$. The question remains, if the values predicted by the imputation process are really correct, and if not, how far they are (assuming that some distance measure can be set up) from the correct values which clearly are unknown? Classical approach, to this kind of problems, is to start with a complete information system and remove randomly from it, e.g. 10 percent of its values and next run the imputation algorithm on the resulting system. The next step is to compare the descriptions of objects in the system which is the outcome of the imputation algorithm with descriptions of the same objects in the original system. But, before we can continue any further this discussion, we have to decide on the interpretation of functors "or" and "and", denoted in this paper by "+" and "*", correspondingly. We will adopt the semantics of terms proposed in [10] since their semantics preserves distributive property, which means: $t_1 * (t_2 + t_3) = t_1 * t_2 + t_1 * t_3$, for any queries t_1, t_2, t_3.
So, let us assume that $S = (X, A, V)$ is an information system of type λ and t is a term constructed in a standard way from values of attributes in V seen as constants and from two

functors + and *. By $N_S(t)$, we mean the standard interpretation of a term t in S defined as (see [10]):

- $N_S(v) = \{(x, p) : (v, p) \in a(x)\}$, for any $v \in V_a$
- $N_S(t_1 + t_2) = N_S(t_1) \oplus N_S(t_2)$
- $N_S(t_1 * t_2) = N_S(t_1) \otimes N_S(t_2)$

where for any $N_S(t_1) = \{(x_i, p_i)\}_{i \in I}, N_S(t_2) = \{(x_j, q_j)\}_{j \in J}$ we have:

- $N_S(t_1) \oplus N_S(t_2) =$
- $\{(x_i, p_i)\}_{i \in I \setminus J}\} \cup \{(x_j, q_j)\}_{j \in J \setminus I} \cup \{(x_i, \max(p_i, q_i)\}_{j \in I \cap J}$
- $N_S(t_1) \otimes N_S(t_2) = \{(x_i, p_i \cdot q_i)\}_{i \in I \cap J}\}$

Assume that AG(S;KB) is an agent, where $S = (X, A, V)$ and KB contains definitions of attribute values in B. Clearly $A \cap B = \emptyset$. Assume also that the set of all incomplete attributes in A is denoted by In(A). Saying another words In(A)={a□A:(□x□X)[card(a(x))≠1]} and $L(D) = \{(t \to v_c) \in D : c \in In(A)\}$ is a consistent set of rules in S.

The null value imputation algorithm Chase, given below, converts information system $S(A \cup B)$ of type λ to a new more complete information system Chase($S(A \cup B)$) of the same type. Initially NULL values are assigned to all attributes in B for all objects in $S(A \cup B)$.

Algorithm Chase(S, $In(A)$, $L(D)$)

INPUT

- System $S = (X, A, V)$,
- Set of incomplete attributes $In(A) = \{a_1, a_2, ..., a_k\}$,
- Set of rules $L(D)$.

BEGIN

$j := 1$;

while $j \le k$ **do**
begin
$S_j := S$;
for all $x \in X$ **do**
$p_j := 0$;
begin
$b_j(x) := \emptyset$;
$n_j := 0$;
for all $v \in V_{a_j}$ **do**
begin
if $card(a_j(x)) \neq 1$ and $\{(t_i \to v) : i \in I\}$ is a maximal subset of rules from $L(D)$ such that $(x, p_i) \in N_{S_j}(t_i)$ **then**
if $\sum_{i \in I}[p_i \cdot conf(t_i \to v) \cdot \sup(t_i \to v)] \ge \lambda$ **then**

begin

$b_j(x) := b_j(x) \cup \{(v, \Sigma_{i \in I}[p_i \cdot conf(t_i \rightarrow v) \cdot \sup(t_i \rightarrow v)])\}$ $n_j :=$

$n_j + \Sigma_{i \in I}[p_i \cdot conf(t_i \rightarrow v) \cdot \sup(t_i \rightarrow v)]$;

end

end

$p_j := p_j + n_j$;

end

if $\Psi(a_j(x)) = [b_j(x) / p_j]$;

then $a_j(x) := [b_j(x) / p_j]$;

$j := j + 1$;

end

$S := \bigcap \{S_j : 1 \leq j \leq k\}$

Chase (*S*, *In*(*A*), *L*(*D*))

END

OUTPUT

System *Chase*2 (*S*).

The algorithm *Chase* is chasing information system *S*, attribute by attribute, changing values of attributes assigned to objects in *X* only after all incomplete attributes in *S* are being chased. This process is continued till the fixed point is reached (no changes in *S* are made by *Chase*) The proposed algorithm is new in comparison to known strategies for chasing NULL values in relational tables because of the assumption about partial incompleteness of data (sets of weighted attribute values can be assigned to an object as its value). Algorithm ERID [3]) is used by Chase algorithm to extract rules from this type of data. Algorithm Chase converts the incomplete information system $S(A \cup B)$ to a new information system of type λ which is more complete.

