I.J. Intelligent Systems and Applications, **2013, 04, 1-19**
Published Online March 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2013.04.01

A Rough Sets-based Agent Trust Management Framework

Sadra Abedinzadeh
Department of Computer Science, University of Regina, Regina, SK, Canada
Email:sadra@cs.uregina.ca

Samira Sadaoui
Department of Computer Science, University of Regina, Regina, SK, Canada
Email: sadaouis@uregina.ca

Abstract— In a virtual society, which consists of several autonomous agents, trust helps agents to deal with the openness of the system by identifying the best agents capable of performing a specific task, or achieving a special goal. In this paper, we introduce ROSTAM, a new approach for agent trust management based on the theory of Rough Sets. ROSTAM is a generic trust management framework that can be applied to any types of multi agent systems. However, the features of the application domain must be provided to ROSTAM. These features form the trust attributes. By collecting the values for these attributes, ROSTAM is able to generate a set of trust rules by employing the theory of Rough Sets. ROSTAM then uses the trust rules to extract the set of the most trusted agents and forwards the user's request to those agents only. After getting the results, the user must rate the interaction with each trusted agent. The rating values are subsequently utilized for updating the trust rules. We applied ROSTAM to the domain of cross-language Web search. The resulting Web search system recommends to the user the set of the most trusted pairs of translator and search engine in terms of the pairs that return the results with the highest precision of retrieval.

Index Terms— Theory of Rough Sets, Trust Management, Multi Agent Systems, Trust-based Service Selection

I. Introduction

1.1 Problem Definition

Trust management in open systems plays a key role in today's growing need for such systems [1]. On the other hand, openness is one of the key requirements for businesses. One of the most important aspects of an open environment is the fact that there are different providers who are able to offer a particular type of service. The service providers are autonomous entities, i.e. there is no supervision on their actions. As a result, the open systems are usually considered as Multi Agent Systems (MASs) [2]. A MAS is a society of intelligent agents in which each agent acts autonomously in a goal oriented manner to achieve the final goal of the system. Although having an open system in which different providers can join and leave at any time and without any supervision might seem promising at first glance, it suffers from the truth that some of these service providers, or agents, may provide incorrect information about the service(s) they offer. In other words, some of them might not be trustworthy and, therefore, the task should not be assigned to them. In the field of e-commerce for instance, an untrustworthy agent can list a product that it doesn't own. Therefore, a buyer agent may lose money without getting his desired service or product.

The final goal of this research is to recommend to the end-users a set of the best service agents, in terms of the most trustworthy ones, in an open system such as the Web. Without losing the generality of the subject, we describe the problem to be solved with a concrete example. Consider the case where you would like to find a specific type of information on the Web, let's say the result of a soccer match between Manchester United and Chelsea in 2003. What would you do? This is the most basic question this research is trying to help the users to answer. One may employ a general search engine, Google for example, to search for this information. Then, by improving the query and, as a result, reducing the number of returned results, he can get the most relevant information. On the other hand, another individual might be able to find useful information on the Web sites of those soccer clubs because the information he is seeking is related to two specific soccer clubs. Another user may browse sport related Web sites to search for such information.

In fact not everybody knows the best way to find the most relevant information using the least number of steps. The huge number of service providers on the Web causes an individual to spend a lot of time analyzing the links and pages he is presented with as the result of a Web search. In addition, there are other factors that may affect this issue. As an example,

consider the fact that there is a lot of information available on the internet in a language that the user might not be able to understand, yet they may contain useful information. For instance, it will be beneficial for a researcher to know if there are any related works reported in the past in a language different than his. As a general description, consider a case where there are several, sometimes hundreds or thousands of, services available to the user to choose from. It looks impossible for the user to be able to select "the best" service amongst them. The question is how to define the term "the best" in the previous sentence. This question cannot be answered in a general form, i.e. it is not possible to define "the best" if the characteristics of the systemand the user's request are not known.

Given a specific application, this paper answers the above question by employing the theory of Rough Sets to find the set of best agents in terms of the most trusted ones by analyzing a number of metrics based on which the services will be classified. These metrics can vary from the time required to complete a task to the degree of reputation that service provider already has. Then, by invoking an analysis process, based on the theory of Rough Sets, over the values of these metrics, named evidences in this paper, a number of implication rules, in the form of if-then, are generated. These rules, ultimately, are used to indicate how trustworthy each service provider is based on the collected evidences.

1.2 ROSTAM: The Proposed Solution

This paper introduces ROSTAM, a general framework for managing trust for MASs based on the theory of Rough Sets. The term ROSTAM consists of two parts: "ROS" stands for ROugh Sets and "TAM" is derived from "Agent Trust Management". "Rostam" is also the name of an ancient Persian hero [3]. The major contributions of ROSTAM are twofold: at first, ROSTAM returns a set of the most trusted agents for each request submitted by the user; secondly, it updates its trust knowledge based on the interaction rating provided by the user for each trusted agent.

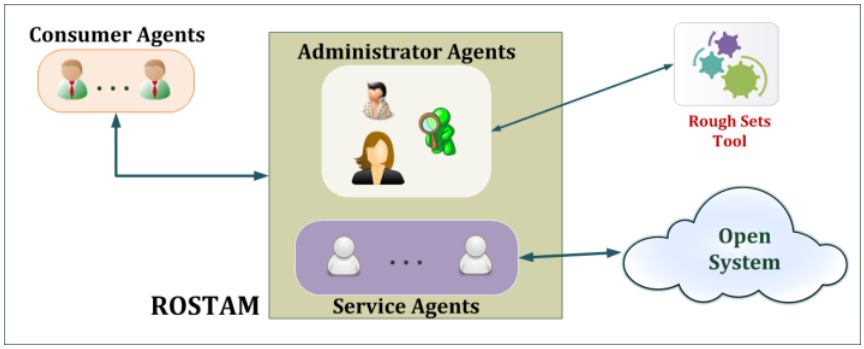

Fig. 1: ROSTAM as a Middleware

As illustrated in Figure 1, ROSTAM plays the role of a middleware between the Open System, where the service providers are located, and the Consumer Agents, those agents willing to use the services in the Open System. ROSTAM comprises of two major components: Service Agents that constitute the MAS needed to implement the Open System, and Administrator Agents that manage trust for Service Agents. In addition, we couple ROSTAM with an external Rough Sets Tool that provides the functionalities of Rough Sets theory. To the best of our knowledge, this is the first time the Rough Sets theory is applied to the trust management field [4].

ROSTAM helps the user find the set of the most trusted service providers according to the features of Service Agents and user's request. In fact, all the agents in such a set are equally trusted. To this end, ROSTAM utilizes the theory of Rough Sets for a specific application with the following steps.

- First, the developer who wants to use ROSTAM for a specific application identifies the metrics based on which the service agents of the application are to be evaluated as trusted. We name these metrics "trust attributes". These attributes are derived from the properties of the user, the user's request and the Open System. One of the attributes must be selected as the decision attribute which is used to assess the trustworthiness of service agents, i.e. the higher the value of the decision attribute, the more trustworthy the agent.
- Second, ROSTAM collects the values of trust attributes. We name these values trust evidences. Different agents in the system, such as Consumer Agents, Administrator Agents and Service Agents, provide the values for the trust attributes.
- Third, ROSTAM applies the theory of Rough Sets to the collected trust evidences to generate the "trust rules". These rules are then utilized to extract the set of the most trusted service agents.
- Fourth, the set of trusted agents are then employed to process the user's request. The results are shown to the user once they are returned. At this point, the user rates the interaction with each agent in the set of the most trusted ones.

- Fifth, the interaction rating values along with the values of other trust attributes are gathered and stored in a repository. Then, the theory of Rough Sets is utilized to generate the new set of trust rules according to the available trust evidences.

The rest of the paper is organized as follows. Section II is an overview of the related works and the core concepts of Rough Sets. Section III presents the trust management process as well as the generic software architecture of ROSTAM and its underlying agents. Section I applies ROSTAM to the field of cross language Web Search by implementing all the agents of ROSTAM. Section V describes an experiment based on two phases: training and execution. Section VI summarizes the contributions of this work and reports future directions of this research.

II. Background

2.1 The Theory of Rough Sets

The theory of Rough Sets introduced by Pawlak [5] may be used as a means for processing data in an environment in which there might not be a certain definition for objects, i.e. the incomplete information about objects will result in vague concepts. For this purpose, the theory of Rough Sets makes use of a set of equivalence equations to define the indiscernibility of every object in the Universe of Discourse U (the set of all objects).

2.1.1 Approximations

Let O be the collection of objects in U. One can partition these objects into an approximation space, A, using an equivalence relation R, i.e. $A = (O, R)$. These equivalence classes are named elementary sets or atoms. As a result, if two objects a and b belong to the same elementary set A, then a and b are indistinguishable in A. Note that R can be a multi-dimensional equivalence relation in which there are more than one attribute of objects being considered. Let $o \in O$. In Rough Sets, the following two approximations can be defined for o in the space A. One can use these definitions to roughly approximate the subsets of O as follows:

- $\bar{A}(o)$ is the smallest set of objects in A containing o (upper approximation/negative region).
- $\underline{A}(o)$ is the largest set of objects in A which are contained in o (lower approximation/positive region).

2.1.2 Information Table

Different attributes of R along with the objects can be represented in a form of an information table in which each column denotes one of the attributes in R as well as the object identifier, and each row the value for the attributes for each object. For example, consider O as the set of patients visiting a clinic in a specific week, and R as the relation consisting of the age and gender of the person as well as whether or not the patient coughs, has fever, has flu, and has allergy to a specific type of medicine. A sample information table for this system is given in Table I. Suppose we define $o = \{x \in O | Age(x) < 40 \; and \; Coughs(x) = Yes\}$. In this case, $\underline{A}(o) = \{P_1, P_n\}$ which denotes that P_1 and P_2 belong to the equivalence class that is characterized by "*Age <40*" and *"Coughs=Yes*".

