Multi-Expert Disease Diagnosis System over Symptom Data Grids on the Internet

Amin Milani Fard, Hossein Kamyar and Mahmooud Naghibzadeh

Department of Computer Engineering, Ferdowsi University of Mashhad, Iran

Abstract: Nowadays practical databases become so huge containing billions of records which are needed to be statistically analyzed for useful knowledge extraction. By the emergence of the new Grid technology, simultaneous effective use of distributed computational and informational resources has been provided. Users in a data grid can access information easily without knowing the resource position. Knowledge discovery from databases is the process of nontrivial extraction of implicit, previously unknown and potentially useful information from data. In this work a new grid knowledge discovery approach was designed and based on the idea a sample multi-expert disease diagnosis system has been developed. Game theory and soft computing mechanisms were also applied to our method for the improvement of knowledge mining and representation.

Key words: Association rule mining • data grids • data mining • Dempster-Shafer theory • game theory • multi-agent systems • soft computing

INTRODUCTION

Considering the fast growth of data contents size and variety, finding useful information from collections of data have been extensively investigated in past decades. Practical databases become so huge containing billions of records which are needed to be statistically analyzed for useful knowledge extraction. Therefore knowledge discovery from databases (KDD) techniques were introduced as the process of nontrivial extraction of implicit, previously unknown and potentially useful information from data [1]. Nowadays Grid technology [2] provides us simultaneous effective use of distributed computational and informational resources. Three main types of this technological phenomenon are known as resource discovery grids, computational grids and data grids. Research works are mostly focused on algorithms regarding discovery grids and computational grids, however in this works we focused on data grids in which users can access information easily without knowing the resource position.

In this project a multi-expert system for disease diagnosis knowledge discovery in an information Grid environment has been designed and developed based on our proposed general architecture. The structure of the paper is as follows. Section 2 describes related works on expert systems and distributed information retrieval in brief. The proposed multi-agent systems architecture and agents behaviours are declared in section 3; and a sample run result is dedicated to section 4. We finalize our work with a conclusion and future work part in section 5.

Related works: Expert systems (ES) are software tools to help experts and specialists for partial knowledge substitution and decision making. These systems work based on a knowledge base composed of Facts and Rules and an inference engine. MYCIN [3] was actually the first successful ES designed in 1970 at Stanford University with the purpose of assisting physician in diagnosis of infectious blood diseases and antibiotics. The system operated using a fairly simple inference engine and a knowledge base of almost 500 rules. It was never actually used in practice not because of any weakness in its performance but much because of ethical and legal issues related to the use of computers in medicine, in case it gives a wrong diagnosis.

There are two main methods of reasoning when using inference rules, forward chaining and backward chaining. Forward chaining [4] starts with the available data and uses inference rules to extract more data until reaching goal. Because the data determines which rules are selected and used, this method is called data-driven. Backward chaining [4], on the other hand, starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents. Because the list of goals determines which rules are

Corresponding Author: Mr. Amin Milani Fard, Department of Computer Engineering, Ferdowsi University of Mashhad, Iran

244
selected and used, this method is called goal-driven. Both of the methods explain are often employed by expert systems. With the growth of distributed processing approaches, methods for combining multiple expert systems knowledge, known as multi-expert systems, have been widely studied in the last decade [5].

Distributed information retrieval (DIR) aims at finding information in scattered sources located on different servers on a network. This problem, also known as federated search, involves building resource descriptions for each database, choosing which databases to search for particular information and merging retrieved results into a single result list [6, 7]. Some applications of information retrieval from multiple sources include meta-search engines, distributed genomic search, newsletter gathering and etc. DIR includes four sub-problems of resource description, resource selection, query translation and result merging [7]. The resource description problem is how to learn and describe the topics covered by each different database. The resource selection means, given a query and a set of resource descriptions, how to select a set of resources to search. The query translation means mapping a given information need in some base representation, automatically to query languages appropriate for the selected databases. Finally, after result lists have been returned from the selected databases, result merging is used to integrate them into a single ranked list.

