13

Data Mining for Decision Making in Multi-Agent Systems

Hani K. Mahdi, Hoda K. Mohamed and Sally S. Attia
Computer and Systems Engineering Department,
Faculty of Engineering, Ain Shams University, Cairo
Egypt

1. Introduction

The intelligent agent paradigm has generated such a remarkable interest in many application domains over the last two decades. It is growing to be a continuously evolving and expanding area. Agents, Software Agents or Intelligent Agents are intelligent in the sense that they are adaptive, independent, and possess reasoning capability. They can plan and execute tasks in cooperation with other agents in order to satisfy their goals. A Multi-Agent System (MAS) is defined as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver (Agent). The increasing interest in MAS research is due to significant advantages inherent in such systems, including their ability to solve problems that may be too large for a centralized single agent, provide enhanced speed and reliability, and tolerate uncertain data and knowledge. Some of the key research issues related to problem-solving activities of agents in a multi-agent system MAS are in the areas of coordination, negotiation, and communication.

With advances in Web technologies, collaborative applications are now server based and the user interface is typically a Web browser. Thus, a collaborative application can be a Web-based solution that runs on a local server that allows people communicate and work together, share information and documents, and talk in real-time over the Internet. Recently, much research has been conducted in distributed artificial intelligence and collaborative applications. Several interesting methodologies and systems have been developed in areas such as distributed multi-agent systems for decision support, web search and information retrieval, information systems modeling, and supply chain management.

This chapter considers applying different data mining techniques for the decision making process in a Multi-Agent System for Collaborative E-learning (MASCE). The dynamism in e-learning can be made more powerful with the help of intelligent agents. Intelligent agents – the so called e-assistants or helper programs - can reside inside a computer and make the learning in e-learning occur dynamically to suit the need of the user. They can track the user's likes and dislikes in different areas, the level of knowledge and the learning style and accordingly recommend the best matching helpers for collaboration.

A previous research outlined the development and the implementation processes of a Multi-Agent System for Collaborative E-learning (MASCE) which is designed to be used to assist

the teaching and learning processes. This system considers the blended learning environment as a supplement to the face-to-face lecture. The goal is to incorporate the intelligence of the multi-agent system in a way that enables it to actively and intelligently support the educational processes, where multiple agents can interact to exchange information so that students may collaborate on how best to gain knowledge.

In this chapter we are going to outline the application of different data mining techniques to discover important information previously unknown from the large database tables obtained from MASCE. The use of data mining facilitates the decision making process. As the world grows in complexity, overwhelming us with the data it generates, data mining becomes our only hope for explaining the patterns that underlie it. Intelligently analyzed data is a valuable resource. It can lead to new insights and, in commercial settings, to competitive advantages.

Data mining is defined as the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage. The data is invariably present in substantial quantities. Useful patterns allow us to make nontrivial predictions on new data. In other words, they help to explain something about the data.

First, we are going to apply Rough Sets techniques to the decision tables in order to obtain decision rules that can be used to classify new unseen cases. Second, Decision Tree algorithms such as ID3 will be applied to the same decision tables to obtain decision trees that can be used in classification of new objects. Third, a hybrid data mining algorithm combining rough sets and decision trees is applied and the results are compared with the previous two techniques to determine which is most suitable for decision making in this particular application of multi-agent systems.

2. Multi-agent systems and their applications

Agents and Multi-Agent Systems (MAS) have emerged as a powerful technology to cope with the increasing complexity of a variety of Information Technology scenarios. We are not going to explain the full details of the agents because these are covered in other chapters. We are only going to provide a basic overview.

2.1 Multi-agent overview

The most widely accepted definition for the "agent" term is that "an agent acts on behalf of someone else, after having been authorized". This definition can be applied to software agents, which are instantiated and act instead of a user or a software program that controls them. The difficulty in defining an agent arises from the fact that the various aspects of agency are weighted differently, with respect to the application domain at hand. Wooldridge & Jennings have succeeded in combining general agent features into the following generic abstract definition integrating all the characteristics into the notion of an agent: "An agent is an autonomous software entity that -functioning continuously - carries out a set of goal-oriented tasks on behalf of another entity, either human or software system. This software entity is able to perceive its environment through sensors and act upon it through effectors, and in doing so, employ some knowledge or representation of the user's preferences" (Wooldridge, 1999).

An agent may have, depending on the domain it is situated in, some or all of the properties listed below (Symeonidis & Mitkas, 2005):

- Autonomy (considered a must-have feature by many researchers in the field of agents)
- Interactivity: Reactivity or Pro-activeness
- Adaptability
- Sociability
- Cooperativity
- Competitiveness
- Mobility
- Learning

In this chapter, we are going to concentrate on the "***learning***" feature which means that an agent should be able to learn (get trained) through its reactions and its interaction with the environment. This is one of the most fundamental features that agents should have. We are going to use different data mining techniques to extract decision rules which can help agents make intelligent decisions. The more efficient the training process, the more intelligent the agents.

(Nwana, 1995) classified agents with respect to the following dimensions:

- Mobility that differentiates agents into static or mobile.
- The *logic paradigm* they employ, which classifies them as either *deliberative* or *reactive.*

The fundamental characteristics that describe the agent are: *autonomy*, *cooperativity* and *learning*. Based on these axes, agents can be classified as collaborative agents, collaborative learning agents, interface agents and truly smart agents as shown in Figure 1. The combination of two or more of the above approaches classifies agents as hybrid, leading to mobile deliberative collaborative agents, static reactive collaborative agents, static deliberative interface agents, mobile reactive interface agents, etc.

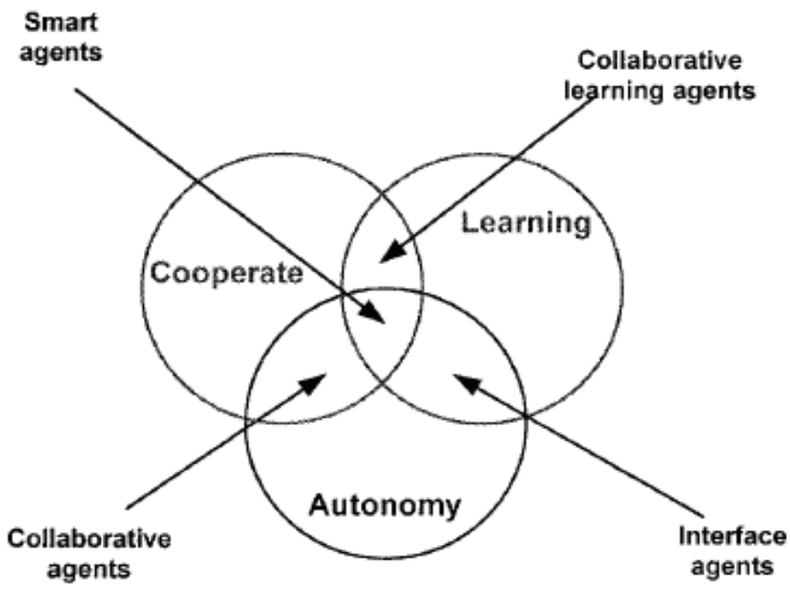


Fig. 1. The Nwana agent classification, according to their fundamental characteristics (Nwana, 1995)

A multi-agent system is one that consists of a number of agents, which interact with one-another. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do (Wooldridge, 2002). Several architectural styles have been used in the development of multi-agent systems (Buse and Wu, 2007; Sellers and Giorgini, 2005).

In order for MAS to solve common problems coherently, the agents must communicate amongst themselves, coordinate their activities, and negotiate once they find themselves in conflict. Agents communicate via messages, which can be of two types: assertions and queries.

MAS architectures exploit the coupling of the agent effectiveness, acting within a distributed environment, with the advantages of a strict and successfully coordinated framework, as far as communication and agent collaboration is concerned. MAS complexity is mainly dependent on the number of agents that participate in it. One could, therefore, argue that the simplest form of MAS consists of a single agent, while more elaborate MAS structures may comprise a substantial number of cooperating agents, or even smaller MAS (Symeonidis & Mitkas, 2005).

2.2 Multi-agent systems applications

The following issues differentiate between the rationale for MAS and single-agent systems (Agent Working Group, 2000):

- A single agent that handles a large amount of tasks lacks performance, reliability, and maintainability. On the other hand, MAS provide modularity, flexibility, modifiability, and extensibility, due to their distributed nature.
- A single agent cannot obtain (and provide to humans) extensive and specialized knowledge. MAS, composed of multiple distributed processes can access more knowledge resources, exploiting concurrent execution.
- Applications requiring distributed computing are better supported by MAS than by a single (often static) agent.
- Intelligence, as described by neuroscientists of the human mind, can be approached by a multi-processing system, rather than serial computing. Such a multi-processing computational system can only be implemented as a wide distributed environment where many agents act. Thus, MAS appear as the optimal solution for implementing.