Table I: Sample Information Table

Id	Age	Gender	Has Fever	Coughs	Has Allergy	Has Flu
P_1	28	Male	No	Yes	No	Yes
P_2	47	Female	No	No	Yes	No
...	...	...	...	...	...	...
P_3	38	Male	Yes	Yes	No	Yes

2.1.3 Rough Sets and Decision Making

One possible application of the theory of Rough Sets is to find if there exists a relation between a particular set of attributes with another set of attributes in an information table. This is known as a decision making problem where the values for a subset of attributes can be used to assess the values of another subset [6]. In other words, it is possible to decide whether or not the values of a set of target attributes, named decision attributes, can be identified if the values of other attributes, called condition attributes, are known. According to this definition, one solution to this problem can be achieved if a rule-base exists such that it relate different values of the decision attribute, i.e. the choice set, to the values of condition attributes, i.e. the criteria. Therefore, given a criteria, one could easily identify if an object belongs to a decision set. For instance, suppose we would like to figure out if a particular patient, in the information table mentioned above, has flu according to the values of other attributes (this is also known as disease diagnostics). Thus, we would define "Has Flu" as the decision attribute, and the rest of attributes as the condition attributes.

2.1.4 Reduct

An intuitive question is to know if all the columns in an information table carry useful information or not.

Specially, in the case of decision making, some columns may impact the decision more than others, and some may have no influence. The theory of Rough Sets introduces the concept of Reduct [7] to address this issue. Reduct is a process than results in a set of attributes which are *"jointly sufficient"* [7] and *"individually necessary"* [7] for evaluating the values of a specific attribute. For instance, suppose we want to find the positive region of "*Has Flue = Yes*". Also, suppose that P_1 and P_n are the only objects with "*Has Flue = Yes*". In this case, with regard to the information in Table I, the attributes "*Gender*", "*Has Allergy*" and "*Coughs*" will play no role in finding this positive region since they all have the same value for all the objects in that region. Consequently, the reduct of attributes, w.r.t. "*Has Flue = Yes*", is the following set of attributes: $\{Age, Has\ Fever\}$. In Rough Sets, a reduct *R*, w.r.t a set of decision attribute, *D*, and a set of condition attributes, C, can be formalized as follows.

$$\underline{A}_R(O_D) = \underline{A}_C(O_D)$$

AND

$$\forall a \in R,\ \underline{A}_{R-\{a\}}(O_D) \neq \underline{A}_C(O_D) \quad (1)$$

where $\underline{A}_X(O_D)$ reveals, for a decision attribute, *D*, the positive region of a set of objects, *O*, with respect to a subset of condition attributes, X ($X \subseteq C$). The first statement in this relation represents *"jointly sufficient"*, and the second one denotes the *"individually necessary"*. For instance, according to the information table in Table I, one can identify if a particular patient "Has Flu" only by knowing the values of "Age" and "Has Fever" attributes (jointly sufficient). However, it will not be possible to decide whether or not the patient has flu if only the value for one of these two attributes is known (individually necessary).

2.2 Applications of the Theory of Rough Sets

Theory of Rough Sets has been applied to many different domains. Authors of [8] provide a broad survey of applications of rough Sets to various fields of Civil engineering such as pavement engineering, water resource management, land management, etc. There are also a lot of Rough Sets-based approaches reported in the literature in the field of pattern recognition. In [9], a hybrid method is introduced that combines neural networks with Rough Sets in the field of pattern recognition. It employs the Rough Sets as an objective function as well as a stochastic method to extract the features from a state space. Moreover, it utilizes neural networks to handle the uncertainty of the system. Another application is reported in [10] where the authors propose a prototype system that makes use of Rough Sets, Boolean algebra, and genetic algorithms in the field of inductive rule learning. This prototype is able to find inconsistency in empirically gathered collections of data. An interesting application of rough sets is introduced in [11] in which they examine the facilities of a building for the performing arts by generating a set of 140 decision rules based on 6 condition attributes. The decisions rules, for instance, reveal that while the backstage amenities are of less importance to the dance performers, clear voice is the least important attribute for the drama performers. In [12], a Rough Sets-based decision making problem for addressing the issue of mapping the users' likings and the components used in designing system is proposed. The issue is due to the fact that the raw data providing such mapping is usually uncertain and non-linear. Their method is successfully finding the hidden patterns that can be used to lead the design. Authors of [13] present a method for online identifying of signatures. Using a training database of different signatures, 31 attributes were extracted for each signature. Then, it calculates the reduct of these attributes which results in 9 attributes. The authors also state that the classification rate is 100% in their experiment. In the field of information retrieval, Rough Sets has been employed for instance in [14]. It proposes an approach based on Rough Sets and Fuzzy Sets to address the vocabulary mining problem. The approach makes it possible to apply different views of vocabulary set in an IR system. Rough Sets has also been applied in the field of clustering. The work in [15], for instance, reviews some of methods that make use of Rough Sets for clustering in the areas such as traffic data (clustering highway sections), and clustering supermarket customers. Using Rough Sets to predict the failure of Chinese companies has been explored in [16] where the authors employ the variable precision Rough Sets model to find the impact of financial and non-financial parameters on companies' performance. To this end, the paper produces a set of rules, based on 13 different conditional attributes, which define if a company will be classified as ST (special treatment).

2.3 Trust Based Service Selection

Methods of Web service selection have been widely reported in the literature. An agent-based framework is introduced in [17] that facilitates the process of dynamically selecting web services by means of an ontology that defines the QoS parameters from both objective (e.g. response time) and subjective (e.g. by concentrating on the user's interaction) perspective. The research reported in [18] identifies the challenges of calculating trust and proposes SCOUT as a middleware to collect the evidences, and a client interface that once implemented makes it possible for the consumer to submit the evidences to the middleware layer. SCOUT helps a software system to easily compute the degree of trust for its entities. The evaluated degrees, then, can be used to select the best service. Authors of [19] propose a trust based service selection method. In this model, the degree of trust that a consumer has in a service is calculated based on both direct and indirect rating for that service. The direct rating is the assessment of the service after the consumer finishes the interaction, and indirect one is provided by third parties. The paper employs the Dempster-Shafer theory of evidence to

calculate such a trust value. As a result, the model is able to overcome the problem of transitive trust evaluation as well as to remove fake evidences from the assessment process. proposes an intuitionistic fuzzy sets-based approach is proposed in [20]. Using an improved version of fuzzy indexing method (proposed in the paper), the authors are able to combine both "concord" and "discord" satisfaction rates in order to assess the satisfaction degree of the service provider. Moreover, a weighting scheme is employed to prioritize different QoS metrics. Authors of [21] introduce a method for evaluating trust and reputation of a service provider in a decentralized user modeling system. The proposed method takes into account two metrics: a number of measures with regard to the QoS of service provider, and the degree of trust over each of these measures in the case where they are provided by other agents that show how much the user trust the referrers values in that context. Finally, by taking into account the defined weights as well as adjusting the referred values according to degree of trust to the provider, the consumer agent is able to generate his degree of trust in a service provider. The work in [22], in opposition to feedback rating-based approaches that introduce subjectivity, proposes a reputation management method for selecting web services in three steps: testing, identifying malevolent, and tuning feedbacks. Their simulation shows improvement in the precision of reputation calculated for an entity.

III. Designing ROSTAM

In this section, we first describe the process for producing the trust rules based on Rough Sets, and then present the generic software architecture of ROSTAM and its underlying software agents.

3.1 A Trust Management Process

3.1.1 Defining Trust Attributes

The first step is to define the trust attributes for the application being developed with ROSTAM. These attributes are the metrics based on which the trustworthiness of service agents are to be evaluated. The developer may identify the set of attributes by firstly examining the features of the request and of each service agent (either human or software). The attributes might be static or dynamic. The values collected for a dynamic attribute may be different each time they are accessed whereas a static attribute will always have the same value. Consider the Web search system where "Supporting Persian Language" is one of the attributes. This is a static attribute because its value does not change over time. On the other hand, the value for "Precision of Retrieval" attribute is dynamic since it might be different for different result sets. Secondly, he needs to determine the domain of each attribute and map it into a set of discrete values. The values can be either fuzzy or crisp. The values of an attribute with a continuous domain must also be discretized. ROSTAM has no limitation on the number of attributes and the domain of their possible values. Yet, a large number of intervals or attributes may result in a longer processing time. Finally, one of these attributes should be chosen as the decision attribute. The rest will form the set of condition attributes. To select the decision attribute, one should answer the following question:

"Which attribute should have higher value in order to trust an agent more than another one?"

In the field of trust management, we define the decision attribute as the one that must be used to evaluate the degree of trust for each service agent. As reviewed in sections 2.2 and 2.3, there are many trust attributes that can be considered such as reputation, popularity, and average response time. For instance, in an IR application, one may determine the following set of attributes: reputation, retrieval method, language of query, average precision of retrieval and domain of query. He might choose the reputation as the decision attribute, i.e. the higher the reputation of the IR system, the more trustworthy the system. However, another IR application might be interested in defining the decision attribute as the average precision of retrieval. In this situation, for two service agents A and B, with P_A and P_B as their values of average precision of retrieval respectively, if $P_A > P_B$, then A is more trustworthy than B.

3.1.2 Collecting Trust Evidences

The second step deals with collecting the trust evidences, i.e. a collection of the values of the trust attributes. ROSTAM gathers the trust evidences from different sources such as Consumer Agent and Agent Management System. ROSTAM stores the trust evidences in the Evidence Repository. The result of this step is the information table as described in Section 2.1. For instance, in an IR application, a collection of documents as well as a set of queries over that collection may be employed to collect the trust evidences.

3.1.3 Analyzing Trust Evidences

The final step is to analyze the evidences collected in step 2. To this end, ROSTAM employs Rough Sets to generate the trust rules. Trust Rules are implication statements that state how the decision attribute is related to the condition attributes according to the information table. ROSTAM will then extract from these trust rules the set of the most trusted service agents.