Let us consider $m$ information servers, denoted by $S_i$ and $i=1...m$, each of which provided with its own information retrieval system (IRS). Here we assume that the retrieval engines produce weighted lists of retrieved documents as an indicator of the estimated relevance of each document with respect to the query. In order to provide user with a single ordered list of relevant documents, the individual lists produced by IRSs must be fused [8]. This fusion is not an easy problem as IRSs produced their result with different criteria and statistics such as average document length, text frequency inverse document frequency and etc [9].

The proposed system: Implementation platform: Our proposed multi-agent system architecture is based on Java Agent Development (JADE) framework [10]. JADE is a software development framework aimed at developing multi-agent systems and applications in which agents communicate using FIPA Agent Communication Language (ACL) messages and live in containers which may be distributed to several different machines. JADE uses RMI method for communication. One of the most important characteristics of this tool is that programmer is not required to handle variables and functions concurrency as it is done automatically by the system.

JADE is capable of linking Web services and agents together to enable semantic web applications. A Web service can be published as a JADE agent service and an agent service can be symmetrically published as a Web service endpoint. Invoking a Web service is just like invoking a normal agent service. Web services’ clients can also search for and invoke agent services hosted within JADE containers.

The Web Services Integration Gateway (WSIG) [11] uses a Gateway agent to control the gateway from within a JADE container. Interaction among agents on different platforms is achieved through the Agent Communication Channel. Whenever a JADE agent sends a message and the receiver lives on a different agent platform, a Message Transport Protocol (MTP) is used to implement lower level message delivery procedures [12]. Currently there are two main MTPs to support this inter-platform agent communication - CORBA IIOP-based and HTTP-based MTP.

Considering large-scale applications over separated networks, agent communications has to be handled behind firewalls and Network Address Translators (NATs), however, the current JADE MTP do not allow agent communication through firewalls and NATs. Fortunately, the firewall/NAT issue can be solved by using the current JXTA implementation for agent communication [13].

JXTA is a set of open protocols for P2P networking. These protocols enable developers to build and deploy P2P applications through a unified medium [14]. Obviously, JXTA is a suitable architecture for implementing MTP-s for JADE and consequently.
JADE agent communication within different networks can be facilitated by incorporating JXTA technology into JADE [13].

A multi-agent system for intelligent information retrieval in heterogeneous networks have been proposed in a previous work [15] and upon that architecture a microorganism DNA pattern search through web-based genomic engine [16] and a web-based criminal face recognition system [17] was proposed. In this work, we also use a same infrastructure to solve agent communication problem in the Grid using XML web services. In our proposed architecture, we use five different types of agents, each having its own characteristics as the followings:

1. Manager Agent (MA): The MA has the responsibility of managing the whole system excluding other agent's creation. The creation node determination is influenced by different criterion such as CPU power, available processor load, total memory amount, used memory amount, traffic around node and etc.

2. Broker Agent (BA): These agents will deliver the query from user to Inference Agents. The query is in the form of facts to be included in local IA knowledge base.

3. Association Rules Miner Agent (ARMA): ARMAs are used to discover useful association rules and convert them to First Order Logic (FOL) to be included in the local IA knowledge base. The local IA is the one which is responsible for its local LAN ARM agents.

4. Inference Agent (IA): Inference Agents use FOL based rules gathered by ARMAs, FOL based facts from BA and apply the inference mechanism (here forward chaining) and return their inference result to the response agent.

5. Response Agent (RA): This agent has the responsibility to show the result of retrieved information. To do so, RA collects IAs results and combine them using Dempster-Shafer method and and then writes them on the screen ordered by relation percentage.

The proposed multi-agent architecture is shown in Fig. 2.

As the most innovative design parts were done on ARMA, IA and RA, we focus on their details in the following sections.