It becomes evident that the choice between MAS or single-agent architecture must be guided by the application needs. If we decide, for example, to implement a system notifying us whenever we have an incoming e-mail, then a static single-agent implementation seems sufficient. On the other hand, if we are to implement a large-scale electronic marketplace, where many agents take part in electronic auctions to buy or sell goods, MAS architecture will be more suitable.

Multi-agent systems and e-learning

Dynamism in e-learning can be made more powerful with the help of intelligent agents. In e-learning, Intelligent Agents help make the learning in e-learning happen dynamically to suit the need of the user (Chen and Chiu, 2005; Shang et al. 2001). They can collect information about the user's likes and dislikes while using the system, the

level of knowledge and accordingly recommend the best matching helpers for collaboration.

E-learning has become one of the most popular teaching methods in recent years. At the same time, in the computational intelligence field, the Intelligent Agent paradigm gained a tremendous interest in many application domains over the last two decades. Current research in this area has approached integrating agents into educational systems. That is why we thought of building a Multi-Agent System for Collaborative E-learning (MASCE) as will be shown in details in the next section.

3. Proposed Multi-Agent System for Collaborative E-Learning (MASCE)

MASCE is to assist teaching and learning process and also to encourage collaborative learning among peers. This system shall be used in a blended learning environment as a supplement to the face-to-face lecture where students can use the system in the lab or from home after attending the traditional lecture in the faculty. The objective is to incorporate the intelligence of the multi-agents system in a way that enables it to actively and intelligently support the educational processes, where multiple agents can interact to exchange information so that students may collaborate on how best to gain knowledge (Mahdi and Attia, 2008).

The proposed MAS system considers two types of users; namely students and instructors. Each of these users has a corresponding agent. These are Student Agent and Instructor Agent. The Student Agent runs on the student's computer. Thus, each student is equipped with a Student Agent, which helps the learning process of the student. It manages the student's personal profile and also tracks the student actions during learning process and updates his profile accordingly. On the other hand, the Instructor Agent provides teaching materials, assesses the progress and participation of different students through quizzes, and manages the progress of the course.

The innovation in the proposed system is the introduction of the Assistant Agent which is initialized as soon as any of the users starts to use the system. The Assistant Agent runs on the system's server. It plays a centric role in the proposed system. It has a collaboration mechanism which will be used for "match-making" and "community-building" to help increase collaboration between peers in a certain course. It also gives hints to the instructor of the course to help in the teaching process such as statistics of the results of quizzes and summaries of students' profiles to help in the final grading. It acts a mediator (facilitator) between Student Agents and Instructor Agent of a specific course. After receiving the preferences (goals) of the instructor and the students, it will run autonomously and self-dependently. All the Assistant Agents of different courses are under the control of the Assistant Manager Agent. Thus the proposed MAS consist of three types of agents.

a. Student Agent

Each student has the corresponding Student Manager Agent that helps the learning process of the student. It acquires the student's preferences and profile. During the learning process, as the student enrolls in new courses, a dedicated student agent for each course is created. It tracks the student actions in that course. Accordingly, the tracking mechanism updates the student's profile and preferences. All the student agents of different courses of the same student are under the control of the Student Manager Agent as shown in Figure 2.

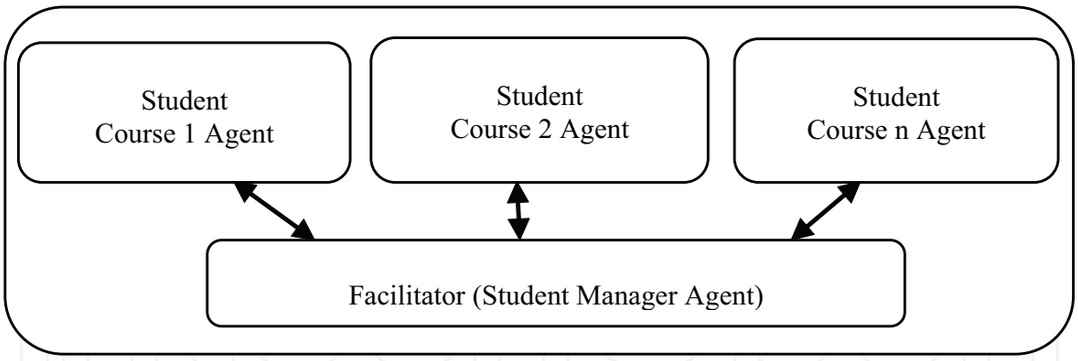


Fig. 2. Student Agent

b. Instructor Agent

The Instructor Manager Agent or simply the Assistant Agent assists the teaching process while interacting with the instructor. It is assigned for each instructor. For each course that is taught by the instructor a dedicated instructor agent is created. It provides teaching materials when requested by Assistant agent for distributing to students' agents, assesses the progress and participation of different students through quizzes, and manages the progress of the course. All the instructor agents of different courses of the same instructor are under the control of the Instructor Manager Agent as shown in Figure 3.

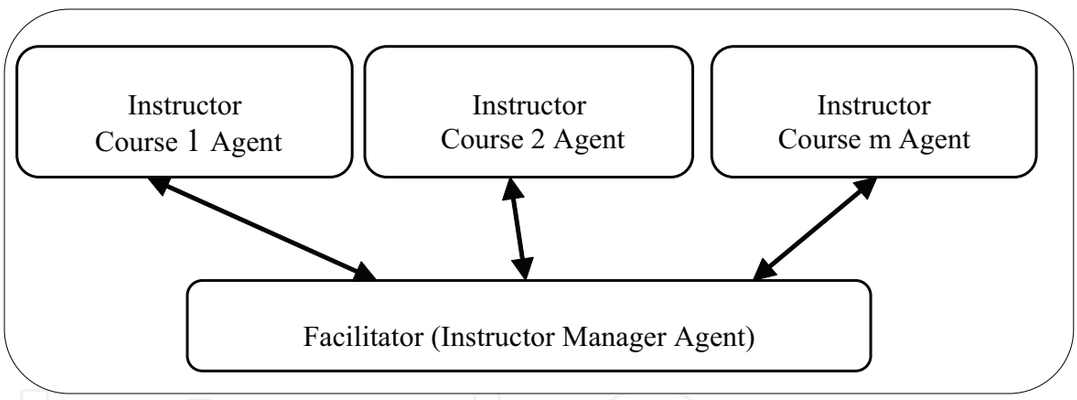


Fig. 3. Instructor Agent

c. Assistant Agent

The innovation in the proposed system is the introduction of the Assistant Agent shown in Figure 3which is initialized as soon as any of the users starts to use the system. The Assistant Manager Agent or simply the Assistant Agent plays a centric role in the proposed system. For each course, a dedicated Assistant Course Agent is created. It has a collaboration mechanism which will be used for "match-making" and "community-building" to help increase collaboration between peers in a certain course. It also gives hints to the instructor of the course to help in the teaching process such as statistics of the results of quizzes and summaries of students' profiles to help in the final grading. It acts a mediator (facilitator) between Student Agents and Instructor Agent of a specific course. After receiving the preferences (goals) of the instructor and the students, it will run autonomously and self-dependently. All the Assistant Agents of different courses are under the control of the Assistant Manager Agent as shown in Figure 4.

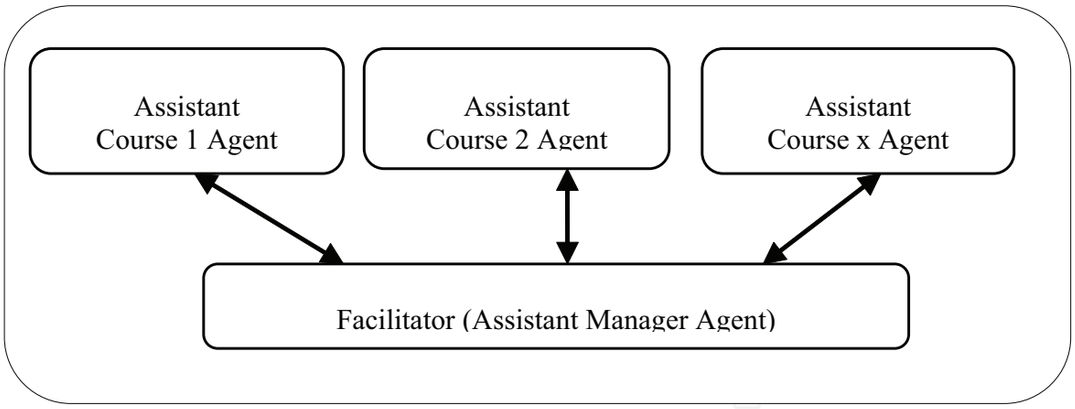


Fig. 4. Assistant Agent

The analysis and design phase of MASCE is done using Beliefs, Desires, Intentions-Agent Based Software Development (BDI-ASPD) (Jo et al., 2004). We find desires first from the system requirements and then find its intention and corresponding belief. This idea comes from the natural approach we usually do in the real world. An agent's beliefs are a set of data describing the state of the environment. They are the knowledge that intentions use to fulfill their goals (desires).