Consider as an example an IR system. Suppose we have chosen the average precision of retrieval as the decision attribute. A sample trust rule, in this case, may look like the following:

$$\begin{bmatrix} (DomainOfQuery = Sport) \\ AND \\ (IRModel = VectorSpace) \end{bmatrix} \Rightarrow \{AvgPrecison = Low\}$$

where:

$$DomainOfQuery \in \{Sport, Science, Economy\}$$
$$IRModel \in \{VectorSpace, Probabilistic\}$$
$$AvgPrecision \in \{Low, Medium, High\}$$

This rule states that, for a given query, if the domain is "*Sport*" and the IR algorithm that is used is "*VectorSpace*", the average precision of retrieval is "*low*".

3.2 System Workflow

The following process takes place for each request submitted by a Consumer Agent to ROSTAM.

- ROSTAM determines the set of trusted service agents based on the current trust rules as well as the values of trust attributes (w.r.t the properties of request, Consumer Agent, and Service Agents).
- The request is forwarded to the Service Agents in the set of trusted agents.
- The results are shown to the Consumer Agent who, in turn, provides feedback in terms of rating the interaction with each Service Agent in the set of trusted agents.
- ROSTAM collects the interaction rating values as well as the values of trust attributes, and appends them to the existing list of trust evidences.
- ROSATM utilizes a Rough Sets tool to produce a set of new trust rules based on the new list of trust evidences.

3.3 Generic Software Architecture

Figure 2 shows the detailed architecture of ROSTAM that manages the trust for a number of Service Agents using the Administrator Agents. Below, we describe all the components of ROSTAM.

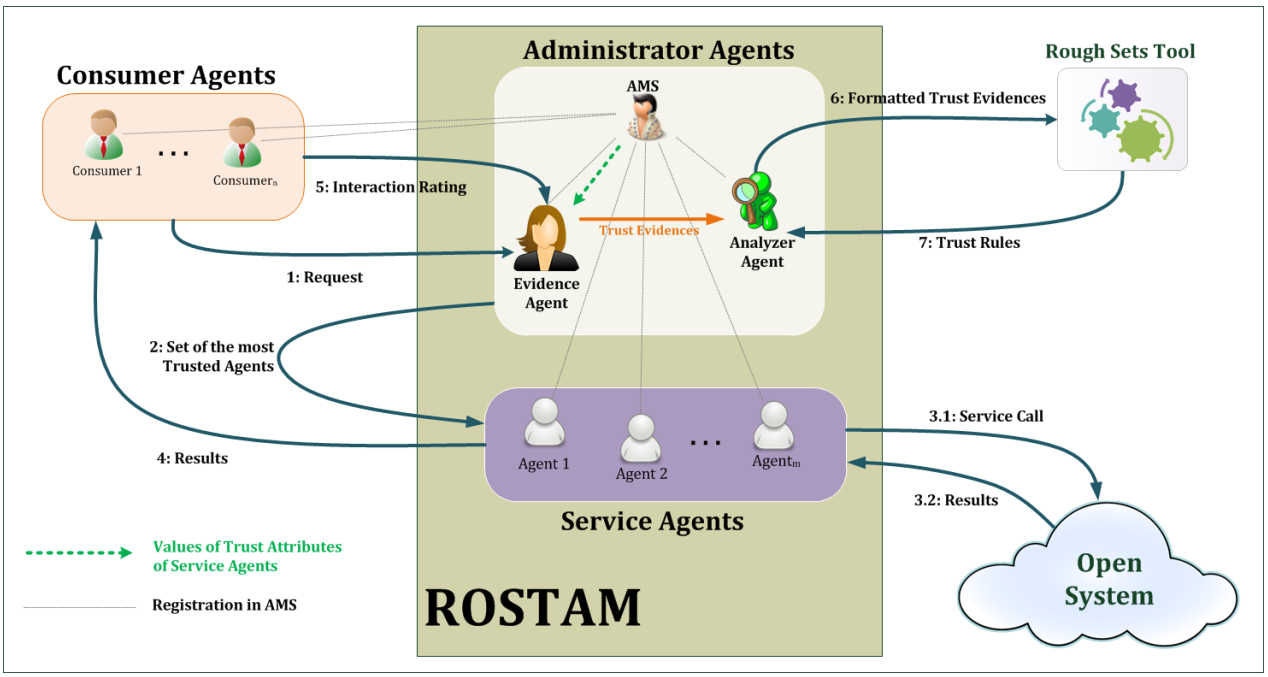

Fig. 2: Generic Software Architecture of ROSTAM

3.3.1 Consumer Agents

Consumer Agents are those agents (human or software agent) willing to make use of the services provided in the Open System. A Consumer Agent issues a request to ROSTAM, and then provides a feedback in terms of rating the interaction with each trusted service provider that has returned results.

3.3.2 Service Agents

For each service provider in the Open System, there exists a Service Agent who is responsible for communicating with that provider. Each service agent changes the format of the request according to its corresponding service provider. Moreover, after getting the results back from that provider, it modifies the format of the results to follow a unique interface usable by the Consumer Agent. As a result, regardless of which service provider will be utilized, the Consumer Agent will have a unique interface to provide their requests, and a distinctive format for the results they get.

3.3.3 Administrator Agents

In our architecture, there are three administrator agents.

3.3.3.1 Agent Management System

One common sub-system in a MAS is the Agent Management System (AMS) [23]. AMS acts as both White and Yellow pages in the society of agents. Each agent upon the time of joining or leaving the society registers itself in the AMS by providing proper information that can uniquely identify that agent. Thus, if an agent wants to know which other agents are currently in the society, or equally if a specific agent is already in the society, it simply queries the AMS. This is the White page service. Moreover, each agent registers the type of services it provides. This will constitute the Yellow Page service. In other words, any agent can query the AMS to find out which agents in the society offer a particular service. The AMS will then respond with a list that may contain no, one or a number of agents offering that service. As a result, in ROSTAM, AMS can be utilized to acquire the values of those trust attributes that are related to Service Agents.

As an example consider a Web search application where the system has access to a number of search engines. Regarding the White Page service, we have the case where an agent wants to find out if it can get the results from Google. However, Yellow Page service might be used in the situation where an agent wants to know which agents can accept a query in Persian language.

3.3.3.2 Evidence Agent

This agent is responsible for performing the second step of our model: collecting the trust evidences. The Evidence Agent must be able to gather and store the values for both dynamic and static attributes. These values are collected from different sources such as AMS and Consumer Agents.

The Evidence Agent, after collecting values, stores the trust evidences in the Evidence Repository. ROSTAM exposes no limitation on the media used to store the trust evidences. However, Evidence Agent must be capable of providing the trust evidences as an information table.

3.3.3.3 Analyzer Agent

Analyzer Agent gets the current trust evidences from Evidence Agent, and changes their format into a representation usable by the Rough Sets tool being used. It then utilizes the Rough Sets Tool to acquire the set of trust rules based on the evidences, and stores them. One may argue that how often the Analyzer Agent should generate the new set of trust rules. One option is to produce the trust rules after each iteration of workflow is completed, i.e. whenever there are new objects being added to the trust evidences. An alternative to this approach is to gradually update such rules. There are a number of researches that target this problem [24-28]. For instance, [24] performs a preliminary process on the information table which results in a consistent decision table (named ∂-decision table). Then, through the algorithm presented in the paper, the rules are learned incrementally by using a matrix that is built based on the ∂-decision table.

3.3.3.4 Rough Sets Tools

Analyzer Agent employs a Rough Sets tool to convert the trust evidences into trust rules. There are many Rough Sets Tools, such as RSES [29], ROSETTA [30], RIDAS-a [31], Rough Enough [32], GROBIAN [33] and ROSE [34], that can be utilized. The Rough Sets tool firstly calculates the reduct of the trust attributes. As described before, this will eliminate those attributes that do not affect the decision attribute. Secondly, it generates the trust rules according to the collected evidences.

3.4 A Summary

Figure 3 summarizes the ROSTAM workflow. For $Request_t$, ROSTAM finds the set of the most $TrustedAgents_t$ by employing the $TrustRules_{t-1}$, i.e. the trust rules generated w.r.t. $Request_{t-1}$. Then, the values of trust attributes w.r.t. $Request_t$ are stored in the $TrustEvidence_t$. Note that $TrustEvidence_t$ contains all evidences available in $TrustEvidence_{t-1}$ in addition to the one collected w.r.t. $Request_t$. The theory of Rough Sets is then utilized to generate the set of $TrustRules_t$ that are used later for finding trusted agents w.r.t. $Request_{t+1}$.

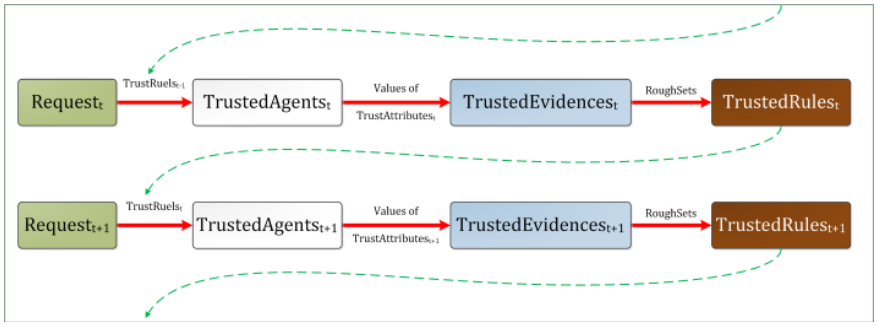

Fig. 3: Summary of Trust Management Workflow

IV. Applying ROSTAM to Cross Language Web Search

We apply ROSTAM to the field of Web search. We also support cross-Language search in which the language of the query and language of the results can be different. We employ a machine translation method [35] for translating the user's query to the target language if required. For this purpose, we make use of two online translation services: Google translate[1] and Babylon[2].

Search engines are a kind of Information Retrieval (IR) systems [36]. However, the collection of documents in a search engine is constantly changing. Thus, to assess how well a search engine could find relevant information, one can utilize some of the metrics for evaluating IR systems, such as precision and recall [37]. Precision is defined as the ratio of the number of relevant documents retrieved to the number of all documents retrieved whereas recall is the ratio of number of relevant documents retrieved to the number of all relevant documents in the collection. According to these two definitions, it will not be possible to calculate the value for recall measure in a Web search system due to the fact that the number of all relevant documents is unknown. Therefore, in this research, we only use the precision factor. Moreover, from the IR perspective, we consider the system of our case study as a high precision system where the ultimate goal is to increase precision as much as possible.