Arm agent behaviour: Having created each ARMA, they would start mining according to their predefined behaviour. The well-known Apriori Association rule mining algorithm [18] is used for this matter. Two main parameters in this algorithm are MinSup and MinCon, which denote minimum acceptable support and confidence respectively. Although mostly these parameters are set to 50%, in our thesis we used a game theory-based mechanism to define almost
1) \[ L_1 = \{ \text{large 1-itemsets} \}; \]
2) for \( k = 2; L_k, \ast; k++ \) do begin
3) \[ C_k = \text{apriori-gen}(L_{k-1}); \] // new candidates
4) forall transactions \( t \in D \) do begin
5) \[ C_k = \text{subset}(C_k,t); \] // candidates contained in \( t \)
6) forall candidates \( c \in C_k \), do
7) \( c\text{count}++; \)
8) end
9) \[ L_k = \{ c \in C_k | c\text{count} \geq \text{minsup} \}; \]
10) end
11) Answer = \( \cup_i L_i \); \]

Fig. 3: Apriori Algorithm

optimal MinSup and MinConf values for them. We continue by Apriori algorithm definition and game theory basics.

**Apriori algorithm:** The Apriori algorithm [18] computes the frequent itemsets in the database through several iterations. Iterations \( i \) computes all frequent \( i \)-itemsets (itemsets with \( i \) elements). Each iteration has two steps: candidate generation and candidate counting and selection. In the first phase of the first iteration, the generated set of candidate itemsets contains all \( i \)-itemsets (i.e., all items in the database). In the counting phase, the algorithm counts their support searching again through the whole database. Finally, only \( i \)-itemsets (items) with a above required threshold will be selected as frequent. Thus, after the first iteration, all frequent \( i \)-itemsets will be known.

Basically, all pairs of items are candidates. Based on knowledge about infrequent itemsets obtained from previous iterations, the Apriori algorithm reduces the set of candidate itemsets by pruning *-a priori*-those candidate itemsets that cannot be frequent. The pruning is based on the observation that if an itemset is frequent all its subsets could be frequent as well. Therefore, before entering the candidate-counting step, the algorithm discards every candidate itemset that has an infrequent subset [19].

**Game theory approach:** Game theory [20] provides us with the mathematical tools to understand the possible strategies that utility-maximizing agents might use when making a choice. It is mostly concerned with modeling the decision process of rational humans, a fact that should be kept in mind as we consider its applicability to multi-agent systems. The simplest type of game considered in game theory is the single-shot simultaneous-move game. In this game all agents must take one action simultaneously. Each agent receives a utility that is a function of the combined set of actions. This is a good model for the types of situations often faced by agents in a multi-agent system where the encounters mostly require coordination [21].

In the one-shot simultaneous-move game we say that each agent \( i \) chooses a strategy \( s_i \in S_i \), where \( S_i \) is the set of all strategies for agent \( i \). These strategies represent the actions the agent can take. When we say that \( i \) chooses strategy \( s_i \) we mean that it chooses to take action \( s_i \). The set of all strategies chosen by all the agents is the strategy profile for that game and it is denoted by \( s \in S = \times_i S_i \).

Once all the agents make their choices and form the strategy profile \( s \) then each agent \( i \) receives a utility which is given by the function \( u_i(s) \). Notice that a player’s utility depends on the choices made by all the agents. A Game [22] is a 3-tuple \( G = \{N, \alpha, \Pi \} \) Where:

- \( N \) is the number of players
- \( \alpha = \{a_i\}, a_i = \{a_i^1, \ldots, a_i^n \} \) is the set of actions available to player \( i \)
- \( \Pi: \times_i a_i \rightarrow R^k \) is the Payoff function, i.e., assigns each player a real number payoff for any combination of all players actions.