The course material is going to be structured in a hierarchical form where the course is divided into chapters and each chapter is divided into sections which in turn are divided into subsections and so on until we reach the leaves (concepts which cannot be divided any further). For each of these leaves the following will be provided:

1. Teaching materials
2. Quizzes to test student's knowledge level
3. Students' notes (blogs)
4. Discussion forums
5. Questions asked by students requiring help

The student can review all the teaching materials provided and add notes to his blogs if he wants to. He will take quizzes after each module to test his understanding (knowledge level) in that part so as to update his profile. If the student asks a question in a particular section, the Assistant Agent (Match Maker) will try to find the best potential helper for that question who is currently available online, willing and able to provide help. The Assistant Agent uses the students' models in that match making process.

Some of the parameters which are going to be used to in the model are static, they are collected from the student himself through a questionnaire given to the student when he first uses the system including:

1. Help willingness
2. Initial availability
3. Preferences such as cognitive style Maximum numbers of concurrent discussions,
4. Initial belief of the student knowledge level through a simple quiz given to the student to classify him as either: novice, beginner, intermediate or advanced.
5. Weighted importance of various attributes: such as if he requires help quickly from any available willing helper, or he would rather wait to be matched with the helper with the best knowledge level in the concept he is asking about.

Some of these parameters are dynamic; they are updated dynamically as the student interacts with the system and more new information is collected. Old information may be

outdated or even wrong. For example, after each help session between two students, an evaluation form is presented to each of them to evaluate his colleague. These peer evaluations along with the collected information about student by the tracking system such as rate of his responses, are all used to update the helpfulness parameter. The actions (parameters) that the tracking system will monitor and that will be collected, modeled and analyzed are:

1. Actual Availability
2. Frequency of logging to the system (number of times in one week period for example)
3. Banned topics or users
4. Preferred users
5. Quizzes taken to update student's knowledge level
6. Downloaded teaching materials
7. Blogs visited and notes added by the student to his blog
8. Number of postings on the forums
9. Frequency and type of questions asked to instructor or peers (not content-based)
10. Number of help requests accepted or rejected
11. Peer evaluation

Whenever a student has a question or a problem in a particular part of the course, he notifies his personal student agent. Interacting with the student personal agent in this manner also helps make users aware of the extent of the effort directed towards locating a suitable helper for them. Users tell their agents any important information that should be considered – e.g. that they know little about a particular topic, and that someone who would help on this topic should be someone who has been voted a good helper by others in the past. Sometimes it might be the case that the learner has a straightforward question, and that speed of response is more important, than depth of knowledge in a topic. This information is then conveyed by the personal student agent to the Assistant Agent (Match Maker) who tries to find the best potential helper who is currently ready or available (online), able (has a good knowledge level of that part) and willing to help. This helps fulfilling the first goal of MASCE.

The assistant agent also groups students together according to their similar preferences and complementing abilities into "buddies groups" to solve group projects or assignments. Thus a community of learners may be built up. So in addition to the one-to-one relationships that grow through the use of MASCE, learners can be arranged into groups with similar concerns who support each other on specific issues. This might be a problem solving session with one or more learners assisting each other with similar problems, or it might be a longer term study group where each member contributes to mutual understanding of the subject matter in some way. Interaction at the one-to-one and the group level is designed to bring learners closer together, and contribute to the evolution of a community of learners anxious to support each other in their learning. Thus the second goal of MASCE which is the promotion of collaboration and knowledge sharing can be fulfilled.

Using this principle of the ready, willing and able helper greatly increases the likelihood of benefit among participants: helpers are dedicated, and are not receiving help requests for which they have too little time or knowledge; the helpee is more likely to receive the kind of help they require, and will perceive the positive attitude of the helper. In return, the helpee may be more enthusiastic to assist the helper should an occasion arise where he is in a position to do so (students can add others to a 'friends list', indicating to their agent their particular willingness to help that person in the future). Thus the kind of community of

learners that is built in MASCE is designed to benefit everybody which is the third goal of MASCE (Mahdi and Attia, 2008 a).

4. Data mining techniques used in MASCE

Data mining refers to the analysis of the large quantities of data that are stored in computers. Data mining is about solving problems by analyzing data already present in databases. Data mining is defined as the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage (Chakrabarti et al., 2009).
Data mining tasks can be categorized into: summarization, classification, clustering, association, and trend analysis. Classification is a method of predicting the decisions for the new cases, based on their conditional features using a model learned from the already known attributes. It has been commonly studied by many data mining researches. Rule Based Classification (RBC) system is a part of classification which represents knowledge via a set of propositional rules. Rule-based data mining algorithms have a number of desirable properties. Rule sets are relatively easy for people to understand. RBC is widely used in the real world applications because of the easy interpretability of the extracted rules (Qin, 2009).
We will be focusing our research on classification task and more specifically on rule based classification. Modeling in RBC starts with the process of extracting a set of rules from data source that identifies key relationship between the attribute and class label. Then, the obtained rules are tested with unseen data. Rough Classifier (RC) and Decision Tree Classifier (DTC) are categorized as RBC. Both techniques apply different approach to perform classification but produce same structure of output with good classification results (Mohsin & Abd Wahab, 2008).

4.1 Three levels of knowledge diffusion for MAS

Following the above perspective, data mining techniques can be applied in three alternative manners, leading to three different types of knowledge, which, in turn, correspond to three distinct ways of knowledge diffusion to the MAS architecture (Symeonidis & Mitkas, 2005):

- Data mining on the application level of MAS

Data mining is performed on available application data, in order to discover useful rules and/or associations and/or patterns. The extracted knowledge is related to the scope of the end-user application and not its internal architecture. In this case, the technology coupling issue is viewed macroscopically, where the knowledge models extracted are intended to improve the efficiency of the MAS.

- Data mining on the behavior level of MAS

In this case, data mining is performed on log files containing agent behavior data (i.e., actions taken, messages exchanged, decisions made). The goal is to better predict future agent behaviors by eliminating unnecessary or redundant agent activity. The extracted knowledge may result in more efficient agent actions. In this case, the coupling issue is viewed microscopically, where the knowledge models extracted are intended to improve the performance of an agent engaged in a MAS (i.e., to improve the internal MAS action flow).

- Data mining on evolutionary agent communities

At this level, we perform evolutionary data mining techniques on agent communities, in order to study agent societal issues. This approach tries to achieve the satisfaction of the

goals of an agent community, which evolves and learns through interaction. In this case, coupling is considered at a higher level of abstraction, where the main focus is on the formulation of the problem that the agent community has to deal with and the way the extracted knowledge is diffused to the agent community.

4.2 Rough sets mining in MASCE

Rough set theory has proved to be useful in Data Mining (DM) and Knowledge Discovery (KD). It constitutes a sound basis for data mining applications. The theory offers mathematical tools to discover hidden patterns in data. It identifies partial or total dependencies in databases, eliminates redundant data and gives an approach to solve some challenges as null values, missing data, dynamic data and others.

In the past decade there has been done a substantial progress in developing rough set methods for DM and KD. In particular, new methods for extracting patterns from data, decomposition of decision tables as well as new methodology for data mining in multi-agent systems have been developed.

Rough set theory was introduced by Pawlak in 1982. Since then, it has often proved to be an excellent mathematical tool for the analysis of a vague description of objects. The adjective vague (referring to the quality of information) is concerned with inconsistency or ambiguity. Rough sets can be applied on information represented in the form of a table called information system. It is composed of a 4-tuple as follows: $S = < U, Q, V, f >$ where U is the closed universe, a finite set of N objects $\{x_1, x_2, \ldots x_N\}$, Q is a finite set of n attributes $\{q_1, q_2, \ldots q_n\}$, the attributes in Q are further classified into disjoint condition attributes C and decision attributes D, $Q = C \cup D$. Such information systems are called decision tables, $V = \cup_{q \in Q} V_q$ where V_q is a domain (value) of the attribute q, $f = U \times Q \rightarrow V$ is the information function such that $f(x, q) \in V_q$ for every $q \in Q$, $x \in U$ (Cios et al., 1998).