The goal of our Web search application is to find the set of the most trusted pairs of translators and search engines in terms of the pairs that have higher precision results. All the pairs in this list are equally trusted in the sense that the precision of the results they return is the same. For this purpose, ROSTAM produces a set of trust rules that are generated as the result of a Rough Sets analysis and show how the value of precision of retrieval, as the decision attribute, is related to the values of other trust attributes identified specifically for the Web search application.

4.1 Universe of Discourse and Objects

The goal of applying ROSTAM to a cross-language Web search is to find those pairs of translators and search engines that result in a higher precision of results in the target language for a given query in the source language. In other words, we would like to classify the combination of queries, translators and search engines and find a set of rules that demonstrates the relation between such combinations with the precision of results. We consider each of these combinations as an individual object. Hence, from the Rough Sets perspective, the set of objects O can be defined as:

$$O = \bigcup < Query, Translator, SearchEngine >$$

4.2 Defining Trust Attributes for Web Search

Here we define the trust attributes w.r.t. our Web search application. To this end, we conducted a survey to identify the trust attributes. We also provide the discretized values of all trust attributes including condition and decision attributes.

4.2.1 A Survey on Trust Attributes

In order to figure out how people evaluate the trust of Web services in the real world and which attributes they use to do so, we developed a survey [3] on the Web and asked a number of individuals to anonymously state which Web sites or services they choose with respect to the following tasks:

- Search Engine
- News Source (general news)
- Sports News
- Shopping
- Weather
- Email

In addition, they were asked to express the reasons why they select such services/Web sites. They are able to provide as much information as they like. Furthermore, we didn't offer any options or sample reasons in order to avoid biasing their opinions. Table II summarizes the results of this survey.

Table II: Summary of Trust Attributes Survey

Service	Reason
Search Engine	personal affection, fast, **support different languages**, popularity, offering different services, **best results**, user friendly
News (general)	popular, extensiveness, user friendly, **language**, locality
Sports	personal interest in a specific sport, uses general news sites, **up-to-date information**
Shopping	local shopping sites, reputation, extensiveness of products, ease of use
Weather	**information for different times**, **long-term forecast**, ease of access and use, reputation (rank in search engine)
Email	features like capacity, affiliation, ease of use and good user interface

[1] - http://translate.google.com/
[2] - http://translation.babylon.com

[3]- This survey has been approved on ethical grounds by the Research Ethics Board of University of Regina on Sep 6th, 2012 (File#: 04S1213).

According to the results of this survey and our case study of Web search, we have identified the following trust attributes: *Domain of Query*, *Update Frequency*, *Is Searchable By Date*. In addition, we need to consider *Search Engine* and *Translator* to show which search engine and translation service has been used. These attributes form the set of condition attributes. The decision attribute is the *Precision* of retrieval. This is because of the fact that we consider our case study of Web search as a high precision retrieval system.

The rational for choosing such trust attributes according to the results of the survey are as follows. The *Domain Of Query* is derived from the type of service that was considered in the survey, i.e. News, Sport, Shopping, etc. Also, "information for different times" and "long-term frequency" results in defining *Is Searchable By Date* attribute. Finally, "up-to-date information" leads us to consider the *Update Frequency* as a trust attribute. However, some other attributes were not relevant to our case study. For instance, due to the fact that our service agents communicate with the service providers in the Open System through their APIs, "User Friendly" will not be taken into account in our case study.

4.2.2 Condition Attributes

4.2.2.1 Domain of Query

Domain of query defines the type of information the user is looking for. Automatically identifying such piece of information has been widely reported in literature [38-40]. [38] depicts an approach to figure out what the user's objective is when he queries a search engine. The approach includes a classification mechanism based on the linguistic attributes of the query as well as the feedbacks collected from the target search engine. Another approach towards a semantic-based IR system is presented in [39]. The approach firstly explores the query to extract its semantic elements such as verb, subject, predicate, etc. Then, by summarizing the content, determining the semantics, and producing semantic patterns respectively, the semantics of the query is identified. [40] employs a n-gram approach to extract the token from SMSs submitted from a cellphone, and then uses those tokens to generate a set of rules that can result in determining the domain of those SMSs.

Table III: Discretizing values of DomainOfQuery Attribute

DomainOfQuery	Discretized Value
Sports	*1*
Science	*2*
Politics	*3*
Economics	*4*
Entertainment	*5*
General	*6*

However, in this case study, we do not detect the domain of the query automatically. Yet, the user chooses one from a list when he is issuing a query. Table III shows the domains along with their numerical representation.

4.2.2.2 Is Searchable By Date

For certain types of queries, a user may be interested in filtering the results by a date range, i.e. by providing the lower and upper bound of a date interval. For instance, if the user is looking for scholar documents, he may want to get the papers that have been published between 2008 and 2012. It will be more beneficial for the user if ROSTAM ranks higher the scholarly engines that support such filtering mechanism. The discretized values of this attribute are presented in Table IV.

Table IV: Discretized Values of IsSearchingByDate

IsSearchableByDate	Discretized Value
YES	1
NO	0

4.2.2.3 Update Frequency

The frequency of updating information in a service provider can affect how trustworthy that service is in the sense that the user may want to find the information that has just been created. For instance, in the case that the user is looking for news, he might be interested in following the news as they happen. Consequently, a news service with the capability of updating in real time will be more preferred (more trusted from ATM point of view) in this situation. However, the value of this attribute will not be important to the user if he is searching for information about an event that has happened in the past. We have identified the following update frequencies along with their values that are shown in Table V.

Table V: Discretized Values for UpdateFrequency Attribute

UpdateFrequency	Discretized Value
Unknown	0
Real Time	1
Hourly	2
Daily	3
Otherwise	4

4.2.2.4 Search Engine

This attribute indicates which search engine has been used to answer the user's query.

Table VI: Values of SearchEngine Attribute

SearchEngine	Discretized Value
Google	GOOGLE
Yahoo!	YAHOO
Bing	BING

4.2.2.5 Translator

This attribute indicates which translator agent has been employed to translate the user's query into English.

Table VII: Values of Translator Attribute

Translator	Discretized Value
Google Translate	1
Babylon	2

4.2.3 Decision Attribute: Precision of Retrieval

In this paper, we have chosen the Precision of retrieval as our decision attribute. As mentioned before, ROSTAM for Web search is a high precision system, from IR perspective, that returns a set containing pairs of translator and search engine agents that will lead in results with higher precision. Therefore, the rules generated by ROSTAM, in the context of the theory of Rough Sets, must reveal the positive region of "Precision = HIGH". However, if this region is empty, i.e. there are no pairs that result in "Precision = HIGH", then ROSTAM must recursively reduce the level of precision (to MEDIUM, and then to LOW) till at least one pair is found. Precision of retrieval has a value between 0 and 1. Table VIII shows how we discretize the values of this attribute.

Table IX. Discretized Values for Precision Attribute

Precision	Discretized Value
[0, 0.33]	LOW = 1
[0.34, 0.66]	MEDIUM = 2
[0.67, 1]	HIGH = 3

4.3 Implementation of ROSTAM

In this research, the underlying MAS, composed of search engines and translators, is implemented with the CAPNET framework [41] in C# .Net. We also developed a GUI and Administrative Agents. CAPNET is a FIPA-Compliant[4]framework that makes it possible to incorporate agents written in other FIPA-Compliant frameworks such as JADE [23] and JACK [42] into our system.

[4] - http://www.fipa.org/

4.3.1 Service Agents for Web Search

This specific implementation of ROSTAM for Web search utilizes three search engines, Yahoo, Google, and Bing. The open APIs of these search engines allow ROSTAM to submit the queries and get the results back as a set of Web pages. For each of these search engines, we have implemented a separate agent that wraps its corresponding API. Moreover, for the translation purpose, we have implemented the APIs for accessing the Babylon[5] and Google Translate[6] services. Through these APIs, and by sending requests to their corresponding agents, we are able to translate the whole query from Persian into English.

4.3.2 A Rough Sets Tool

In this research, we use the RSES (version 2) [29] as our Rough Sets Tool. RSES V2 presents the rules in the following format:

$$(C_1 = V_{c_1}) \& \dots \& (C_n = V_{c_n}) \Rightarrow (D = \{V_D^1[M_D^1], \dots, V_D^m[M_D^m]\})$$

Where C_i represents the i^{th} condition attribute, V_{C_i} is the value of C_i, D denotes the decision attribute, V_D^j shows the j^{th} value of D, and $[M_D^j]$ depicts how many object in the information table comply with this rule w.r.t. V_D^j. The set $\{V_D^1[M_D^1], \dots, V_D^m[M_D^m]\}$ is called "descriptor" of the premise.

RSES V2 provides the consumer with an option to shorten the generated rules, in addition to the basic functionalities presented in the theory of Rough Sets. Rule shortening and generalization has been investigated by researchers [43-45]. In RSES, rule shortening is an advanced process that tries to condense the descriptor provided along with the premise of the rules [46]. There are fewer rules after a shortening process. The shortened rules are also more general, i.e. more objects comply with those rules. Nevertheless, the shortened rules are less precise in the sense that some incorrect objects may fulfill a rule. RSES exposes a tuning parameter named *Shortening Ratio* that defines a degree based on which the RSES tool can compensate losing the precision while gaining more generality and less rules.

V. Web Search Experiment

Our experiment includes two phases. In the training phase, we use a number of queries to generate the initial set of trust rules. To this end, we send each query to all the pairs of translators and search engines and judge all the results returning from them. In the execution phase, we first extract the set of the most trusted pairs of translator and search engines (based on the current set

[5] - http://translation.babylon.com/persian/to-english/
[6] - http://translate.google.com/

of trust rules and values of trust attributes w.r.t. the query) and then get the results only from these pairs.