It is possible that a player will choose randomly between its action choices, using different prior probabilities for each choice. These types of strategies are called mixed strategies and they are a probability distribution over an agent’s actions. We say that a mixed strategy for agent \( i \) is \( \sigma_i \in \Sigma_i = P(S_i) \) where \( P(S_i) \) is the set of all probability distributions over the set of pure strategies \( S_i \) [21]. The Nash equilibrium in an n-player game is a set of strategies, \( \Sigma = \{a' , \ldots, a_n' \} \); such that, given that for all 1 player \( i \) plays \( a' \), no player \( j \) can get a higher payoff by playing a strategy other then \( a' \) [23]. It has been shown that every game has at least one Nash equilibrium, as long as mixed strategies are allowed. If the system is in equilibrium then no agent will be tempted to take a different action.

A good classic example for equilibrium point in a game is two prisoners dilemma in which there are two prisoners kept in separated cells. If both confess they would be sentenced three years in jail. If one confess and the other one don’t (testify the one who confess is guilty) the first one goes 4 years in jail and another one would release. And if none of them confess they would be sentenced for a less important crime and will go one year in the jail each. The matrix in Fig. 4 shows that the best action a player can do in not to confess and the Nash equilibrium is (don’t confess, don’t confess).
Our proposed game model is to find best MinCon and MinSup for ARMAs so that the total system performance improves. To model the game let's assume in the real world a computer conference is going to be held and our players' aim is to publish at least a paper in the conference anyway possible. We also assume players are multi-expert so they can publish paper in different fields of computer science and engineering. Also we know that in each field only a limited number of papers would be accepted. As a result each player tries to submit any paper in any field he can and with any quality. So if another player does not present a paper in a particular filed that the other one did, there would be a 100% chance of acceptance in contrast with the one who did not proposed. However, if both players submit papers since we do not have any information about the quality, there would be an equal 50% chance for each player. Therefore it is trivial that the game equilibrium is the case that each player submits any paper and with any quality he can. (We assume different paper submission does not have negative effects such as time wasting and etc.)

The scenario is somehow the same for our ARMAs. ARM agents which are in charge of gathering first level knowledge, will not get any payoff if can not present any amount of knowledge. This is considered by choosing a suitable value for MinCon and MinSup in the run-time. So in contrast with suggested 50% confidence and 50% support, our game-theory based method will select the highest possible confidence and support even if they are under 50% for instance an association rule with 35% confidence.

This mechanism assures that all ARMAs do their best to propose something as a first level knowledge. One of the best examples that this approach would be successful is Diagnosis Expert Systems in which diagnostic test among different experts are different and maybe symptoms represent different disease with respect to conditions such as country, race, age and etc. The ARMA game matrix in Fig. 5 shows agents' payoff in winning percentage regarding their knowledge presentation.

Note that players also try to maximize their knowledge presentation lift (progress ratio) which is a measure of strength for an association.

**Inference agent behaviour:** The extracted association rules from ARMAs will be sent to IAs with their corresponding support and confidence. These rules plus the fact query sent by the BA will construct the IA knowledge base. By this time agents are able to inference results. Expert systems mostly use backward chaining; however, we decided to use forward chaining due to our system model which is goal-driven. The forward chaining starts by adding new fact P to the knowledge base and finds all conditional combinations having P in assumption.

**Response agent behaviour:** The response agent finally obtains all results from IAs and then combines them using Dempster-Shafer theory. These results will be then proposed to the user in an ordered list according to their relevancy. Now if the user is interested in selecting some particular results, the user profiling through system feedback is done using a hybrid neuro-fuzzy mechanism.

**Dempster-Shafer theory of evidence:** The Dempster-Shafer (DS) theory was first introduced by Dempster [24] and then extended by Shafer [25], but the kind of reasoning the theory uses can be found as far back as the seventeenth century. This theory is actually an extension to classic probabilistic uncertainty modeling. Whereas the Bayesian theory requires probabilities for each question of interest, belief functions allow us to base degrees of belief for one question on probabilities for a related question. These degrees of belief may or may not have the mathematical properties of probabilities; how much they differ from probabilities will depend on how closely the two questions are related. This theory has been used in information retrieval in [26-29].