The rough set philosophy is based on the assumption that with every object of the universe U there is associated a certain amount of information (data, knowledge). This information can be expressed by means of a number of attributes. The attributes describe the object. Attributes with preference-ordered domains are called criteria because they involve an evaluation such as: bad, medium, good.

Objects which have the same description are said to be indiscernible (similar) with respect to the available information. The indiscernibility relation thus generated constitutes the mathematical basis of rough set theory. It induces a partition of the universe into blocks of indiscernible objects, called elementary sets, which can be used to build knowledge about a real or abstract world. The use of the indiscernibility relation results in information granulation.

Any subset X of the universe may be expressed in terms of these blocks either precisely (as a union of elementary sets) or approximately. In the latter case, the subset X may be characterized by two ordinary sets, called the lower and upper approximations. A rough set is defined by means of these two approximations, which coincide in the case of an ordinary set. The lower approximation of X is composed of all the elementary sets included in X (whose elements, therefore, certainly belong to X), while the upper approximation of X consists of all the elementary sets which have a non-empty intersection with X (whose elements, therefore, may belong to X).

The difference between the upper and lower approximation constitutes the boundary region of the rough set, whose elements cannot be characterized with certainty as belonging or not

to X (by using the available information). The information about objects from the boundary region is, therefore, inconsistent or ambiguous. The cardinality of the boundary region states, moreover, the extent to which it is possible to express X in exact terms, on the basis of the available information. For this reason, this cardinality may be used as a measure of vagueness of the information about X (Cios et al., 1998).

We say that two objects of an information system are indiscernible if for a subset of the attributes A, they have the same value for each attribute: IND(A) = {(x, y) ∈ U × U: for all a ∈ A, f (x, a) = f (y, a)}. The indiscernibility relation IND(A) splits the universe U into a family of equivalence classes $\{X_1, X_2, X_3, \ldots..X_r\}$ and is denoted by A*. Objects belonging to the same equivalence class X_i are indiscernible. The equivalence classes X_i's are called A-elementary sets. $Des_A(X)$ denotes the description of A-elementary set X∈A*: $Des_A(X)$ = {(a = b): f (x, a) = b, ∀x ∈ X, a ∈ A}.

A given subset of attributes A∈ Q determines the approximation space AS = (U, IND (A)) in S. For a given A ∈ Q and a concept X ∈ U, the A-lower approximation $\underline{A}X$ of set X is the union of all those elementary sets each of which is fully contained by X i.e. $\underline{A}X$ contains all objects that can be classified as certainly belonging to the concept X (based on knowledge from A). $\underline{A}X$ is also called the A-positive region $POS_A(X)$ of X in S.

$$\underline{A}X = \{x \in U: [x]_A \subseteq X\} = \{Y \in A^*: Y \subseteq X\} \quad (1)$$

The A-upper approximations of set X is $\overline{A}X$ is the union of those elementary sets each of which has non-empty intersection with X i.e. it contains all objects that cannot be classified as not belonging to the concept X (based on knowledge from A).

$$\overline{A}X = \{x \in U: [x]_A \cap X \neq \Phi\} = \cup\{Y \in A^*: Y \cap X \neq \Phi\} \quad (2)$$

For given information system some attributes may be redundant (superfluous) with respect to a specific classification A*. Using the dependency properties of attributes, we can find a reduced set of the attributes without loss of classification power.

For an information system S, with A ⊆ Q an attribute a∈A is called dispensable if IND(A) = IND(A−{a}), otherwise a parameter a is indispensable. Absence of dispensable attributes does not reduce the classificatory power while the indispensable attributes carry the essential information about objects and cannot be removed without changing the classificatory power. The set of all indispensable attributes is called a core of A = Core (A). It may be an empty set. (from master thesis)

A proper subset E ⊂ A that preserves the classification generated by A is called a reduct of A, RED(A) if E is orthogonal (cannot be further reduced) and if E preserves classification as A i.e. E is a minimal set of attributes that discerns all objects in S that are discernable by A. More than one reduct of A can be identified forming a family of reducts $RED^F(A)$, their intersection forms core of A i.e. Core(A) = ∩ $RED^F(A)$ (Cios et al., 1998).

One of the important applications of rough sets is the generation of decision rules for a given information system for classification of known objects, or prediction of classes for new objects. One can find the rules classifying objects through different methods. The mostly adopted algorithm for extracting rules extracts rules using the intersections between classifications and partitions: Let decision table DT= <U, C ∪ D, V, f> where C represents the set of condition attributes, D represents the set of decision attributes, $C^* = \{X_1, X_2, \ldots.., X_r\}$ represents the condition classification of U and $D^* = \{Y_1, Y_2, .., Y_m\}$ represents the decision

classification of U. The decision rules are logically described as follows: If (conditions) then (decisions). The decision rule which is called τij can be extracted as follows:

$$\tau_{ij} = Des_C(X_i) \Rightarrow Des_D(Y_j) \text{ such that } X_i \cap Y_j \neq \Phi \text{ for } X_i \in C^* \text{ and } Y_j \in D^* \quad (3)$$

A rule is said to be deterministic if $X_i \subseteq Y_j$ i.e. $X_i \cap Y_j = X_i$, otherwise a rule is non deterministic. The set of all decision rules $\{\tau_{ij}\}$ for all classes $Y_j \in D^*$ is called the decision algorithm of the information system S (Slowinski, et al. 2005; Komorowski, et al. 1998).

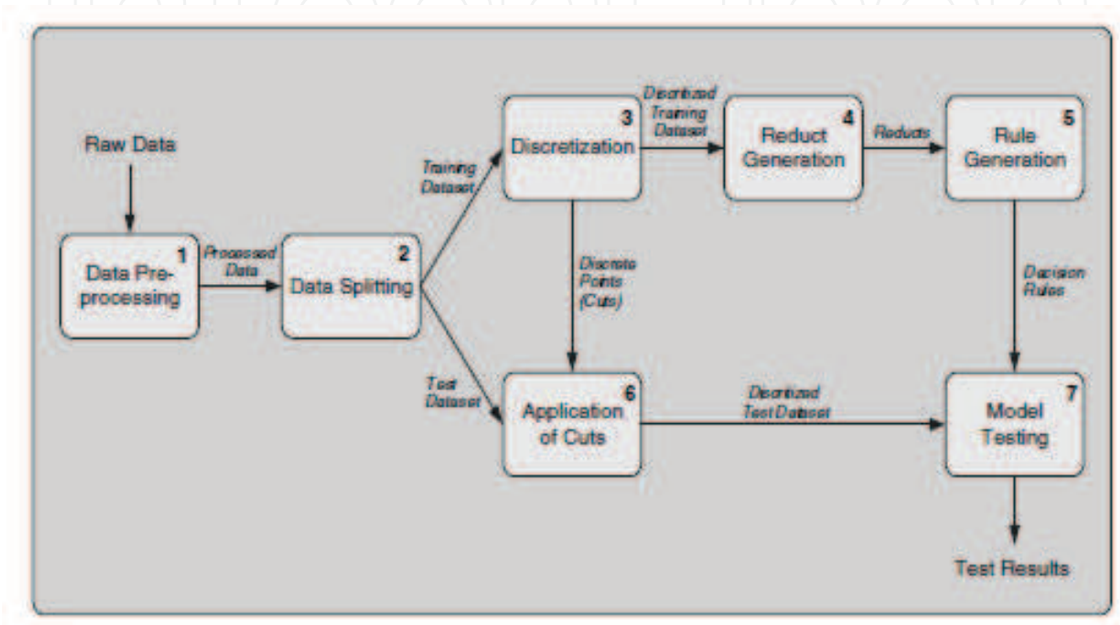


Fig. 5. Process map and the main steps of the rough sets analysis (Olson & Delen, 2008)

The rough sets algorithm implemented in MASCE can be summarized as follows:

This algorithm takes as input a decision table S = (U, C∪ D, V, f) and produces as output the set of decision rules $\{\tau_{ij}\}$ (Attia at al., 2004).

Step 1: Vertical reduction: The vote value is calculated for all the tuples (similar tuples are collapsed into one and their number is added to the vote). Then tuples, with vote values less than the noise filter threshold, are removed from the database table.

Step 2: Horizontal reduction: Attributes reduction is made by calculating the best reduct RED as follows, let all attributes be called AR and the user preferred attributes if any be UA.