5.1 Workflow of ROSTAM for Web Search

Figure 4 shows the workflow of ROSTAM w.r.t. cross language Web search case study. At first, the user, through the GUI component, submits a query. He also selects the domain of the query from the list in Table III. Evidence Agent stores the value of domain of query as well as the values of other trust attributes sent from AMS in the Evidence Repository. Subsequently, the Rule Matcher component returns the set of the most trusted pairs of translator and search engine and ROSTAM employs them to retrieve the results of the query. Each Service Agent utilizes its corresponding service provider in the Open System. After getting the result back from the Service Agents, the user judges each and every result by assigning a Boolean value that shows whether or not the result was relevant to our query. The precision of the retrieval, computed by GUI, is then stored by the Evidence Agent in the Evidence Repository. The collected evidences are then sent to the Analyzer Agent in order to obtain the set of trust rules by employing the Rough Sets tool (RSES V2 in this experiment).

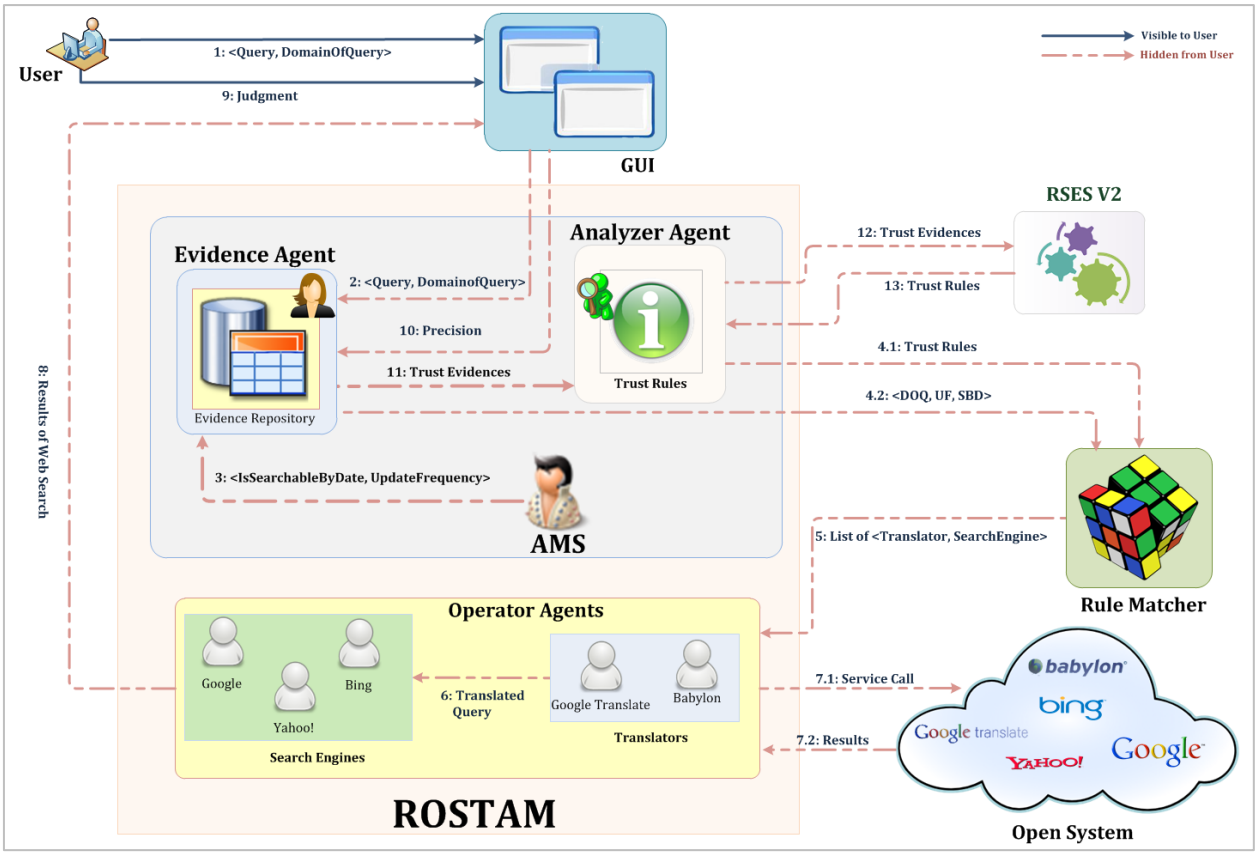

Fig. 4: Workflow of Web Search

5.2 Rule Matcher Component

The role of Rule Matcher component, as depicted in Figure 4, is to infer the list of the most trusted pairs of translators and search engines according to the current set of trust rules and the values of trust attributes w.r.t. the submitted query. As shown in Algorithm 2, it accepts as input the variable *E* which encapsulates the values for DomainOfQuery, IsSearchableByDate and UpdateFrequency attributes. It also gets the set of current trust rules from the Analyzer Agent as input.

Algorithm 1 begins by checking if Rule Matcher is operating in the training phase. In the training phase, Algorithm 1 returns all the pairs of translators and search engines. However, in the execution phase, it starts by setting the output variable *SetOfTrusted* to empty and initializing the *Red* variable to *DomainOfQuery* by considering the reduct given in Figure 5 (the variable *Red* includes all the attributes in the reduct set, Figure 5, except the *Translator* and *SearchEngine*). Then, it assigns to the set *HP* those trust rules with "HIGH" precision value. It iterates through all the rules *r* in *HP* and invokes Algorithm 2 by passing the variables *r*, *E*, and *Red*. Next, it adds the rules returned by Algorithm 2 to *SetOfTrusted* (note that Algorithm 2 may return no pairs. In this case, nothing will be added to the *SetOfTrusted*). In step 6, if *SetOfTrusted* is empty, i.e. there is no matching pair of

Translator and Search Engine added w.r.t. HIGH precision, it finds the pairs which result in a MEDIUM precision. Through the same iterative approach as in step 5, it adds the matching pairs to *SetOfTrusted*. In step 7, if *SetOfTrusted* is still empty, the algorithm tries to determine the matching pairs for LOW precision in the same way as step 5. However, if *SetOfTrusted* is still empty (i.e. there are no evidences that match the attributes of this query), the algorithm adds all the possible combinations of Translators and Search Engines to *SetOfTrusted*.

Algorithm 2 parses each rule to find the matching pair of translator and search engine according to the input parameters *r*, *E*, and *Red*. It first initializes variable *TmpList* to empty. Then, it checks whether or not the value of a_i w.r.t. *r* (denoted by $r.a_i$) equals the value of a_i w.r.t. *E* (referred to as $E.a_i$). If such condition holds, it adds the pair of Translator and Search Engine in the corresponding rule to *TmpList*. Note that if the value of $r.a_i$ is NULL (i.e. the value of a_i does not affect the value of decision attribute), the condition of *IF* statement in step 4.1 always holds. In other words, if $r.a_i$ is NULL, no comparison between $r.a_i$ and $E.a_i$ is performed and the execution moves to the body of *IF* statement. Similarly, when *r.Translation* or *r.SearchEngine* is NULL, it is implied that the choice of Translator or Search Engine does not affect the precision of retrieval. Therefore, all the Translator or Search Engine agents must be added to the output List accordingly (step 2 and step 3).

```
Input: TR //Trust Rules,    E //An object encapsulating the trust attributes
       IsTrainingPhase //Boolean
Output: SetOfTrusted //List of Trusted Pair <Translator, SearchEngine>
Resources: Reduct //Reduct of Attributes
{
   // In training phase, return all the pairs of translator and search engines
   1. IF (IsTrainingPhase)
        Return All pairs of <Translator, Search Engine>.
   2. SetOfTrusted = {}
   3. Red =  Reduct – {Translator, SearchEngine}
   // Finding pairs of <Translator, SearchEngine> that result in High Precision
   4. HP = {r ∈TR | r.Precision = HIGH} // High Precision list
   5. For each r in HP Do{
        5.1.  Let tmp = Algorithm 2(r, E, Red)
        5.2.  Add all elements of tmp to SetOfTrusted
   }
   // Finding pairs of <Translator, SearchEngine> that result in Medium Precision
   6. If SetOfTrusted is empty Then {
        6.1.  MP = {r ∈TR | r.Precision = MEDIUM} // Medium Precision list
        6.2.  For each r in MP Do
              6.2.1.  Let tmp = Algorithm 2(r, E, Red)
              6.2.2.  Add all elements of tmp to SetOfTrusted
      }
   // Finding pairs of <Translator, SearchEngine> that result in Low Precision
   7. If SetOfTrusted is empty Then {
        7.1.  LP = {r ∈TR | r.Precision = LOW} // Low Precision list
        7.2.  For Each r in LP Do
              7.2.1.  Let tmp = Algorithm 2(r, E, Red)
              7.2.2.  Add all elements of tmp to SetOfTrusted
      }
    // If SetOfTrusted is empty, add all the pairs of <Translator, Search Engine>
   8. If SetOfTrusted is empty {
        8.1.  For each Translator Ti and each Search Engine SEj
                 Add <Ti, SEj > to SetOfTrusted.
      }
   9. Return SetOfTrusted .
}
```

Algorithm 1: Rule Matcher


```
Input: r //a Trust Rules,   E //An object representing the trust attributes,
       Red //Reduct of Attributes
Output: TmpList //Temp List of Trusted Pair <Translator, SearchEngine>
Resources: Red //Reduct of Attributes
{
   1. TmpList = {}
   2. If (r.Translator == NULL)
         Then let r.Translator represent all Translator Agents.
   3. If (r.SearchEngine == NULL)
      Then let r.SearchEngine represent all Search Engine Agents.
   4. For Each attribute ai in Red Do{
        4.1. If  (E.ai == (Case When r.ai == NULL Then E.ai Else r.ai))
           Then  add <r.Translator, r.SearchEngine> to TmpList.
   }
   5. Return TmpList
}
```