This theory is a generalization to the Bayesian Theory of Probability. The most significant differences between DS theory and probability theory are the explicit representation of uncertainty and evidence combination mechanism in which made it more effective in document processing fields [26]. In information retrieval the uncertainty occurs in three cases.
• Existence of different evidences regarding relation of a document to a query
• Unknown number of evidences regarding relation of a document to a query
• Existence of incorrect evidences regarding relation of a document to a query

In the DS theory of evidence, Belief is a value to express certainty of a proposition. This belief is calculated with respect to a density function \( m: \mathcal{P}(U) \to [0,1] \) called Basic Probability Assignment (BPA), where \( m(A) \) represents Partial Belief amount of \( A \).

\[
m(\phi) = 0 \quad \sum_{A \in \mathcal{U}} m(A) = 1
\]

To measure the Total Belief amount of \( A \subseteq \mathcal{U} \) the belief function is defined as:

\[
Bel(A) = \sum_{B \subseteq A} m(B)
\]

Shafer defined doubt amount in \( A \) as the belief in \( A' \) and the Plausibility function as the total belief amount in \( A \)

\[
Dou(A) = Bel(A') \quad PI(A) = 1 - Dou(A) = \sum_{B \subseteq \mathcal{U}} m(B) - \sum_{B \subseteq A} m(B) = \sum_{B \subseteq A' \phi} m(B)
\]

\( Pl(A) \) is actually the high boundary of belief in \( A \) so that the correct belief in \( A \) is in the interval of \([Bel(A), Pl(A)]\) as shown in Fig. 6.

Dempster's rule of combination is a generalization of Bayes' rule. This rule strongly emphasises the agreement between multiple sources and ignores all the conflicting evidence through a normalization factor. Let \( m_1 \) and \( m_2 \) are the BPA's in a frame of discernment. The combination BPA is calculated in the following manner:

\[
m(A) = m_1 \otimes m_2 = \frac{\sum_{B \cap C = A \cap \phi} m_1(B)m_2(C)}{\sum_{B \cap C = \phi} m_1(B)m_2(C)}
\]

**Fig. 6: Graphical view of the belief concept**

**Soft computing method in user behaviour modeling:**
Soft computing-based methods which are intelligent hybrid combination of techniques such as fuzzy logic and control, artificial neural networks, evolutionary algorithms and etc, are widely used in data mining and web content search in current years sometimes known as Soft Web Mining [30]. This is due to data properties in web environment such as being unlabeled, heterogeneous, semi-structured, time variant and etc.

Since users of information retrieval systems have nonlinear behaviour, it is not possible to model their profile using classic methods. Different user profiling approaches have been investigated in [31, 32] and one of the suggested models are based on neuro-fuzzy systems for information filtering and personalization. In our project we take advantage of ANFIS (Adaptive-Network-based Fuzzy Inference Systems) [33] to satisfy the demand. This network is more flexible in contrast with the simple ones using a hybrid of fuzzy logic approach. It has been shown that ANFIS is equivalent to a fuzzy system.

Our suggested network has two inputs for support and confidence values and membership functions are partitioned to low, medium and high. In user behaviour modeling, we consider a neural network for each rule with the same THEN clause and train the network considering rule selection or not. Therefore, the user can personalize the knowledge retrieval system regarding desired support and confidence for particular rules.