Begin

2.1. Construct the modified discernibility matrix M(C):

Each entry m_{ij} contains the condition attributes whose values are not identical on both x_i and x_j where x_i, x_j belong to different classes of IND(D) i.e. they represent different decision concepts. M(C) is a diagonal symmetric matrix.

$m_{ij} = 0$ if $x_i, x_j \in$ same IND(D)

$= \{c \in C: f(c, x_i) \neq f(c, x_j)\}$ if $x_i, x_j \in$ different IND(D)

2.2. Find the CORE from discernibility matrix:

For any $c \in C$, $c \in CORE(C)$ if and only if there exists i, j, $1 \leq j < i \leq N$ such that $m_{ij} = \{c\}$. Note that a core may be empty

2.3. Determine the attribute set UA which user prefers to emphasize. If UA is empty that means that the user does not have preference for any attribute.
2.4. Let RED = CORE ∪ UA
2.5. AR = AR – RED
2.6. Find attribute a in AR which has the maximum SGF(a, RED, D)
2.7 RED = RED ∪ {a_i}, AR = AR – {a_i} (i =1, 2,…m)
2.8 If k(RED, D) = 1, then stop, otherwise go to step 2.6

/*End of Step 2 */
Step 3: Generate the reduced relation by removing those attributes which are not in the best reduct RED.
Step 4: Combine similar tuples in the reduced relation.
Step 5(a): Transform tuples in the reduced relation into decision rules for each class in D.
Step 5(b): For the same class in the reduced table, two tuples can be combined if the values of the condition attributes differ in only one attribute, thus obtaining a more general set of decision rules.
Or instead of steps 5(a) and 5(b) we can use the following alternative method for generation of decision rules:
Step 6(a): Extract the decision rule which is called τij as follows: $\tau_{ij} = Des_C(X_i) \Rightarrow Des_D(Y_j)$ such that $X_i \cap Y_j \neq \Phi$ for $X_i \in C^*$ and $Y_j \in D^*$.
Step 6(b): Call A rule (deterministic) if $X_i \subseteq Y_j$ i.e. $X_i \cap Y_j = X_i$, otherwise a rule is non-deterministic. The set of all decision rules $\{\tau_{ij}\}$ for all classes $Y_j \in D^*$ is called the decision algorithm of the information system S.
/* End of the algorithm*/
The significance of an individual attribute {a} added to the set A with respect to the dependency between A and D (Decision set) is represented by significant factor SGF, given by:

$$SGF(a, A, D) = k(A+\{a\}, D) - k(A, D) \text{ where } k(A, D) = card(POS_A(D)) / card(U) \quad (4)$$

Complexity of the chosen algorithm

This algorithm can learn a set of decision rules from databases efficiently and effectively. A basic role is played by the reduct and the core. Intuitively, a reduct of the relation is its essential part, which defines all basic concepts occurring in the considered data, whereas core is its most important role. Reducing a relation consists of removing irrelevant or superfluous attributes in such a way that the set of elementary categories in the relation is preserved. This procedure eliminates all unnecessary data from the relation, preserving only useful data. Thus concise, accurate decision rules are derived from the reduced relation.
Suppose there are N tuples in the database which is relevant to the learning task and A attributes for each tuple. The time complexity for the worst case is analyzed as follows. During reduction, to construct the discernibility matrix, it takes O(N×N) steps. To search core attributes in the discernibility matrix costs O(N×N) steps. The best reduct or user minimal attribute subset can be found in A × O(N×N) steps. Since A is usually much less than N, the worst case in the reduction process is O(N×N).
In this research work, we introduce the idea of using Rough Sets as a data mining technique to find the candidate helpers for collaboration with peers. In our case, the condition attributes C are the parameters of the student model explained in Section 3 such as quiz

results, peer evaluation, skill level and number of help sessions ignored. The decision parameter D is whether the student is a candidate helper or not. Each of the family of equivalence classes X_1, X_2,….X_r describes a group of students having the same values for different parameters in their models and thus can be candidate helpers for each other. They can be matched together in a one-to-one help session where each of the helper and the helpee profits the other (Mahdi & Attia, 2008 b).

4.3 Decision trees mining in MASCE

Decision tree induction is a well-known discipline in Machine Learning presented by Quinlan in 1986 (Quinlan, 1986). The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner. In the process of constructing a tree, the criteria of selecting test attributes influences the classification accuracy of the tree. Presently, there are many criteria for choosing the test attribute in building decision tree, such as ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) which use an entropy-based measure known as information gain as a heuristic for selecting the attribute.

Decision trees represent a supervised approach to classification. A decision tree is a simple structure where non terminal nodes (internal) represent tests on one or more attributes and terminal (leaf) nodes reflect decision outcomes. The ordinary tree consists of one root, branches, nodes (places where branches are divided) and leaves. In the same way the decision tree consists of nodes which stand for circles, the branches stand for segments connecting the nodes. A decision tree is usually drawn from left to right or beginning from the root downwards, so it is easier to draw it. The first node is a root. The end of the chain "root - branch - node-...- node" is called "leaf". From each internal node (i.e. not a leaf) may grow out two or more branches. Each node corresponds with a certain characteristic and the branches correspond with a range of values. These ranges of values must give a partition of the set of values of the given characteristic (Quinaln, 1986).

Constructing decision tress

The problem of constructing a decision tree can be expressed recursively. First, select an attribute to place at the root node and make one branch for each possible value. This splits up the example set into subsets. If at any time all instances at a node have the same classification, stop developing that part of the tree. The only thing left to decide is how to determine which attribute to split on. There are a number of possibilities for each split. Which is the best choice? The number of yes and no classes are shown at the leaves. Any leaf with only one class—yes or no—will not have to be split further, and the recursive process down that branch will terminate (Witten & Frank, 2005).

Because we seek small trees, we would like this to happen as soon as possible. If we had a measure of the purity of each node, we could choose the attribute that produces the purest child nodes. The measure of purity that we will use is called the information and is measured in units called bits. Associated with a node of the tree, it represents the expected amount of information that would be needed to specify whether a new instance should be classified yes or no, given that the example reached that node.

ID3

One of the oldest decision tree algorithms is ID3. It was designed for when there are many attributes and the training set contains many objects, but where a reasonably good decision

tree is required without much computation. The basic structure of ID3 is iterative. ID3 adopted an information based method that depends on two assumptions. Let C contain p objects of class P and n of class N. The assumptions are:
(1) Any correct decision tree for C will classify objects in the same proportion as their representation in C. An arbitrary object will be determined to belong to class P with probability p/(p + n) and to class N with probability n/ (p + n).
(2) When a decision tree is used to classify an object, it returns a class. A decision tree can thus be regarded as a source of a decision 'P' or 'N', with the expected information needed to generate this decision given by

$$l(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n} \tag{5}$$

If attribute A with values [A_1, A_2 . . . . A_v] is used for the root of the decision tree; it will partition C into [C_1, C_2 . . . . C_v] where C_i contains those objects in C that have value A_i of A. Let C_i contain p_i objects of class P and n_i of class N. The expected information required for the subtree for C_i is $I(p_i, n_i)$. The expected information required for the tree with A as root is then obtained as the weighted average:

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i) \tag{6}$$

where the weight for the i^{th} branch is the proportion of the objects in C that belong to C_i. The information gained by branching on A is therefore:

$$\text{gain}(A) = I(p, n) - E(A) \tag{7}$$

A good rule of thumb would seem to be to choose that attribute to branch on which gains the most information. ID3 examines all candidate attributes and chooses A to maximize gain(A), forms the tree as above, and then uses the same process recursively to form decision trees (Witten & Frank, 2005).

Complexity of ID3 Algorithm

At each non-leaf node of the decision tree, the gain of each untested attribute A must be determined. This gain in turn depends on the values p_i and n_i for each value A_i of A, so every object in C must be examined to determine its class and its value of A. Consequently, the computational complexity of the procedure at each such node is O(|C|.|A|), where |A| is the number of attributes above. ID3's total computational requirement per iteration is thus proportional to the product of the size of the training set, the number of attributes and the number of non-leaf nodes in the decision tree. The same relationship appears to extend to the entire induction process. No exponential growth in time or space has been observed as the dimensions of the induction task increase, so the technique can be applied to large tasks (Zhao & Zhang, 2008).
The main problem of ID3 is that a sub-tree may repeat several times in a decision tree, and that an attribute may be used for several times in some certain paths of the tree ,which degrades the efficiency of classification. Missing values pose an obvious problem. It is not clear which branch should be taken when a node tests an attribute whose value is missing. Sometimes, missing value is treated as an attribute value in its own right. If this is not the

case, missing values should be treated in a special way rather than being considered as just another possible value that the attribute might take. A simple solution is to record the number of elements in the training set that go down each branch and to use the most popular branch if the value for a test instance is missing.