Algorithm 2: Rule Parser

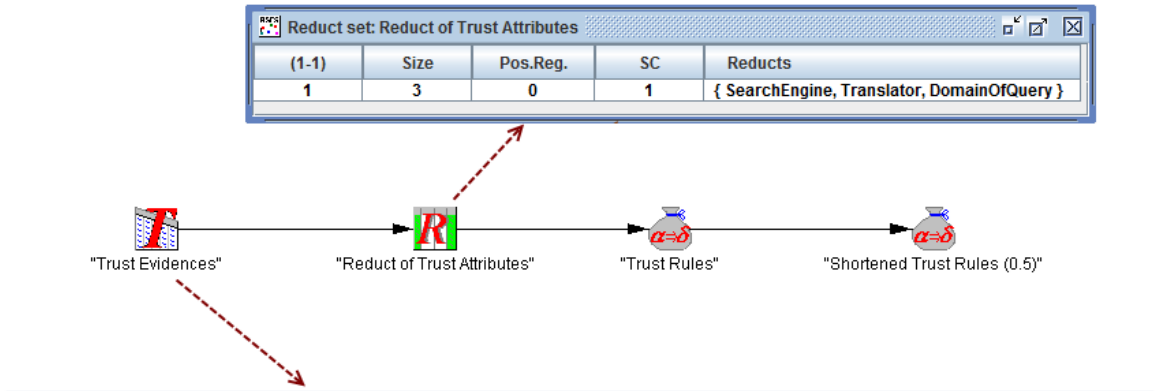

Table: Trust Evidences

240 / 6	SearchEngine	Translator	DomainOfQuery	IsSearchingForDate	UpdateFrequency	Precision
O:1	Yahoo	GoogleTranslate	2	1	0	3
O:2	Yahoo	GoogleTranslate	2	1	0	2
O:3	Yahoo	GoogleTranslate	6	1	0	1
O:4	Yahoo	GoogleTranslate	4	1	0	2
O:5	Yahoo	GoogleTranslate	4	1	0	1
O:41	Google	GoogleTranslate	2	1	0	1
O:42	Google	GoogleTranslate	2	1	0	3
O:43	Google	GoogleTranslate	6	1	0	1
O:44	Google	GoogleTranslate	4	1	0	2
O:45	Google	GoogleTranslate	4	1	0	3
O:81	Bing	GoogleTranslate	2	1	0	3
O:82	Bing	GoogleTranslate	2	1	0	2
O:83	Bing	GoogleTranslate	6	1	0	1
O:84	Bing	GoogleTranslate	4	1	0	3
O:85	Bing	GoogleTranslate	4	1	0	3
O:121	Yahoo	Babylon	2	1	0	2
O:122	Yahoo	Babylon	2	1	0	3
O:123	Yahoo	Babylon	6	1	0	3
O:124	Yahoo	Babylon	4	1	0	2
O:125	Yahoo	Babylon	4	1	0	1
O:161	Google	Babylon	2	1	0	3
O:162	Google	Babylon	2	1	0	2
O:163	Google	Babylon	6	1	0	2
O:164	Google	Babylon	4	1	0	2
O:165	Google	Babylon	4	1	0	1
O:201	Bing	Babylon	2	1	0	2
O:202	Bing	Babylon	2	1	0	1
O:203	Bing	Babylon	6	1	0	2
O:204	Bing	Babylon	4	1	0	3
O:205	Bing	Babylon	4	1	0	1

Fig. 5: Steps taken to generate initial trust rules

5.3 Training Phase

The training phase of our case study is described in this section. As mentioned before, the objects in this application are the combination of query, translator and search engine. In the training phase, the set of objects O are derived from 40 Persian queries [4]. For each query, we use two translator services (Google Translate and Babylon), and three search engines (Yahoo, Google, and Bing). Therefore, we have a total of $40 \times 2 \times 3 = 240$ objects which are used to generate the initial set of trust rules.

The training phase also addresses the Cold Start problem [47] which refers to the situation where the system cannot infer due to the lack of information. This problem usually happens at the beginning of the operation of the system. There are many researches in the literature that address the cold start problem [47, 48]. For example, [47] presents a similarity metric based on neural network learning to address the cold start problem w.r.t. users. It finds the similarity according to only the other users' rating information in the core process of content filtering. The approach in [47] introduces more risk and complexity comparing to other approaches that utilize more information such as those provided by the user's profile and user's tags.

The training phase produces an initial set of trust rules to be used in the execution phase. Therefore, the execution phase does not suffer from the cold start problem. On the other hand, in order to overcome the cold start problem in the training phase, we return all possible pairs of translator and search engines as the trusted ones. This is achieved as the result of first step in Algorithm 1.

5.3.1 Results

After collecting all the trust evidences for all queries, the evidences are sent to the Analyzer Agents who will generate the trust rules by employing the RSES V2 tool. Figure 5 shows the process of generating the rules. We first change the format of the collected evidences into an information table consumable by the RSES tool. Then, the RSES tool calculates the reduct of attributes in this table. In this experiment, the reduct contains three attributes as shown in Figure 5. The reduct will eliminate those attributes that have no impact on the value of the decision attribute (Precision of retrieval in this case study).

According to the reduct of attributes and trust evidences, the RSES tool is able to generate a set of trust rules given in Figure 6. Moreover, to shorten the trust rules as described in Section 4.3.2, RSES tool lets us choose a shortening parameter that has a value between 0 and 1 which defines how aggressively the rules must be shortened [29]. To select the most appropriate value for this parameter, we shortened the trust rules in Figure 6 using 9 different values for shortening ratio (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9). Table X reveals some statistics, provided by the RSES tool, for each shortening ratio. In this table, the value of "*Support of Rules*" is the number of objects satisfying the rules, and "*Mean Length of Premise*" is the mean of the number of attributes in the premise of the rules. A lower value of "Mean Length of Premise" means that the shortening process is using a larger degree of generalization for the condition attributes. According to Table X, shortening parameters 0.1, 0.2, and 0.3 are not appropriate because the attributes of the premise in the resulting rules will be very general because of very low mean length of premise.

On the other hand, we would like to use the shortening parameters with greater support for the rules. As stated before, support of the rules shows the number of objects that satisfy a rule. According to the values of support of rules, especially the Mean value, we do not choose shortening parameters 0.6, 0.7, or 0.8. Finally, to choose between shortening parameters 0.4 and 0.5, we consider the value of "Number of Rules". Preferably, we would like to have fewer rules in order to improve the performance of the Rule Matcher component (the Rule Matcher components must traverse all the rules). Therefore, we choose 0.5 as our shortening parameter.

Table X: Statistics for Different Shortening Ratios

Shortening Ratio	Number of Rules	Support of Rules <Min, Max, Mean>	Mean Length of Premise
0.1	11	<17, 120, 50.1>	1
0.2	11	<17, 120, 47.1>	1
0.3	15	<3, 93, 26.3>	1.4
0.4	26	<2, 50, 12>	2.1
0.5	17	<3, 42, 9>	2.6
0.6	6	<4, 13, 7.7>	2.5
0.7	5	<4, 10, 6.2>	2.8
0.8	3	<4, 7, 5.7>	3
0.9	0	N/A	N/A

Figure 7 depicts the rules after employing the shortening process by setting the shortening parameter to 0.5. Figure 6 and Figure 7 clearly demonstrate how we are able to reduce the number of rules (from 36 to 17 rules in this case study).

5.3.2 Results Analysis

Below, we briefly describe some of the trust rules presented in Figure 7. Rule #1, for instance, indicates that for a query in the General domain (*DomainOfQuery = 6*), the combination of "Google Translate" and "Yahoo" returns HIGH precision results. It also points out that there are 4 objects (combinations of query, translator, and search engine) in the information table (the collection of Trust Evidences) that satisfy such an implication. This is denoted as "*Precision={3[4]}*".

Rule set: "Trust Rules"

(1-36)	Match	Decision rules
1	7	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=2)=>(Precision={3[2],2[5]})
2	8	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=6)=>(Precision={1[3],3[4],2[1]})
3	5	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=4)=>(Precision={2[4],1[1]})
4	6	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=5)=>(Precision={1[3],2[2],3[1]})
5	7	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=3)=>(Precision={1[1],3[2],2[4]})
6	7	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=1)=>(Precision={2[6],3[1]})
7	7	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=2)=>(Precision={1[1],3[2],2[4]})
8	8	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=6)=>(Precision={1[3],2[4],3[1]})
9	5	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=4)=>(Precision={2[2],3[2],1[1]})
10	6	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=5)=>(Precision={2[3],1[3]})
11	7	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=3)=>(Precision={1[4],2[2],3[1]})
12	7	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=1)=>(Precision={1[3],2[1],3[3]})
13	7	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=2)=>(Precision={3[2],2[4],1[1]})
14	8	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=6)=>(Precision={1[4],3[4]})
15	5	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=4)=>(Precision={3[2],2[2],1[1]})
16	6	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=5)=>(Precision={3[3],2[2],1[1]})
17	7	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=3)=>(Precision={1[2],3[3],2[2]})
18	7	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=1)=>(Precision={2[3],3[3],1[1]})
19	7	(SearchEngine=Yahoo)&(Translator=Babylon)&(DomainOfQuery=2)=>(Precision={2[2],3[2],1[3]})
20	8	(SearchEngine=Yahoo)&(Translator=Babylon)&(DomainOfQuery=6)=>(Precision={3[1],2[7]})
21	5	(SearchEngine=Yahoo)&(Translator=Babylon)&(DomainOfQuery=4)=>(Precision={2[3],1[1],3[1]})
22	6	(SearchEngine=Yahoo)&(Translator=Babylon)&(DomainOfQuery=5)=>(Precision={3[4],1[2]})
23	7	(SearchEngine=Yahoo)&(Translator=Babylon)&(DomainOfQuery=3)=>(Precision={2[4],3[2],1[1]})
24	7	(SearchEngine=Yahoo)&(Translator=Babylon)&(DomainOfQuery=1)=>(Precision={2[4],1[1],3[2]})
25	7	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=2)=>(Precision={3[3],2[4]})
26	8	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=6)=>(Precision={2[3],3[4],1[1]})
27	5	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=4)=>(Precision={2[1],1[2],3[2]})
28	6	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=5)=>(Precision={3[3],2[2],1[1]})
29	7	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=3)=>(Precision={2[2],3[4],1[1]})
30	7	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=1)=>(Precision={2[5],1[1],3[1]})
31	7	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=2)=>(Precision={2[2],1[3],3[2]})
32	8	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=6)=>(Precision={2[4],3[3],1[1]})
33	5	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=4)=>(Precision={3[2],1[2],2[1]})
34	6	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=5)=>(Precision={2[3],1[2],3[1]})
35	7	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=3)=>(Precision={3[3],1[4]})
36	7	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=1)=>(Precision={3[3],2[3],1[1]})