**System run sample:** In order to show the knowledge mining process in our system, we take advantage of disease symptoms database. Since using diseases with completely separated symptoms are not desirable as a test case, we chose those with common signs which are: Cold, Flu, Bronchitis, Allergy, Asthma, Sinusitis, Strep throat and Gastroenteritis. At the first step, the extracted association rules done by ARMAs are as bellow:
(runny nose) $\rightarrow$ (cold) sup: 0.58 conf: 1.0 (fever headache) $\rightarrow$ (strep throat) sup: 0.55 conf: 1.0
(sore throat) $\rightarrow$ (cold) sup: 0.55 conf: 1.0 (sore throat headache) $\rightarrow$ (strep throat) sup: 0.44 conf: 1.0
(headache) $\rightarrow$ (cold) sup: 0.62 conf: 1.0 (sore throat fever) $\rightarrow$ (strep throat) sup: 0.34 conf: 1.0
(fever) $\rightarrow$ (cold) sup: 0.51 conf: 1.0 (sore throat fever headache) $\rightarrow$ (strep throat) sup: 0.24 conf: 1.0
(cough) $\rightarrow$ (cold) sup: 0.72 conf: 1.0 (breathlessness) $\rightarrow$ (asthma) sup: 0.58 conf: 1.0
(sore throat fever) $\rightarrow$ (cold) sup: 0.34 conf: 1.0 (wheeze) $\rightarrow$ (asthma) sup: 0.79 conf: 1.0
(headache) $\rightarrow$ (cold) sup: 0.20 conf: 1.0 (cough) $\rightarrow$ (asthma) sup: 0.62 conf: 1.0
(runny nose headache) $\rightarrow$ (cold) sup: 0.24 conf: 1.0 (wheeze cough) $\rightarrow$ (asthma) sup: 0.44 conf: 1.0
(sore throat runny nose) $\rightarrow$ (cold) sup: 0.37 conf: 1.0 (wheeze breathlessness) $\rightarrow$ (asthma) sup: 0.44 conf: 1.0
(sore throat headache) $\rightarrow$ (cold) sup: 0.24 conf: 0.9 (cough breathlessness) $\rightarrow$ (asthma) sup: 0.24 conf: 1.0
(cough fever) $\rightarrow$ (cold) sup: 0.41 conf: 1.0
(cough runny nose) $\rightarrow$ (cold) sup: 0.37 conf: 1.0
(sore throat cough) $\rightarrow$ (cold) sup: 0.34 conf: 1.0
(fever runny nose) $\rightarrow$ (cold) sup: 0.27 conf: 1.0
(cough headache) $\rightarrow$ (cold) sup: 0.41 conf: 1.0
(cough fever runny nose) $\rightarrow$ (cold) sup: 0.24 conf: 1.0
(sore throat cough fever) $\rightarrow$ (cold) sup: 0.27 conf: 1.0
(sore throat fever runny nose) $\rightarrow$ (cold) sup: 0.24 conf: 1.0
(sore throat cough runny nose) $\rightarrow$ (cold) sup: 0.24 conf: 1.0
(chills) $\rightarrow$ (flu) sup: 0.62 conf: 1.0
(runny nose) $\rightarrow$ (flu) sup: 0.58 conf: 1.0
(fever) $\rightarrow$ (flu) sup: 0.51 conf: 1.0
(muscle ache) $\rightarrow$ (flu) sup: 0.55 conf: 1.0
(cough) $\rightarrow$ (flu) sup: 0.72 conf: 1.0
(chills muscle ache) $\rightarrow$ (flu) sup: 0.24 conf: 1.0
(muscle ache cough) $\rightarrow$ (flu) sup: 0.54 conf: 1.0
(cough runny nose) $\rightarrow$ (flu) sup: 0.57 conf: 1.0
(chills cough) $\rightarrow$ (flu) sup: 0.41 conf: 1.0
(fever runny nose) $\rightarrow$ (flu) sup: 0.27 conf: 1.0
(cough fever) $\rightarrow$ (flu) sup: 0.41 conf: 1.0
(chills runny nose) $\rightarrow$ (flu) sup: 0.24 conf: 1.0
(muscle ache runny nose) $\rightarrow$ (flu) sup: 0.37 conf: 1.0
(muscle ache fever) $\rightarrow$ (flu) sup: 0.34 conf: 1.0
(cough fever runny nose) $\rightarrow$ (flu) sup: 0.24 conf: 1.0
(muscle ache cough runny nose) $\rightarrow$ (flu) sup: 0.24 conf: 1.0
(muscle ache fever runny nose) $\rightarrow$ (flu) sup: 0.24 conf: 1.0
(fever runny nose) $\rightarrow$ (muscle ache flu) sup: 0.24 conf: 0.875
(cough muscle ache fever) $\rightarrow$ (flu) sup: 0.27 conf: 1.0
(muscle ache cough fever runny nose) $\rightarrow$ (flu) sup: 0.24 conf: 1.0
(headache) $\rightarrow$ (stroke) sup: 0.51 conf: 1.0
(fever) $\rightarrow$ (stroke) sup: 0.51 conf: 1.0
(bronchitis) $\rightarrow$ (stroke) sup: 0.51 conf: 1.0
(itchy) $\rightarrow$ (allergy) sup: 0.62 conf: 1.0
(sneeze) $\rightarrow$ (allergy) sup: 0.79 conf: 1.0
(runny nose) $\rightarrow$ (allergy) sup: 0.58 conf: 1.0
(itchy runny nose) $\rightarrow$ (allergy) sup: 0.24 conf: 1.0
(sneeze itchy) $\rightarrow$ (allergy) sup: 0.44 conf: 1.0
(itchy sneeze) $\rightarrow$ (allergy) sup: 0.44 conf: 0.72
(sneeze runny nose) $\rightarrow$ (allergy) sup: 0.44 conf: 1.0
(runny nose sneeze) $\rightarrow$ (allergy) sup: 0.44 conf: 0.76
(itchy) $\rightarrow$ (allergy) sup: 0.62 conf: 1.0
(bronchitis) $\rightarrow$ (bronchitis) sup: 0.62 conf: 1.0
(breathlessness) $\rightarrow$ (bronchitis) sup: 0.62 conf: 1.0
(fever) $\rightarrow$ (bronchitis) sup: 0.51 conf: 1.0
(fever wheezing) $\rightarrow$ (bronchitis) sup: 0.41 conf: 1.0
(breathlessness wheezing) $\rightarrow$ (bronchitis) sup: 0.41 conf: 1.0
(eyepain) $\rightarrow$ (sinusitis) sup: 0.51 conf: 1.0
(eyepain) $\rightarrow$ (sinusitis) sup: 0.51 conf: 1.0
(headache) $\rightarrow$ (sinusitis) sup: 0.79 conf: 1.0
(cough) $\rightarrow$ (sinusitis) sup: 0.62 conf: 1.0
(cough eye pain) $\rightarrow$ (sinusitis) sup: 0.44 conf: 1.0
(eyepain headache) $\rightarrow$ (sinusitis) sup: 0.44 conf: 1.0
(eyepain) $\rightarrow$ (sinusitis) sup: 0.51 conf: 1.0
(headache) $\rightarrow$ (strep throat) sup: 0.79 conf: 1.0
(sore throat) $\rightarrow$ (strep throat) sup: 0.58 conf: 1.0
(fever) $\rightarrow$ (strep throat) sup: 0.72 conf: 1.0

The produced knowledge base by the LA is then...
IF has % vomit/diarrhea THEN has % gastroenteritis(s0.72,c1.0)
IF has % fever THEN has % gastroenteritis(s0.51,c1.0)
IF has % vomit/diarrhea has % fever
THEN has % gastroenteritis(s0.41,c1.0)
IF has % muscleache has % headache
THEN has % gastroenteritis(s0.24,c1.0)
IF has % muscleache has % vomit/diarrhea
THEN has % gastroenteritis(s0.34,c1.0)
IF has % fever has % headache
THEN has % gastroenteritis(s0.20,c1.0)
IF has % muscleache has % fever
THEN has % gastroenteritis(s0.34,c1.0)
IF has % vomit/diarrhea has % headache
THEN has % gastroenteritis(s0.41,c1.0)
IF has % muscleache has % vomit/diarrhea has % fever
THEN has % gastroenteritis(s0.27,c1.0)
IF has % headache THEN has % streptthroat(s0.79,c1.0)
IF has % sorethroat THEN has % streptthroat(s0.58,c1.0)
IF has % fever THEN has % streptthroat(s0.72,c1.0)
IF has % fever has % headache THEN has % streptthroat(s0.55,c1.0)
IF has % sorethroat has % headache THEN has % streptthroat(s0.44,c1.0)
IF has % sorethroat has % fever THEN has % streptthroat(s0.34,c1.0)
IF has % sorethroat has % fever has % headache THEN has % streptthroat(s0.24,c1.0)
IF has % runny nose THEN has % cold(s0.58,c1.0)
IF has % sorethroat THEN has % cold(s0.55,c1.0)
IF has % headache THEN has % cold(s0.62,c1.0)
IF has % fever THEN has % cold(s0.51,c1.0)
IF has % cough THEN has % cold(s0.72,c1.0)
IF has % sorethroat has % fever THEN has % cold(s0.34,c1.0)
IF has % fever has % headache THEN has % cold(s0.20,c1.0)
IF has % streptthroat has % headache THEN has % cold(s0.24,c1.0)
IF has % streptthroat has % fever THEN has % cold(s0.24,c1.0)
IF has % streptthroat has % fever has % headache THEN has % cold(s0.24,c1.0)
IF has % streptthroat has % runny nose THEN has % cold(s0.37,c1.0)
IF has % streptthroat has % cough THEN has % cold(s0.41,c1.0)
IF has % streptthroat has % cough has % fever THEN has % cold(s0.27,c1.0)
IF has % streptthroat has % cough has % runny nose THEN has % cold(s0.24,c1.0)
IF has % streptthroat has % fever then has % runny nose THEN has % cold(s0.27,c1.0)

Now the query containing facts of “headache” and “fever” is added to the KB:

has patient headache
has patient fever

In this sample the two IAs use forward chaining and send the two results to the RA depicted below. Finally in the last phase, RA implies D-S combination method and returns the final result to the user.

<table>
<thead>
<tr>
<th>Bronchitis: 0.0181</th>
<th>Flu: 0.0181</th>
<th>Sinusitis: 0.0181</th>
<th>Gastroenteritis 0.0363</th>
<th>Streptthroat: 0.0181</th>
<th>Cold: 0.0363</th>
</tr>
</thead>
</table>

m(Gastroenteritis) = 1.0171 * 0.00129591 = 0.001318070061
m(Flu) = 1.0171 * 0.00129591 = 0.001318070061
m(Sinusitis) = 1.0171 * 0.00129591 = 0.001318070061
m(Bronchitis) = 1.0171 * 0.00129591 = 0.001318070061
m(Fr) = 1.0171 * 0.00129591 = 0.001318070061
m(Sinusitis) = 1.0171 * 0.00129591 = 0.001318070061
Finally, most probable diagnosis would be *Gastroenteritis, Cold, Strep throat, Bronchitis, Flu* and *Sinusitis* respectively.

**CONCLUSIONS AND FUTURE WORK**

In this thesis, the KDD process and data mining methods was investigated and then a new multi-agent architecture for grid environment have been proposed. The most innovative techniques in our design of the system include game-theory modeling for competitive knowledge extraction, hierarchical knowledge mining and Dempster-Shafer result combination. A distributed desease diagnosis expert system was designed and implemented upon the proposed method in which results shows its performance in knowledge gathering and inferencing.

Regarding project novelty and open research areas in the field, there are surely good potential to complete hierarchical knowledge mining process including usage of approximation algorithms in knowledge extraction, modeling the process as a mixed strategic games and finding Nash equilibrium and developing an infrastructure for a high performance grid based search engine.

**ACKNOWLEDGEMENTS**

Authors would like to thank Dr. Mohsen Kahani and Dr. Mohammad R. Akbarzadeh, for many helpful comments that improved the work and Saeed Moghaddaszaeh who contributed in system implementation.

**REFERENCES**