4.4 Integrated Rough Sets and Entropy (RSE) algorithm in MASCE

Rough Set Entropy (RSE) algorithm for combining Rough Sets and Entropy heuristics is used to induce classification rules. This algorithm is based on Information gain and equivalence relation. It is applied to discrete-valued attributes. We divide the algorithm into three stages (Yang et al., 2003):

(1) First, we select a condition attribute based on information gain; let us call the selected condition attribute c.

(2) Second, use the concepts of rough sets to build equivalence classes. For our knowledge representation system J = (U, C ∪ D), subset P={c, d} ⊆ C ∪ D (c is the selected condition attribute in step (1), d is the decision attribute) determines an equivalence relation in U: IND(P)={(u, w) ∈ U x U: q(u) = q(w) for every q∈ P} where the number of U/IND(P) is no more than k x m (k is the number of values of c, m is the number of values of d).

(3) Finally, classification rules can be extracted from equivalence classes, for each equivalence class, we get classification IF-THEN rule by extracting attribute values which are identical for all the samples in the equivalence class, we use the condition attribute values as the rule antecedent and use the class label (decision) attribute value as the rule consequent.

Complexity of the Integrated RSE Technique

Comparing RSE with traditional Rough Sets, RSE is simpler because we do not need to build discernibility matrix, nor reduct calculations which can be very time consuming instead of large number of attributes and samples. If we use the RSE, we only need selecting a condition attribute once at the root node, so its time complexity is O(A*N), then it uses rough set theory to establish the equivalence classes then extract classification rules (Yang et al., 2003).

Suppose that the total number of training sample is n, the number of condition attributes is a. If we use ID3 algorithm to get a decision tree, at the worst case, the maximal length from the root node to each leaf node is a, so the total node number of decision tree is less than A*N. At the root node, ID3 algorithm requires analyzing each sample for each condition attribute is O(A*N). For the other nodes, the time complexity is less than root node, so the worst complexity is O(A * N * A *N) i.e. $O(A^2 * N^2)$.

5. Results and analysis

Whenever the student using the system needs help in a certain topic in a certain course, he can either choose among three different machine learning techniques so that MASCE can help him to find the candidate helpers: rough sets, decision trees or an integrated approach Rough Sets Entropy (RSE) combining both rough sets and decision trees. These different classifiers are applied every 12 hours on the decision table - a snapshot of it is shown in Table 1-after it has been updated with students' data collected during the last 12 hours such that the extracted decision rules are always up to date. The decision attribute is Decision with binary values {Yes, No}.

ID	topic	is online	willing to help	maximum concurrent discussions	is preferred agent	is banned agent	is banned topic	quiz result	help evaluation	help ignore	decision
1	Word	Y	N	N	N	Y	N	F	VG	P	N
2	Presentation	Y	N	N	N	Y	N	F	F	P	N
3	XML	Y	Y	Y	N	Y	N	G	F	P	N
4	Power Point	Y	Y	N	N	Y	N	F	E	F	N
5	HTML	N	Y	N	N	Y	N	VG	F	E	N
6	Java	Y	N	Y	Y	Y	Y	G	F	P	N
7	Java Script	Y	N	N	Y	Y	Y	F	P	F	N
8	Power Point	Y	Y	N	Y	N	N	VG	F	VG	N
9	Power Point	N	N	N	N	N	Y	P	P	P	N
10	PDF	Y	Y	Y	N	Y	Y	F	F	F	N
11	PDF	Y	Y	Y	N	N	Y	F	G	E	N
12	UML	N	Y	Y	Y	Y	N	VG	E	P	N
13	PDF	N	Y	N	Y	N	Y	G	F	P	N
14	Java	Y	Y	Y	N	N	Y	F	P	Et	N
15	UML	N	Y	Y	Y	Y	N	F	G	P	N
16	Programming	Y	Y	N	N	Y	N	F	P	F	N
17	Programming	N	Y	Y	N	N	N	F	G	P	Y
18	Java	N	Y	N	Y	N	N	VG	E	P	N
19	Oracle	N	N	Y	N	N	Y	E	VG	VG	N
20	Power Point	N	Y	Y	N	N	Y	G	VG	E	N

E = Excellent, VG = Very Good, G= Good, F=Fair, P = Poor

Table 1. Snapshot (only 20 records) of MASCE Decision Table (100 records) filled with random data

The major criticism of rough sets is that it requires all of the variables to be in nominal format. Rough sets cannot work with numerical continuous valued variables. In order to use such variables, one should perform discretization. Discretization, is a process of converting continuous numerical variables into range labels. We had to assign range labels for some of the condition attributes which have numerical continuous values such as: quiz results, help

evaluation and help ignore. The condition attributes of the decision table after discretization are as follows:

- Topic, with possible values {Java, Oracle, Java Script, HTML, UML, PowerPoint, Presentation, PDF, Programming, Documentation and Word}
- Willing to help, with binary values {Yes, No} but filled according to the assumption the 75% of the students are willing to help while only 25% are not willing to help. This assumption agrees with data obtained when trying system with real users
- Help ignore, with possible values {Poor, Fair, Good, Very Good, Excellent} following a normal distribution with mean 25 and standard deviation 10 for those who are willing to help but ignoring help requests.
- Maximum concurrent discussions, with binary values {Yes, No}
- Is preferred agent, with binary values {Yes, No}
- Is banned agent, with binary values {Yes, No}
- Is banned topic, with binary values {Yes, No}
- Quiz results, with possible values {Poor, Fair, Good, Very Good, Excellent} according to normal distribution with mean 65 and standard deviation 15.

Help evaluation, with possible values {Poor, Fair, Good, Very Good, Excellent} with normal distribution dependent on the result of the quiz-result, i.e. mean of this normal distribution is the result of the quiz-results according to the assumption that high quiz results implies higher probability of good peer evaluation.

5.1 Results of Rough Sets classifier

Applying Rough Sets Algorithm to MASCE decision table to determine whether a student is a good candidate for help or not. We first tried the Rough Sets Classifier on MASCE decision table consisting of 100 records filled with random data as shown in Table 1. Applying RS algorithm for finding best reduct, it was found to be: best reduct = {willing to help, maximum concurrent discussions, is banned user, is banned topic}. Using these attributes to build decision rules used for classification new unseen cases asking for candidate helpers, we obtained the following decision rules:

willing to help	maximum concurrent discussions	is banned agent	is banned topic	decision	vote	count of attributes
Y	Y	N	N	Y	10	4
Y	Y		Y	N	9	4
N		Y	Y	N	7	4
Y	N	N	Y	N	7	4
	N	N	N	N	8	4
N	N	N		N	3	4
Y	Y	Y	N	N	11	4
N		N	Y	N	3	4
Y	N	Y	Y	N	6	4

Table 2. Extracted Decision Rules from Rough Sets Classifier

5.2 Results of Decision Trees (ID3) classifier

Choosing to apply decision trees as the machine learning technique on the same decision table to find candidate helpers, we obtained the following decision tree as shown in Table 3.

willing to help	maximum concurrent discussions	is banned agent	quiz result	topic	decision	vote	count of attributes
		Y			N	56	1
Y	Y	N		Documentation	Y	2	4
N	Y	N		Documentation	N	1	4
	N	N		Documentation	N	4	3
Y		N		HTML	Y	2	3
N		N		HTML	N	1	3
Y		N		Java	Y	1	3
N		N		Java	N	1	3
		N	Very Good	Java Script	N	1	3
		N	Poor	Java Script	Y	1	3
		N	Fair	Java Script	N	3	3
		N	Excellent	Java Script	N	1	3
	Y	N		Oracle	Y	2	3
	N	N		Oracle	N	2	3
		N		PDF	N	3	2
		N		Power Point	N	3	2
		N		Presentation	N	2	2
		N		Programming	N	3	2
		N		UML	N	4	2
		N		Word	N	2	2
	Y	N		XML	Y	2	3
	N	N		XML	N	2	3

Table 3. Extracted Decision Rules from Decision Trees Classifier

5.3 Results of Integrated Rough Sets Entropy (RSE) classifier

Choosing to apply an integrated approach (Rough Sets Entropy) to the same decision table for finding the candidate helpers, we obtained the root node for the decision tree as (is

banned agent). Using this root node and following the RSE algorithm we obtained the following decision rules shown in Table 4.

willing to help	maximum concurrent discussions	is banned agent	is banned topic	decision	vote	count of attributes
		Y		N	55	1
Y	Y	N	N	Y	10	4
		N		N	44	1

Table 4. Extracted Decision Rules from the Integrated Rough Sets Entropy

5.4 Evaluating deducerd ules

We often need to compare a number of different learning methods on the same problem to see which one is the best to use. It seems simple: estimate the error (using different estimation procedures that will be explained shortly), perhaps repeated several times, and choose the scheme whose estimate is smaller. This is quite sufficient in many practical applications: if one method has a lower estimated error than another on a particular dataset, the best we can do is to use the former method's model. However, it may be that the difference is simply caused by estimation error, and in some circumstances it is important to determine whether one scheme is really better than another on a particular problem. This is a standard challenge for machine learning researchers. If a new learning algorithm is proposed for a certain problem, it must be shown that it improves on the state of the art for the problem at hand and demonstrate that the observed improvement is not just a chance effect in the estimation process.