Fig. 6: Generated Trust Rules

Rule set: Shortened Trust Rule

(1-17)	Match	Decision rules
1	4	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=6)=>(Precision={3[4]})
2	8	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=6)=>(Precision={3[4],1[4]})
3	3	(SearchEngine=Bing)&(Translator=GoogleTranslate)&(DomainOfQuery=5)=>(Precision={3[3]})
4	4	(SearchEngine=Yahoo)&(Translator=Babylon)&(DomainOfQuery=5)=>(Precision={3[4]})
5	4	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=6)=>(Precision={3[4]})
6	3	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=5)=>(Precision={3[3]})
7	4	(SearchEngine=Google)&(Translator=Babylon)&(DomainOfQuery=3)=>(Precision={3[4]})
8	21	(DomainOfQuery=2)=>(Precision={2[21]})
9	42	(SearchEngine=Yahoo)=>(Precision={2[42]})
10	4	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=6)=>(Precision={2[4]})
11	6	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=5)=>(Precision={2[3],1[3]})
12	14	(Translator=Babylon)&(DomainOfQuery=6)=>(Precision={2[14]})
13	22	(DomainOfQuery=1)=>(Precision={2[22]})
14	3	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=5)=>(Precision={2[3]})
15	3	(SearchEngine=Yahoo)&(Translator=GoogleTranslate)&(DomainOfQuery=5)=>(Precision={1[3]})
16	4	(SearchEngine=Google)&(Translator=GoogleTranslate)&(DomainOfQuery=3)=>(Precision={1[4]})
17	4	(SearchEngine=Bing)&(Translator=Babylon)&(DomainOfQuery=3)=>(Precision={1[4]})

Fig. 7. Shortened Trust Rules (Shortening Parameter = 0.5)

In Figure 7, also consider rules #8, #9, #12 and #13 in which at least one of the attributes in the reduct set is missing. This shows that the missing attribute plays no role in deciding the value of the decision attribute. For instance, in #8, the values for "SearchEngine" and "Translator" attributes are missing which implies the fact that when the domain of query is Science (*DomainOfQuery=2*), then the precision of retrieval will be MEDIUM regardless of which search engine and the translator had been chosen. Rule #12, as another

example, also depicts the fact for "General" queries (*DomainOfQuery=6*) in the sense that when the Babylon translator is used, regardless of the search engine, the precision of the retrieval is MEDIUM.

5.4 Execution Phase

After the training phase is completed, the system is trained so it can infer, for each query, which pairs of search engines and translators will result in higher precision results. We explain the execution phase through two examples.

5.4.1 Example 1

In the first interaction, the user enters the query "تاریخچه تئوری ذرات زیر اتمی" ("the history of the theory of subatomic particles") and chooses "Science=2" as its domain (*DomainOfQuery=2*). By following Algorithm 1 and based on the rules shown in Figure 7, ROSTAM returns all pairs of Translators and Search Engines as the list of trusted ones

Algorithm 1, firstly, initializes *the Red* variable. In step 4, it assigns the set {#1, #2, #3, #4, #5, #6, #7} to HP. These numbers represent those rules of Figure 7 in which the conclusion equals 3 (*Precision = 3*). Then, by iterating through *HP*, the algorithm tries to find the rules for which the value of *DomanOfQuery* equals 2. Note that steps 2 and 3 in Algorithm 2 are not applicable here since the value of *Translator* and *SearhEngine* is not NULL in any of the rules in *HP*. Moreover, step 4 in Algorithm 2 will not add any entry to the output list due to the fact that the value of *DomanOfQuery* for all the rules in *HP* is not 2. The algorithm moves on to step 6.1 because at this point *SetOfTrusted* is empty. It assigns the set {#8, #9, #10, #11, #12, #13, #14} to variable *MP*. By looping through the rules in *MP*, the following steps are taken:

- In rule #8, since *r.Translator* and *r.SearchEngine* are null, they will represent *{Google Translate, Babylon}* $^{\oplus}$ and *{Google, Yahoo, Bing}* $^{\oplus\oplus}$ respectively. Moreover the value of *DomainofQuery* is 2, so we add to the *SetOfTrusted* all the combinations of $^{\oplus}$ and $^{\oplus\oplus}$ as follows:

$$SetOfTrusted = \{ \langle GoogleTranslate, Google\rangle, \langle Babylon, Google\rangle, \langle GoogleTranslate, Yahoo\rangle, \langle Babylon, Yahoo\rangle, \langle GoogleTranslate, Bing\rangle \langle Babylon, Bing\rangle \}$$

- In rule #9, *r.Translator* is NULL, therefore it denotes both translators. Also, the value of *r.DomainOfQuery* is NULL. Thus, the condition of the IF statement in step 4.1 in Algorithm 2 returns TRUE, and as a result the pairs <*Google Translate*, *Yahoo*> and <*Babylon*, *Yahoo*> will be added to the list.
- Rules #10, #11, #12, #13 will not add any entries to the *SetOfTrusted* because they do not match the values of input variable E.

At this point, variable *SetOfTrusted* is not empty which makes steps 7 and 8 not applicable. Finally, the *SetOfTrusted* contains the trusted pairs of agents. The query is then forwarded to the pairs in the List and the rest of workflow follows as described before.

5.4.2 Example 2

The user, at this point, enters the query " نظام پارلمانی اروپای شرقی" ("Parliamentary system in Eastern Europe"). "*Politics*" is also selected as the domain (*DomainOfQuery =3*). In step 4 of Algorithm 1, the value of *HP* is set to {#1, #2, #3, #4, #5, #6, #7}. Step 5 iterates through these rules one at a time. However, step 2 and step 3 in Algorithm 2 are not applicable here. In step 4, the algorithm adds to the *TmpList* the pair of <*Babylon, Google*> as the result of processing rule #7. Note that #7 is the only rule in *HP* for which the value of *DomainOfQuery* is 3. Steps 6, 7 and 8 will not be performed because the *SetOfTrusted* is not empty. Finally, <*Babylon, Google*>, as the only entry in *SetOfTrusted*, will be returned as the trusted pair of Translator and Search Engine.

VI. Conclusions and Future Work

In this paper, we introduced ROSTAM, a generic agent trust management framework based on the theory of Rough Sets. In order to apply ROSTAM to a specific area, one must first identify the set of attributes (features of consumers, consumers' requests and service agents) based on which the service agents' degrees of trust are to be evaluated. Subsequently, one of the attributes must be selected as the decision attribute which is used to assess the trustworthiness of service agents, i.e. the higher the value of decision attribute, the more trustworthy the agent.

For each user request, ROSTAM collects the trust evidences, i.e. the values of trust attributes, from different sources in the system. According to the current set of trust rules, ROSTAM extracts the list of the most trusted service agents. The request is then submitted to the trusted agents and the results are presented to the user. At this point, the user rates the interaction with each trusted agent. ROSTAM stores these rating values along with other trust evidences. Finally, ROSTAM generates a new set of trust rules according to the trust evidences by employing the theory of Rough Sets. These rules are then used to determine the set of most trusted agents for the consecutive requests.

In this paper, we applied ROSTAM to the domain of cross language Web search. The application consists of implementing the APIs for three search engines (Yahoo, Google, and Bing) as well as two machine translation services (Google Translate, and Babylon). The ultimate

goal here was to increase the precision of the Web search process as much as possible. Consequently, we chose the precision of retrieval as our decision attribute. To define the condition attributes, we conducted a survey asking people which Web service providers they choose as their search engine, news source, and sports news as well as shopping, weather and email. We also asked them the reasons why they prefer such services. By examining their responses, we defined five condition attributes: *Domain Of Query*, *Is Searchable By Date*, *Update Frequency*, *Translator*, and *Search Engine*.

Furthermore, we conducted an experiment in two phases. In the first phase, the system was trained using a set of 40 queries that seek information in different domains. ROSTAM collected the values of trust attributes for all possible combinations of the translators and search engines, and generated the initial trust rules. The second phase was the execution phase in which the users were able to submit their queries. By utilizing the Rule Matcher component, ROSTAM successfully returned those pairs of translator and search engines that would result in higher precision of retrieval according to the trust rules.

This work can be extended by employing the extensions of Rough Sets, like Fuzzy Rough Sets [49] and Variable Precision Rough Sets [50] in order to generate probabilistic trust rules. As used in this research, the basic theory of Rough Sets results in Boolean rules. However, a better way of stating these rules would be the case in which the rules are expressed by a probability of correctness. In our case study, for each request, we generate a new set of trust rules at the end of the workflow. One can extend this research by gradually updating the set of trust rules at the end of the workflow. Also, in our case study, we ask the user to choose the domain of his query. Automatically identifying such information can form a further extension for this research. Agent trust management has been applied widely to E-Commerce systems. The goal of these systems is to employ a number of intelligent agents who are able to negotiate and trade on behalf of the users. The next step of this research may be replacing the trust model of an existing e-commerce system with ROSTAM and perform a comparison between the two systems.

References

[1] A. Gutscher, A trust model for an open, decentralized reputation system, Trust Management, (2007) 285-300.

[2] S. Abedinzadeh, S. Sadaoui, Trust Management based on Human Plausible Reasoning: Application to Web Search, in: The 4th ASE/IEEE International Conference on Information Privacy, Security, Risk and Trust IEEE, Amsterdam, Netherlands, 2012.

[3] Firdawsi, D. Davis, Shahnameh : the Persian book of kings, Viking, New York, 2006.

[4] S. Abedinzadeh, S. Sadaoui, A Rough Set Approach to Agent Trust Management in: The Second IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT2010), IEEE, Minneapolis, Minnesota, US, 2010.

[5] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning about Data Kluwer, Boston, 1991.

[6] S.-M. Chen, J.-M. Tan, Handling multicriteria fuzzy decision-making problems based on vague set theory, Fuzzy Sets and Systems, 67 (1994) 163-172.

[7] Y. Yao, Y. Zhao, Attribute reduction in decision-theoretic rough set models, Information Sciences, 178 (2008) 3356-3373.

[8] W. Weijie, R. Gang, W. Wei, The Applications of Rough Set Theory in Civil Engineering, in: Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on, 2010, pp. 27-31.

[9] K.A. Cyran, A. Mrózek, Rough sets in hybrid methods for pattern recognition, International Journal of Intelligent Systems, 16 (2001) 149-168.

[10] L.-P. Khoo, L.-Y. Zhai, A prototype genetic algorithm-enhanced rough set-based rule induction system, Computers in Industry, 46 (2001) 95-106.

[11] B. Chang, H.-M. Pei, J.-R. Chang, Using the Rough Set Theory to Investigate the Building Facilities for the Performing Arts from the Performer's Perspectives Intelligent Decision Technologies, in: J. Watada, G. Phillips-Wren, L.C. Jain, R.J. Howlett (Eds.), Springer Berlin Heidelberg, 2011, pp. 647-657.

[12] L.-Y. Zhai, L.-P. Khoo, Z.-W. Zhong, A rough set based decision support approach to improving consumer affective satisfaction in product design, International Journal of Industrial Ergonomics, 39 (2009) 295-302.

[13] W. Al-Mayyan, H.S. Own, H. Zedan, Rough set approach to online signature identification, Digital Signal Processing, 21 (2011) 477-485.

[14] P. Srinivasan, M.E. Ruiz, D.H. Kraft, J. Chen, Vocabulary mining for information retrieval: rough sets and fuzzy sets, Information Processing & Management, 37 (2001) 15-38.

[15] P. Lingras, G. Peters, Applying rough set concepts to clustering, Rough Sets: Selected Methods and Applications in Management and Engineering, (2012) 23-37.

[16] P. Yin, Z.-j. Wang, H.-x. Li, Corporate failure prediction of Chinese listed companies: A variable precision rough set theory, in: International

Conference on Management Science and Engineering, 2009. ICMSE 2009. , 2009, pp. 1290-1296.

[17] E.M. Maximilien, M.P. Singh, A framework and ontology for dynamic web services selection, Internet Computing, IEEE, 8 (2004) 84-93.

[18] C.H. Yew, H. Lutfiyya, A middleware-based approach to supporting trust-based service selection, in: IFIP/IEEE International Symposium on Integrated Network Management (IM), , IEEE, London, ON, Canada, 2011, pp. 407-414.

[19] P. Wang, K.M. Chao, C.C. Lo, R. Farmer, An evidence-based scheme for web service selection, Information Technology and Management, 12 (2011) 161-172.

[20] P. Wang, QoS-aware web services selection with intuitionistic fuzzy set under consumer's vague perception, Expert Systems with Applications, 36 (2009) 4460-4466.

[21] S. Nusrat, J. Vassileva, Recommending services in a trust-based decentralized user modeling system, Advances in User Modeling, (2012) 230-242.

[22] S. Wang, Q. Sun, H. Zou, F. Yang, Reputation measure approach of web service for service selection, Software, IET, 5 (2011) 466-473.

[23] F. Bellifemine, A. Poggi, G. Rimassa, JADE: a FIPA2000 compliant agent development environment, in: Proceedings of the fifth international conference on Autonomous agents, ACM, Montreal, Quebec, Canada, 2001, pp. 216-217.

[24] L. Tong, L. An, Incremental learning of decision rules based on rough set theory, in: Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on, 2002, pp. 420-425 vol.421.

[25] N. Shan, W. Ziarko, Data-based Acquisition and Incremental Modification of Classification Rules, Computational Intelligence, 11 (1995) 357-370.

[26] G. Sen, W. Zhi-Yan, W. Zhi-Cheng, Y. He-Ping, A novel dynamic incremental rules extraction algorithm based on rough set theory, in: Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, 2005, pp. 1902-1907 Vol. 1903.

[27] H. Chen, T. Li, S. Qiao, D. Ruan, A rough set based dynamic maintenance approach for approximations in coarsening and refining attribute values, International Journal of Intelligent Systems, 25 (2010) 1005-1026.

[28] C. Hongmei, L. Tianrui, H. Chengxiang, J. Xiaolan, An incremental updating principle for computing approximations in information systems while the object set varies with time, in: Granular Computing, 2009, GRC '09. IEEE International Conference on, 2009, pp. 49-52.

[29] G.B. Jan, S.S. Marcin, RSES and RSESlib - A Collection of Tools for Rough Set Computations, in: Revised Papers from the Second International Conference on Rough Sets and Current Trends in Computing, Springer-Verlag, 2001.

[30] K. Er Øhrn Jan, ROSETTA -- A Rough Set Toolkit for Analysis of Data, in: P. Wang (Ed.) Proceedings of the Third Joint Annual Conference on Information Sciences, Durham, NC, 1997, pp. 403-407.

[31] W. Guo-Yin, Z. Zheng, Z. Yi, RIDAS - a rough set based intelligent data analysis system, in: Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on, 2002, pp. 646-649 vol.642.

[32] A.T. Bjorvand, "Rough enough"-a system supporting the rough sets approach, in: Proceedings of the sixth Scandinavian conference on Artificial intelligence, IOS Press, Helsinki, Finland, 1997, pp. 290-291.

[33] I. Düntsch, G. Gediga, The rough set engine GROBIAN, in: The proceedings of the 15th IMACS World Congress, Berlin, Germany, 1997, pp. 613-618.

[34] B. Prędki, S. Wilk, Rough set based data exploration using ROSE system Foundations of Intelligent Systems, in: Z. Ras, A. Skowron (Eds.), Springer Berlin / Heidelberg, 1999, pp. 172-180.

[35] D. Wu, D. He, Exploring the Further Integration of Machine Translation in English-Chinese Cross Language Information Access, Program: electronic library and information systems, 46 (2012) 3-3.

[36] L. Shaozi, Z. Changle, C. Huowang, Web document retrieval based on multi-agent, in: Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design, 2005., 2005, pp. 469-474 Vol. 461.

[37] E. Greengrass, Information Retrieval: A Survey, in, DOD Technical Report: TR-R52-008-001, , 2001.

[38] W. Dayong, Z. Yu, Z. Shiqi, L. Ting, Identification of Web Query Intent Based on Query Text and Web Knowledge, in: Pervasive Computing Signal Processing and Applications (PCSPA), 2010 First International Conference on, 2010, pp. 128-131.

[39] M.-Y. Chen, H.-C. Chu, Y.-M. Chen, Developing a semantic-enable information retrieval mechanism, Expert Systems with Applications, 37 (2010) 322-340.

[40] A. De, S.K. Kopparapu, A rule-based Short Query Intent Identification System, in: Signal and Image Processing (ICSIP), 2010 International Conference on, 2010, pp. 212-216.

[41] E. German, L. Sheremetov, An agent framework for processing FIPA-ACL messages based on interaction models, in: Proceedings of the 8th international conference on Agent-oriented software engineering VIII, Springer-Verlag, Honolulu, HI, USA, 2008, pp. 88-102.

[42] M. Winikoff, Jack™ Intelligent Agents: An Industrial Strength Platform Multi-Agent Programming, in: R. Bordini, M. Dastani, J. Dix, A. Fallah Seghrouchni (Eds.), Springer US, 2005, pp. 175-193.

[43] E. Makosa, Rule tuning, Uppsala University, Sweden, (2005) 1-51.

[44] M. Sikora, Decision Rule-Based Data Models Using TRS and NetTRS – Methods and Algorithms Transactions on Rough Sets XI, in: J. Peters, A. Skowron (Eds.), Springer Berlin / Heidelberg, 2010, pp. 130-160.

[45] E. Tsang, Z. Suyun, Decision Table Reduction in KDD: Fuzzy Rough Based Approach Transactions on Rough Sets XI, in: J. Peters, A. Skowron (Eds.), Springer Berlin / Heidelberg, 2010, pp. 177-188.

[46] J. Bazan, M. Szczuka, The Rough Set Exploration System Transactions on Rough Sets III, in: J. Peters, A. Skowron (Eds.), Springer Berlin / Heidelberg, 2005, pp. 25-42.

[47] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal, A collaborative filtering approach to mitigate the new user cold start problem, Knowledge-Based Systems, (2011).

[48] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, Tampere, Finland, 2002, pp. 253-260.

[49] D. Tingquan, C. Yanmei, X. Wenli, D. Qionghai, A novel approach to fuzzy rough sets based on a fuzzy covering, Inf. Sci., 177 (2007) 2308-2326.

[50] W. Ziarko, Variable precision rough set model, Journal of computer and system sciences, 46 (1993) 39-59.

Sadra Abedinzadeh was born in Tehran, Iran. He received his B.Sc. in computer science and his M.Sc. in software engineering from University of Tehran, Tehran, Iran in 2004, and 2007 respectively.

He started his Ph.D. in computer science at department of Computer Science, University of Regina, Canada in 2008. His main research area is trust management in multi agent systems. His other research interests include human plausible reasoning, theory of rough sets, information retrieval, software architecture, and database systems.

Dr. Samira Sadaoui obtained her MSc and PhD degrees in Computer Science from the University of Nancy I in France. She is currently Professor of Computer Science at the University of Regina in Canada.

Her research interests are in the area of Software Engineering and include Formal Methods; Multi-Agent Systems; Multi-Attribute and Reverse Auctions; Trust Management; Constraint Programming and Boolean Satisfiability. Her research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) federal grant.

How to cite this paper: Sadra Abedinzadeh, Samira Sadaoui,"A Rough Sets-based Agent Trust Management Framework", International Journal of Intelligent Systems and Applications(IJISA), vol.5, no.4, pp.1-19, 2013.DOI: 10.5815/ijisa.2013.04.01