Now, let us assume that agent AG(*S;KB*) represents the client site, where *S* is a partially incomplete information system of type λ. When a query *q*(*B*) is submitted to AG($S = (X, A, V)$;*KB*), its query answering system QAS will replace *S* by Chase(*S*) and next will solve the query using, for instance, the strategy proposed in [10]. Clearly, we can argue why the resulting information system obtained by Chase cannot be stored aside and reused when a new query is submitted to AG(*S;KB*)? If AG(*S;KB*) does not have many updates, we can do that by keeping a copy of Chase(*S*) and next reuse that copy when a new query is submitted to AG(*S;KB*). System Chase(*S*), if stored aside, cannot be reused by QAS when the number of updates in the original *S* and/or *KB* exceeds a given threshold value.

5. In Search for the Best Agent

Assume again that agent AG(*S;KB*) represents the client site. As we already pointed out, the knowledge base *KB*, contains rules extracted from information systems representing other agents. Our goal is to find optimal i-agent AG(*Si;KB*) for client AG(*S;KB*), where by optimal we mean an agent of maximal precision and recall. The distance between two agents is calculated using the formula:

$$d(S_1,S_2) = \frac{\sum_r d_r(S_1 \to S_2) + \sum_r d_r(S_2 \to S_1)}{\sum_r sup\, rS_1 \cdot conf\ rS_1 + \sum_r sup\, rS_2 \cdot conf\ rS_2}$$

where

$$d_r(S_1 \to S_2) = \left| \frac{sup\, rS_2 \cdot conf\ rS_2}{\max\ (sup\, rS_1, 1)} - \frac{sup\, rS_1 \cdot conf\ rS_1}{\max\ (sup\, rS_2, 1)} \right|$$

From all the agents we have, we choose the agent with the minimal value of $d(S_1 \to S_2)$ which corresponds to the closest agent to the client. The definition of distance *d* discovered at the closest agent to the client and stored in KB of the client will guarantee that Query Answering System connected with the client has maximal precision and recall in group of agents having *d*.

Example 3

Let us assume we have three information systems *S*, S_1, S_2, as represented as Table 8, Table 9 and Table 10. Client *S* has no information about attribute *d*, which appears in other systems such as S_1 and S_2. Our goal is to choose one of the system either AG(*S1,KB*) or AG(*S*2,*KB*), from which we will be able to predict values of *d* in system *S*.

Table8. Information System S

Z	A	b	c	d
z1	1	2	L	
z2	1	1	H	
z3	2	1	H	
z4	0	2	H	
z5	2	2	L	
z6	0	3	L	

Table 9. Information System S1

X	a	b	c	d	E
x1	0	1	H	3	-
x2	0	2	L	3	+
x3	1	3	L	3	+
x4	1	1	H	3	-
x5	0	2	L	1	+
x6	1	3	L	1	+
x7	2	1	H	3	-

Table 10. Information System S2

Y	a	b	c	d
y1	1	1	H	1
y2	1	1	H	3
y3	0	2	L	1
y4	0	3	L	3
y5	2	2	L	1
y6	2	3	H	3

Because attributes *a, b, c* are common in all of the systems, first we extract rules describing them. For each rule we calculate support and confidence in a standard way. For system S_1 we have:

$$b_1 \to a_0, \mathrm{sup} = 1, conf = \frac{1}{3}$$
$$b_1 \to a_1, \mathrm{sup} = 1, conf = \frac{1}{3}$$
$$b_1 \to a_2, \mathrm{sup} = 1, conf = \frac{1}{3}$$
$$b_2 \to a_0, \mathrm{sup} = 2, conf = 1$$
$$b_2 \to a_1, \mathrm{sup} = 2, conf = 1$$
$$c_H \to a_0, \mathrm{sup} = 1, conf = \frac{1}{3}$$
$$c_H \to a_1, \mathrm{sup} = 1, conf = \frac{1}{3}$$
$$c_H \to a_2, \mathrm{sup} = 1, conf = \frac{1}{3}$$
$$a_0 \to b_1, \mathrm{sup} = 1, conf = \frac{1}{3}$$

...

$$a_0 * c_H \to b_1, \mathrm{sup} = 1, conf = 1$$
$$a_0 * c_L \to b_2, \mathrm{sup} = 2, conf = 1$$

...

For system S2 we have:

$$b_1 \to a_0, \mathrm{sup} = 2, conf = 1$$
$$b_1 \to a_1, \mathrm{sup} = 2, conf = 1$$
$$b_2 \to a_0, \mathrm{sup} = 1, conf = \frac{1}{2}$$
$$b_2 \to a_2, \mathrm{sup} = 1, conf = \frac{1}{2}$$
$$b_3 \to a_0, \mathrm{sup} = 1, conf = \frac{1}{2}$$
$$b_3 \to a_2, \mathrm{sup} = 1, conf = \frac{1}{2}$$

$$c_H \to a_1, \mathrm{sup} = 2, conf = \frac{2}{3}$$
$$c_H \to a_2, \mathrm{sup} = 1, conf = \frac{1}{3}$$
$$c_L \to a_0, \mathrm{sup} = 2, conf = \frac{2}{3}$$
$$c_L \to a_2, \mathrm{sup} = 1, conf = \frac{1}{3}$$

$a_0 \rightarrow b_2, sup = 1, conf = \frac{1}{2}$

...

$a_2 * b_3 \rightarrow c_H, sup = 1, conf = 1$
$a_2 * d_1 \rightarrow c_L, sup = 1, conf = 1$

...

We do the same for system S.

The distance between S and S_1 is calculated: $d(S \rightarrow S_1) = \frac{33.36+20.31}{20+34.66} = 0.83$

and the distance between S and S_2: $d(S \rightarrow S_2) = \frac{33.36+17.17}{20+29.66} = 0.85$

Because the distance between S and S_1 is smaller than between S and S_2, we choose S_1 as the better agent for contact with S.

In next step the chosen - the closest agent S_1 contacts with information system S, to improve this system, using the containment relation mentioned earlier.

6. Conclusion

In this paper we assumed we have a group of collaborating agents, and the user submits a query to an agents from this group. The knowledge base corresponding to given information systems, contains rules extracted from information systems representing other agents. We proposed method of finding and identifying optimal agent for client, where by optimal we mean an agent of maximal precision and recall. The distance between two agents was calculated and tested. To improve our strategy, we can look for additional hidden slots taking into consideration. We can choose attributes with the highest support.

References

[1] Benjamins, V. R., Fensel, D., Prez, A. G. (1998) Knowledge management through ontologies, in Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM-98), Basel, Switzerland.

[2] Chandrasekaran, B., Josephson, J. R., Benjamins, V. R. (1998) The ontology of tasks and methods, in Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management, Ban_, Alberta, Canada

[3] Dardzinska, A., Ras, Z.W. (2003) On Rules Discovery from Incomplete Information Systems, in Proceedings of ICDM'03 Workshop on Foundations and New Directions of Data Mining, (Eds: T.Y. Lin, X. Hu, S. Ohsuga, C. Liau), Melbourne, Florida, IEEE Computer Society, 2003, 31-35

[4] Fensel, D., (1998), Ontologies: a silver bullet for knowledge management and electronic commerce, Springer-Verlag, 1998

[5] Guarino, N., ed. (1998) Formal Ontology in Information Systems, IOS Press, Amsterdam

[6] Guarino, N., Giaretta, P. (1995) Ontologies and knowledge bases, towards a terminological clarification, in Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, IOS Press

[7] Pawlak, Z. (1991) Information systems - theoretical foundations, in Information Systems Journal, Vol. 6, 1981, 205-218

[8] Ras, Z.W., Dardzinska, A. (2004) Ontology Based Distributed Autonomous Knowledge Systems, in Information Systems International Journal, Elsevier, Vol.29, No. 1, 2004, 47-58

[9] Ras, Z.W., Dardzinska, A. (2006) Solving Failing Queries through Cooperation and Collaboration, World Wide Web Journal, Springer, Vol. 9, No. 2, 2006, 173-186

[10] Ras, Z.W., Joshi, S. Query approximate answering system for an incomplete DKBS, in Fundamenta Informaticae Journal, IOS Press, Vol. 30, No. 3/4, 1997, 313-324
[11]Sowa, J.F. (2000a) Ontology, metadata, and semiotics, in B. Ganter & G. W.Mineau, eds., Conceptual Structures: Logical, Linguistic, and Computational Issues, LNAI, No. 1867, Springer-Verlag, Berlin, 2000, pp. 55-81
[12] Sowa, J.F. (2000b) Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole Publishing Co., Paci_c Grove, CA.
[13] Sowa, J.F. (1999a) Ontological categories, in L. Albertazzi, ed., Shapes of Forms: From Gestalt Psychology and Phenomenology to Ontology and Mathematics, Kluwer Academic Publishers, Dordrecht, 1999, pp. 307-340.
[14] Van Heijst, G., Schreiber, A., Wielinga, B. (1997) Using explicit ontologies in KBS development, in International Journal of Human and Computer Studies, Vol. 46, No. 2/3, 183-292.

Authors

Agnieszka Dardzinska works at the Bialystok University of Technology in the Department of Mechanics and Applied Computer Science. Her main interest is based on a field of data mining and knowledge discovery in incomplete information systems. She works with null values using chase methods.