It is important that the testing data that will be used was not used in any way to create the classifier. As we did in our research, we tried out several learning schemes on the training data and then we want to evaluate them to see which one works best. Other measurements are: the time taken during mining, the complexity of algorithm, and the coverage of the rules. The performance of a classification system cannot be based only on the higher accuracy but the quality of knowledge such as minimum number of rules, rule length, and rule strength also need to be assessed. A good rule set must has a minimum number of rules and each rule should be short as possible. Moreover, an ideal model should be able to produces fewer rule with shorter rule and classify new data with good accuracy.

A comparative study is carried out for the three used classifiers: Rough Set Classifier (RC), Decision Tree Classifier (DTC) and the integrated RSE in terms of accuracy, rule number, rule length and rule coverage. Integrated Rough Sets and Entropy classifier (RSE) outperformed both Rough Sets (RS) classifier and Decision Tree Classifier (ID3) since it performs smaller number of simpler, shorter rules (less number of attributes) as well as a higher coverage.

5.5 Predicting performance

Repeated cross-validation

The question of predicting performance based on limited data is an interesting, and still controversial, one. There are different techniques but the repeated cross-validation is probably the evaluation method of choice in most practical limited-data situations.

The standard way of predicting the error rate of a learning technique given a single, fixed sample of data is to use stratified n-fold cross-validation. The data is divided randomly into n parts in which the class is represented in approximately the same proportions as in the full dataset. Each part is held out in turn and the learning scheme trained on the remaining nine-tenths; then its error rate is calculated on the holdout (test) set. Thus the learning procedure is executed a total of n times on different training sets (each of which have a lot in common). Finally, the n error estimates are averaged to yield an overall error estimate. Tests have also shown that the use of stratification improves results slightly. Thus the standard evaluation technique in situations where only limited data is available is stratified n-fold cross-validation (Witten & Frank, 2005).
Different n-fold cross-validation experiments with the same learning method and dataset often produce different results, because of the effect of random variation in choosing the folds themselves. Stratification reduces the variation, but it certainly does not eliminate it entirely. When seeking an accurate error estimate, it is standard procedure to repeat the cross-validation process 10 times and average the results. This involves invoking the learning algorithm n*n times on datasets. Obtaining a good measure of performance is a computation-intensive undertaking (Witten & Frank 2005).

Leave-One-Out

Leave-one-out cross-validation is simply n-fold cross-validation, where n is the number of instances in the dataset. Each instance in turn is left out, and the learning method is trained on all the remaining instances. It is judged by its correctness on the remaining instance, one or zero for success or failure, respectively. The results of all n judgments, one for each member of the dataset, are averaged, and that average represents the final error estimate. This procedure is an attractive one for two reasons. First, the greatest possible amount of data is used for training in each case, which increases the chance that the classifier is an accurate one. Second, the procedure is deterministic: no random sampling is involved. There is no point in repeating it 10 times, or repeating it at all: the same result will be obtained each time.
On the other hand, its disadvantage is the high computational cost, because the entire learning procedure must be executed n times and this is usually quite infeasible for large datasets. Another disadvantage is that it cannot be stratified, worse than that, it guarantees a non-stratified sample.

Bootstrapping

An alternative, called **Bootstrapping**, involves sampling with replacement to choose elements of the training and test sets. Starting with n training examples, first sample n times, **with replacement**, to give another set (a bag, actually) of n examples. There will nearly always be duplicates in the training set; use the elements not in the training set as a test set. Compute error = 0.632 * test-error + 0.368 * training-error. Repeat steps 1-3 and output average accuracy. These coefficients come from the probability that a particular example will not be picked in n tries is as follows:

$$\left(1-\frac{1}{n}\right)^{n} \cong e^{-1} = 0.386 \tag{8}$$

thus the test set will contain roughly 36.8% of the examples; the training set 63.2% on the average.

Cost of evaluation

In the two-class case with classes yes and no, a single prediction has the four different possible outcomes TP, TN, FP and FN. The True Positives (TP) and True Negatives (TN) are correct classifications. A False Positive (FP) occurs when the outcome is incorrectly predicted as yes (or positive) when it is actually no (negative). A False Negative (FN) occurs when the outcome is incorrectly predicted as negative when it is actually positive (Witten & Frank, 2005).

Confusion matrix

In a multiclass prediction, the result on a test set is often displayed as a two dimensional confusion matrix with a row and column for each class. Each matrix element shows the number of test examples for which the actual class is the row and the predicted class is the column. Good results correspond to large numbers down the main diagonal and small, ideally zero, off-diagonal elements.

	Predicted +	Predicted -
Actual +	True Positives (TP)	False Negatives (FN)
Actual -	False Positives (FP)	True Negatives (TN)

Table 5. Confusion Matrix Parameters

The performance measures can consist of any or all of the following properties of this matrix (Witten & Frank, 2005):

Property	Formula	Interpretation
Accuracy	(TP+TN)/(TP+TN+FP+FN)	Proportion of classifications that were correct
Sensitivity	TP/(TP+FN)	Proportion of actual positives that were predicted that way
Specificity	TN/(TN+FP)	Proportion of actual negatives that were predicted that way
Positive Predictivity	TP/(TP+FP)	Proportion of predicted positives that really were
Negative Predictivity	TN/(TN+FN)	Proportion of predicted negatives that really were

Table 6. Different Performance Measures

We chose to use *Leave-One-Out* technique for testing. As was mentioned earlier, Leave-One-Out is most suitable for small size datasets. Each instance of the dataset is left out, and the learning method is trained on all the remaining instances. It is judged by its correctness on

the remaining instance, one or zero for success or failure, respectively. The results of all n judgments, one for each record of the dataset, are averaged, and that average represents the final accuracy estimate.

Confusion matrix using rough sets classifier

	Predicted +	Predicted -
Actual +	TP = 10.0	FN = 0.0
Actual -	FP = 17.0	TN = 73.0

Table 7. Confusion Matrix using Rough Sets Classifier

Confusion matrix using decision trees classifier

	Predicted +	Predicted -
Actual +	TP = 8.0	FN = 2.0
Actual -	FP = 10.0	TN = 80.0

Table 8. Confusion Matrix using Decision Trees Classifier

Confusion matrix using integrated rough sets and entropy classifier

	Predicted +	Predicted -
Actual +	TP = 0.0	FN = 10.0
Actual -	FP = 0.0	TN = 90.0

Table 9. Confusion Matrix using Integrated Rough Sets and Entropy

Comparison of performance measures for three classifiers

Property	Rough Sets	Decision Tree	Rough Sets & Decision Tree
Accuracy	0.83	0.88	0.90
Sensitivity	1.00	0.80	0.00
Specificity	0.82	0.88	1.00
Positive Predictivity	0.37	0.44	Not a Number (as both TP and FP = 0)
Negative Predictivity	1.00	0.98	0.90

Table 10. Comparison of Performance Measures for Three Classifiers

Analyzing the previous data it can be shown that regarding *Accuracy* (Proportion of classifications that were correct) RSE classifier (with 90% accuracy) outperformed both RS classifier (with 83% accuracy) and DT classifier (with 88% accuracy). Also, regarding

Specificity (proportion of actual negatives that were predicted that way), RSE classifier (with 100% specificity) outperformed both RS classifier (82% specificity) and DT classifier with (88% specificity).
On the other hand, regarding *Sensitivity* (i.e. proportion of actual positives that were predicted that way) RSE scored 0 as it was not able to predict any actual positives, while RS classifier had sensitivity of 100% followed by DT classifier with 80% sensitivity.

6. Conclusions and future work

The use of Data Mining techniques in a Multi-Agent System for Collaborative E-Learning (MASE) is introduced. A comparative study is carried out for the three used classifiers: Rough Set Classifier (RC), Decision Tree Classifier (DTC) and the integrated RSE Classifier in terms of accuracy, sensitivity and specificity.
Regarding accuracy & specificity measures, the Integrated Rough Sets and Entropy (RSE) Classifier outperformed the two other techniques.
Quality of the knowledge extracted is also compared in terms of rule number, rule length and rule coverage. Integrated Rough Sets and Entropy classifier (RSE) outperformed both Rough sets (RS) classifier and Decision Tree Classifier (ID3) since it performs smaller number of simpler, shorter rules (less number of attributes) as well as a higher coverage.
Comparison of the two techniques, Rough sets and ID3, in general terms is very difficult. These two methods use different classification criteria. Rough sets is typically based on relations between condition and decision attributes, concepts of positive and boundary region, core and reducts. On the other hand, decision trees (ID3) uses entropy (Information gain) for the classification process. Another difference is the way to represent the derived knowledge. Rough sets uses information tables where as ID3 uses decision trees. Certain kinds of problems are best represented by tables, others by trees while some need a totally different type of data structures. When the sample size is small or when the underlying distribution of data deviates significantly from multivariate normal distribution, rough sets may perform better than decision trees since there is no assumption on the data size and the distribution. Rough sets may also perform better when the data is imprecise, incomplete. To summarize, there is no best approach for all problems. That is why instead of a single method, a hybrid approach benefiting from the combined strengths of both rough sets and decision trees is used and which gave good accuracy and fewer, simpler, shorter rules with high coverage. So according to these experiments, the integrated RSE approach is the most suitable (among the three techniques investigated) for application in our system MASCE.
We find that the decision rules constructed by the integrated method are much simpler in structure (less number of rules and shorter rules) than the constructed decision rules extracted by the other two classifiers. Since the complexity of this algorithm is much less, the computation time is much less. Furthermore, they have higher classification accuracy. All this indicators lead to the conclusion that the integrated rough sets and entropy approach is the most suitable one among the implemented three machine learning techniques to be used in data mining in Multi-Agent System for Collaborative E-learning (MASCE).
We are thus providing an online web mining opposite to the offline web mining which is an aftermath analysis that could give some hints on how an on-line course is effectively used and how its structure could be improved. We are presenting in this research an integrated web mining where the patterns automatically discovered are used to assist learners in their on-line learning. In other words, mined patterns are used on-the-fly by the system to

improve the application or its functions. We claim that this is quite a promising approach that successfully combines machine learning techniques with agent technology in e-learning systems in order to provide higher quality services towards the end users of e-learning systems (both students and instructors).

This study showed that advanced data mining methods could successfully be used to help decision making in multi-agent systems with a relatively high degree of accuracy. However, in the context of predictive accuracy, one should be aware of several issues that delimit the applicability and predictive accuracy of data mining models: (i) the nature of the data (including the richness, correctness, completeness and representation of the data itself), (ii) the data mining methods (including their capabilities and limitations to handle different types and combination of data) and (iii) the application domain (including understandability and availability of related factors needed to develop accurate predictive models).

The models (extracted decision rules in our case) are only as good and as predictive as the data used to build them. One of the key determinants of predictive accuracy is the amount and quality of the data used for a study. Although all of the models built in this study provided an acceptable level of prediction accuracy based on variables at hand, their performance levels could be improved by including more relevant variables.

It is recommended that more (and relevant) variables be added to the model when possible to increase the accuracy levels. It is also recommended to test the three classifiers using different techniques other than Leave-One-Out such as Bootstrapping and n-fold cross validation and comparing the results with those obtained here as a future work to this research. Different data mining techniques can also be tested and their results compared with the three classifiers studied in this research.

7. References

Agent Working Group (2000). *Agent Technology, Green Paper*, Technical report, version 1.0, Object Management Group, available at: http://www.jamesodell.com/ec2000-08-01.pdf, last accessed 12/12/2010

Attia, S. S., Mahdi H. K. and Mohammad, H. K. (2004). Data Mining in Intelligent Tutoring Systems using Rough Sets, *Proceedings of the 2004 International Conference on Electrical, Electronic and Computer Engineering (ICEEC'04*), Cairo, Egypt

Buse. D. P. and Wu, Q. H. (2007). *IP Network-based Multi-Agent Systems for Industrial Automation*, Springer-Verlag, London

Chakrabarti, S., Cox, E., Frank E. et al. (2009). *Data Mining: Know It All*, Morgan Kaufmann Publishers, Burlington, MA

Chen, S. Y., Chiu, M. L. (2005). *Building an Agent-Based System for E-Learning in Digital Design*, *Computer-Aided Design Applications*, Vol. 2

Cios, K. J., Pedrycz W. and Swiniarski, R. W. (1998), *Data Mining Methods for Knowledge Discovery*, Kluwer Academic Publishers

Jo, Ch., Chen, G., Choi, J. (2004). A New Approach to the BDI Agent-Based Modeling, *Proceedings of the 2004 ACM Symposium on Applied Computing*

Komorowski, Pawlak, Z., Polkowski, L. and Skowron, A. (1998). Rough Sets: A Tutorial, in *Rough-Fuzzy Hybridization: A New Method for Decision Making*, Pal S. K. and Skowron, A. (Ed.), Springer-Verlag

Mahdi, H. K., Attia, S. S. (2008 a). Prioritizing of MAS Architectures for Developing Collaborative E-Learning, *Proceedings of the International Conference on Technology Communication and Education*, Kuwait

Mahdi, H. K., Attia, S. S. (2008 b). Small Dataset Size Clustering in MASCE, *Proceedings of the 2008 International Conference on Frontiers in Education: Computer Science and Computer Engineering*, Las Vegas, Nevada, USA

Mohsin, M., Abd Wahab, M. (2008). Comparing the Knowledge Quality in Rough Classifier and Decision Tree Classifier in Information Technology, In *International Symposium on Information Technology 2008*, Kuala Lumpur, Malaysia

Nwana, H. S. (1995). Software agents: An overview, Knowledge Engineering Review, 11(2):205-244

Olson D. L., Delen D. (2008). *Advanced Data Mining Techniques,* Springer-Verlag, Berlin, Heidelberg

Qin, B., Xia, Y., Prabhaka, S., Tu, Y. (2009). A Rule-Based Classification Algorithm for Uncertain Data, in *IEEE International Conference on Data Engineering*.

Quinlan, R. (1986). Induction of Decision Trees, *Machine Learning*, Vol. 1, No. 1, Kluwer Academic Publishers, Boston, 1986

Quinlan, R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers

Sellers, B. H. and Giorgini, P. (2005). *Agent-Oriented Methodologies*, Idea Group Inc.

Shang, Y., Shi, H. and Chen, S. (2001). An Intelligent Distributed Environment for Active Learning, *ACM Journal of Educational Resources in Computing*, Vol. 1, No. 2

Slowinski, R., Greco, S., Matarazzo, B. (2005). Rough Set Based Decision Support, in *Search Methodologies Introductory Tutorials in Optimization and Decision Support Techniques*, Burke, E. K., Kendall, G., (Ed.), Springer

Symeonidis, A. L., Mitkas P. A. (2005). *Agent Intelligence through Data Mining*, Springer Science + Business Media, Inc, USA

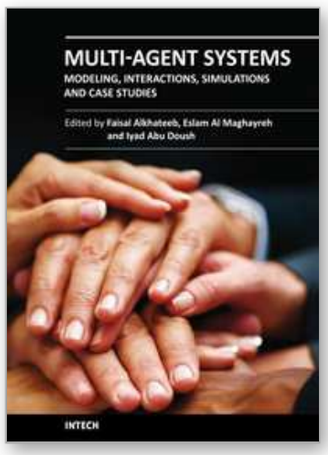
Wooldridge, M. (1999). Intelligent agents, In *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, Weiss & Gerhard, (Ed.), The MIT Press, Cambridge, MA, USA

Wooldridge, M. (2002). *Introduction to Multi-Agent Systems*, John Wiley & Sons, first edition

Witten, I. H., Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan-Kaufmann, CA, USA

Yang, J., Wang, H., Hu, S. and Hu Z. (2003). A New Classification Algorithm based on Rough Set and Entropy, *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, November 2003

Zhao Y. and Zhang Y. (2008). Comparison of Decision Tree Methods for Finding Active Objects, *Advances in Space Research,* Vol. 41, No. 12



Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-176-3
Hard cover, 502 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hani K. Mahdi, Hoda K. Mohamed and Sally S. Attia (2011). Data Mining for Decision Making in Multi-Agent Systems, Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-176-3, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-interactions-simulations-and-case-studies/data-mining-for-decision-making-in-multi-agent-systems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen