A Trust Model for Multiagent Recommendations

Fabiana Lorenzi[1,2], Gabriel Baldo[2], Rafael Costa[2], Mara Abel[1], Ana Bazzan[1], Francesco Ricci[3]
[1]Instituto de Informática, UFRGS
Caixa Postal 15064 91.501-970 Porto Alegre, RS, Brasil
[2]Universidade Luterana do Brasil
Av. Farroupilha, 8001 Canoas, RS, Brasil
[3]Free University of Bozen-Bolzano
Bozen-Bolzano, Italy
lorenzi, mara, bazzan@inf.ufrgs.br
baldo12@gmail.com, rafabpc@gmail.com
fricci@unibz.it

***Abstract*— This paper describes a trust model for multiagent recommender systems. A user's request for a travel recommendation is decomposed by the system into subtasks, corresponding to travel services. Agents select tasks autonomously, and accomplish them using knowledge derived from previous solutions or with the help of other agents. Agents maintain local knowledge bases and, when requested to support a user in a travel planning task, they may collaborate exchanging information stored in their local bases. During this exchange process trusting other agents is fundamental. It helps agents to improve the quality of the recommendations and to avoid communication with unreliable agents. In the proposed model, the trust is also used to allow agents to become experts in particular subtasks, helping them to generate better recommendations. In this paper, we propose and validate a multiagent trust model showing the benefits of such model in a travel planning scenario.**

***Index Terms*— Multiagent recommender system, Trust model**

I. Introduction

The expansion of the types and number of information services offered nowadays in Internet is making more and more urgent the need of tools that can help the users to select the information or items that they need and to avoid the information overload problem.

Recommender systems have been developed to deal with these problems; they are able to aggregate information and recommendations coming from a community of users and can match these recommendations with the information needs of the users, delivering to them only relevant and personalized suggestions [1]. These systems are being applied to several domains [2] such as books, CDs, movies, travels and may more.

In domains where the knowledge is distributed, agents techniques are being used for getting recommendations from different information sources [3]. Agents can operate in a cooperative way, retrieving, filtering and using information relevant to the recommendations. Agents may autonomously perform different parts of the recommendation and the assembling of all parts represents the final recommendation. However, when agents become fully autonomous they become forced to make decisions about when to engage with other agents [4].

In this paper we propose a novel approach where agents have their own knowledge bases and solve their assigned tasks searching for the information in their bases. Agents become experts in specific parts of the recommendation and they are able to provide better recommendations through this specialization.

However, when an agent does not find locally the necessary information to compose the recommendation, it may decide to communicate with other agents and exchange information needed to serve the user with the requested recommendation. In fact, this communication process may become a weakness of the multiagent approach because it can generate a large number of messages between agents, hence deteriorating response time. To deal with this problem we propose a trust model forcing agents to exchange information with a subset of the agents, the trusted ones.

We introduce a trust model that has two important functions: 1) it helps an agent to decide which agent to trust; and 2) it helps an agent to become expert in more specific tasks of a recommendation process. The proposed model was validated in a multiagent travel recommender system.

This paper is organized as follows: the next section discuss the related work on multiagent recommender systems and trust. Section III presents the trust model and Section IV presents the experiments we conducted in the tourism domain. Finally section V summarizes the contributions of the paper.

II. Related Work

A. Multiagent Recommender Systems

Multiagent techniques can be applied to retrieve, filter and use information relevant to recommendations. These systems have been developed in different domains such as legal [5], marketplace [3], e-commerce [6], and tourism [7], [8].

In [5], for instance, the authors present a multiagent recommender system for the legal domain. The system

doi:10.4304/jetwi.2.4.310-318

classifies legal normative instruments into legal branches. Each user specifies his/her interests for certain legal branches and receives recommendations of instruments they might be interested in.

More in general, in multiagent recommender systems a collection of interacting agents manage the recommendation generation process, trying to improve the recommendation quality that can be obtained by a single agent. Agents cooperate and negotiate in order to complement their partial solutions and satisfy the users preferences.

MAPWEB [9] is an example of a multiagent recommender that plans travels suited for the user's preferences. The following agents were developed: the UserAgent, which is responsible for the communication between user and system, receives the user request and presents the final result to him; the PlannerAgent, that is responsible for planning the travel; the Webbot, responsible for searching information in Internet; and the CoachAgent, which acts like a coach to the group of agents, controlling them and giving tasks to them.

This system has the peculiarity that agents store the plans performed previously, they use domain knowledge to build new plans, and a communication protocol among agents allows them to help each other in completing the task. However, the communication is not a natural process among the agents. The CoachAgent agent controls the possible communication and indicates which agent should help an agent. The drawback is that the control and management of the tasks are centralized in the CoachAgent and there is no established process to validate the knowledge of the agents. Moreover, agents may operate with outdated information during the planning process, hence generating bad recommendations.

SmartClient is another agent-based system which helps the user to plan a flight route, and models the space of the possible solutions by a constraint satisfaction problem (CSP) [10]. User enters some constraints such as departure city, the cities which he wants to visit, and the travel dates.

The server accesses databases to generate the corresponding CSP considering the constraints informed by the user. The CSP is packaged with a search algorithm to compose an agent (called the SmartClient). Then, this agent is responsable for interacting with the user, showing all possible routes.

However, the system collects information from the server only once to avoid repeated accesses, and this limits the effectiveness of the solution as it is not possible to modify the user preferences and to refine the query.

In [3], the authors investigated the feasibility of building a recommender system as a marketplace in which the various recommender agents compete to get their recommendations displayed to users. The market works as a coordinator of the multiple recommendation techniques in the system. The main idea of the system is to improve the recommendation quality choosing the best recommendation method to the current situation. The market correlates the agents' internal evaluation and the user's evaluation of the recommendations by invoking a bidding process and a rewarding regime. Auctions are used to decide the recommendation winners and the agents who provided good recommendations (chosen by the user) receive some reward in return. Agents that receive more reward become richer and are able to get their recommendations advertised more frequently. This behavior has a drawback because it avoids the serendipity which means that potentially the system will never surprise the user with unexpected recommendations.

B. Trust Models

In multiagent systems, trust is defined as a measurable level of the probability which an agent knows that another agent will perform a particular action [11].

In e-commerce applications, for example, a trust model is used during the decision-making process when an agent must select a product, which it has not experienced, based on the recommendation of another agent [12] [13].

In recommender systems, trust networks contribute to effectiveness of the recommendation process by allowing users to form a better opinion about the items exploiting the judgment of trusted sources or agents that have evaluated those items [14].

As presented in [15], several research communities gave attention to trust, and this resulted in many different views about how to measure and use trust.

An example of multiagent recommender system that deals with trust is presented in [16]. In this trust model, agents consider other agents as personal entities, which may be more or less reliable, and their trust values can be computed on the basis of a conversational exchange in which one agent solicits the opinions of the other on a set of items.

Another trust model is presented in [17] and the goal is to provide recommendations based on trust estimation rather on rating prediction. It is built from trust data provided by users in the popular website epinions.com (that allows users to review various items such as books, cars and music). The trust data can be extracted and used as part of the recommendation process, softening the sparsity and cold start problems presented in collaborative filtering algorithms, where a large number of ratings from similar users are required to build reliable recommendations for a target user.

One challenge for trust models is the context of the agents' interactions [11]. In fact, the majority of the existing models does not take into account that interactions between agents take place within a environmental context. For instance, if an agent performed poorly because of some unforeseen changes in the environment, it should not be considered unreliable. Conversely, there should be the possibility to take into account the environmental variables in deciding to trust another agent.

All the approaches so far assume a centralized trust matrix modelling the trust relationships between all the pairs of agents. Our approach differs from these approaches due the fact that each agent has a local list of trust degrees on

other agents and they are not able to see the trust degrees of other agents. Moreover, in our approach agents are fully autonomous and they use trust degrees also to decide which task they should perform in the recommendation process.

III. The Trust Model

In real life, if a person wants to buy a product and wants to limit the risk to make a wrong decision, could ask a friend for an opinion about the product. This same behavior is found in Multiagent Recommender Systems, where agents work in a cooperative way to compose a recommendation and they may ask other agents for the information necessary to complete the recommendation process.

We have designed and implemented a trust model for MATRES, a multiagent recommender system that we applied in a travel planning scenario [18]. MATRES has two important characteristics: trustworthiness of the agents and the agents specialization. The trustworthiness of the agents represents the trust degree that each agent has in other agents in order to know who agent is reliable. The specialization helps agents to become experts in performing specific tasks which allows them to improve the results of the recommendations provided.

A. Multiagent Recommender Scenario

In MATRES a recommendation cycle starts when the user defines a query, specifying some preferences about the travel [18].

As shown in figure 1, in the main page the user chooses the travel preferences. Here the user's query originates different sub-tasks for each needed travel services: flight, accommodation and attractions in the city the passenger wants to visit.

For instance, if the user states that s/he wants to travel to Rome in June and stay there for 10 days in a 4-star hotel, then these preferences generate a set of sub-tasks: $t_1 =$ *flight* from the original city to Rome and back 10 days later; $t_2 = 4$-star *hotel* in Rome; and $t_3 =$ *Attractions* to visit in Rome during the traveller stay. These sub-tasks t_1, t_2, t_3 are then performed by the agents in the community.

Each task is described by a set of attributes representing the features of the searched travel service. As we can see in figure 1, t_1 has 4 attributes: *class of flight*, *type of flight*, *stops* and *connections*, t_2 has 5 attributes: *category of the hotel*, *type of the room*, *pool*, *wi-fi* and *pets*, and t_3 has 7 attributes that represent types of attractions that the user may choose to visit.

Agents are distributed in a community $C = \{a_1, a_2, ..., a_n\}$, where each element of this set has the following characteristics:

- **Knowledge**: an agent has its knowledge base where it stores all the performed recommendations. The knowledge is stored in a case base, and each case includes the description of the user's query and the recommendation generated by the agent for that query.
- **Cooperation**: each agent performs a task, and the results of the tasks related to a single query are assembled in the final recommendation. Agents co-operate in order to achieve the best recommendation for the user. By cooperation we mean that agents may exchange information when necessary.
- **Trust**: agents have a trust mechanism letting them to increase or decrease the trust in other agents according to their evaluations, which are provided by the users.
- **Specialization**: an agent may become an expert in a specific type of task during its activity by solving more tasks of that type.

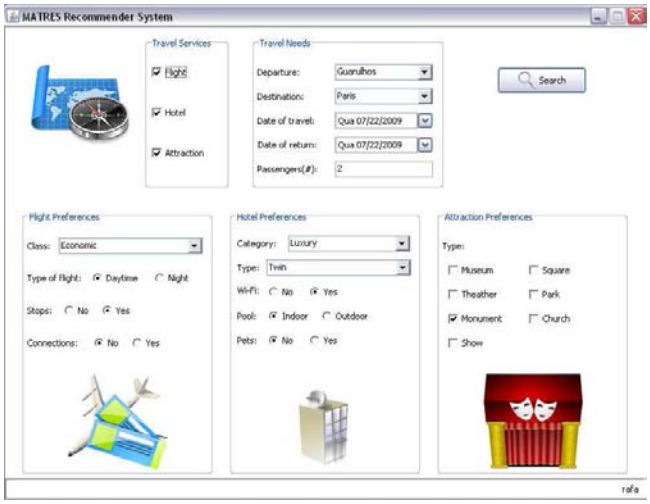

Figure 1. Main page for entering the user's query

1) Solving Tasks: In MATRES each sub-task can be handle by a different agent. Every time an agent has to perform a task, i.e., to compose a part of the request recommendation, it can initiate a search process in its knowledge base for the information necessary to generate the recommendation.

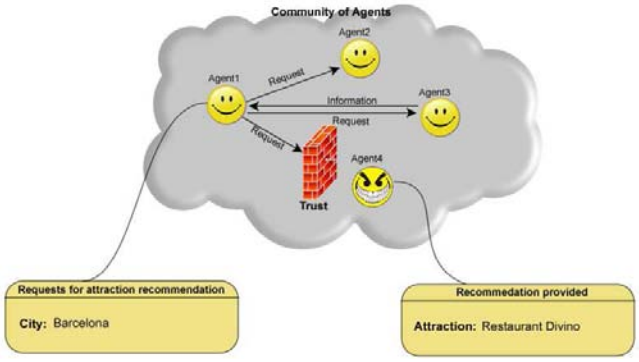

Figure 2. The process of exchanging information among agents

If the agent does not find any useful solution in its knowledge base, it may communicate with other agents, asking for the required information. Figure 2 shows an

example of agents exchanging information.

The motivation of this process is that multiple agents may generate better recommendations if they cooperate when one of them has problems to generate a recommendation.

Algorithm 1 Solving tasks

1. {C is the community of agents}
2. {$\mathcal{L}$ is the set of tasks to be performed}
3. {ι is the task to be performed}
   Function Solve_tasks (a_i,$\mathcal{L}$,C)
4. Chooses a task ι of type of task t_k that has the highest confidence degree
5. Changes the state of the task ι
6. $\mathcal{S} \leftarrow$ LocalKBSearch(ι)
7. **if** $\mathcal{S} = \emptyset$ **then**
8. **for** $a_j \in C$ with the highest *trust* **do**
9. $\mathcal{S} \leftarrow$ CommunitySearch(a_j, ι)
10. **if** $\mathcal{S} = \emptyset$ **then**
11. $\mathcal{S} \leftarrow$ SearchWeb(ι)
12. **end if**
13. **end for**
14. **end if**
15. **return** ($\mathcal{S}$)

The algorithm 1 presents the main procedures of an agent to solve a task in the recommendation process. Given a list of task L, the agent picks a task ι to solve (line 4), according to its confidence degree in each type of task (that will be explained in section III-B.2).

After choosing task ι, the agent changes the status of the task (line 5), avoiding that other agent tries to solve the same task.

In order to performed task ι, the agent first searches for the information in its knowledge base (line 6). If the agent does not have the information in its knowledge base, it starts the CommunitySearch procedure (line 9), where the agent communicates with other agents to obtain the necessary information to complete the part of the recommendation.

In the CommunitySearch process, the agent communicates with trusted agents asking for information necessary to perform task ι. For instance, $agent_1$ is performing a task about flights from Paris to Rome, but it does not have this information in its knowledge base. Thus, it sends a request for trusted agents asking for flights from Paris to Rome. It accepts the first answer received and with this information it continues performing task ι, generating the recommendation that will be presented to the user. It stores the shared information in its knowledge base.

If no agent in the community has the information then the agent starts the Search Web procedure (line 11) which means that it searches for the information into the Web.

After all agents involved in the user's request performing the assigned tasks, the results are presented to the user. Figure 3 shows how the recommendation is presented to the user.

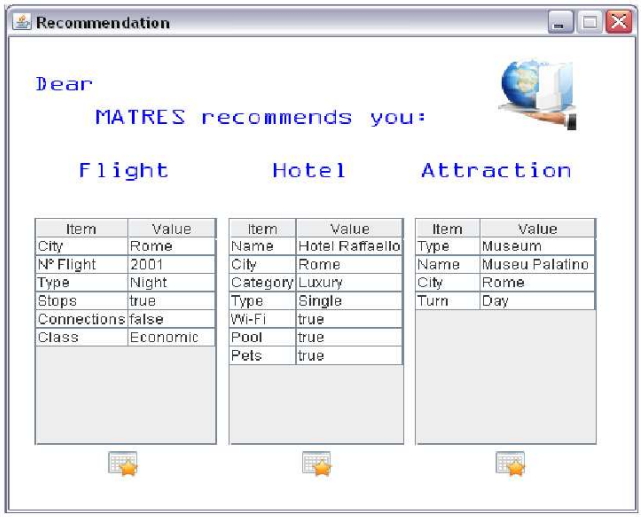

Figure 3. An example of recommendation presented to the user

When agents share information during the recommendation process, the trust between them is very important. For instance, an agent may try to maliciously influence the system towards a solution that is not the best for the user, or it may try to provide information that is not updated. These situations must be avoided by using an appropriate protocol that leveraging the log of past agent activity can establish a degree of trust on the agent.

For example, in figure 2, $agent_4$ represents a maliciously actor that keeps recommending the same attraction in all the requests. Using a trust model another agent can detect the potential risk of relying on that recommendation, it can better choose with whom to communicate, hence avoiding the interaction with this kind of unreliable agents.

Thus, we propose a trust model where cooperating agents use the estimated trust degree in other agents in order to improve their ability to deliver accurate recommendations.

B. Trust Model

In the proposed trust model, an agent computes the *trust degree* in other agents. The trust degree measures how effective were the previous cooperations between agents.

Agents wait for the user evaluation in order to update their trust degrees. This evaluation consists of the ratings given by the user to the presented results. Actually, the user rates each single attribute value in the received solution with an evaluation: "I liked it" (positive) or "I did not like it" (negative).

Figure 4 shows examples of flight, hotel and attraction evaluations. For each travel service, the user rates the attributes and then saves the evaluations.

When the agent a_i receives an evaluation from the user concerning a task of type t_k ($i = 1, \ldots, n$) two possible situations may occur:

1) a_i was not helped to perform the task;

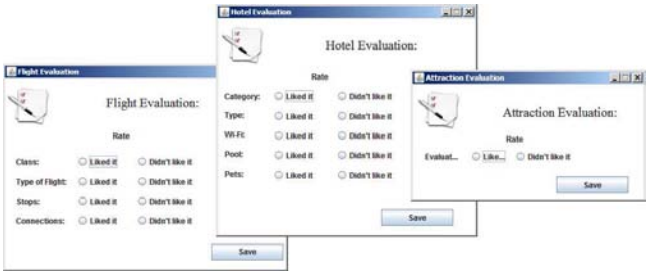

Figure 4. Examples of evaluations screens

2) a_i received help from another agent a_j when performed the task.

In the next section we will explain how these two situations are managed.

1) Trust Degree: The *trust degree* helps the agent when choosing another agent to exchange information necessary to the accomplishment of a task.

Agents are autonomous and they are capable of performing their tasks. However, when they do not have information in their knowledge bases, they may share information with agents from the community in order to get to be able to provide the recommendation.

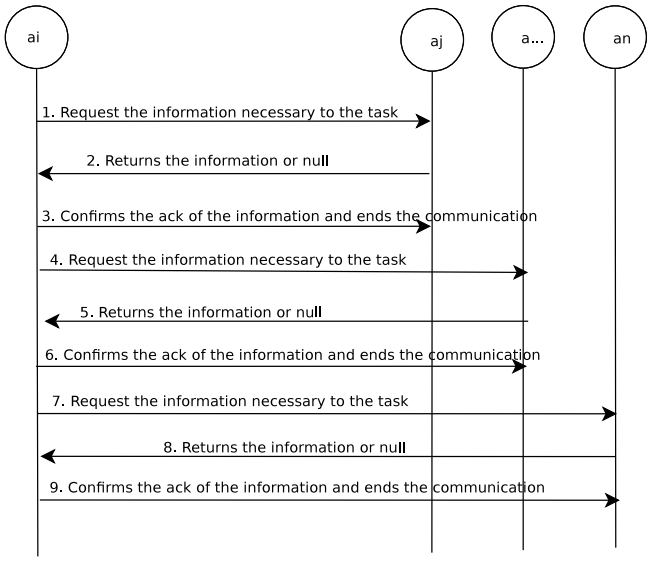

Figure 5. Communication between agent a_i the agents it trusts

Figure 5 shows the flow of communication between agents. In this example, a_i (who is becoming an expert in hotel task) has to provide a recommendation of a hotel in Rome. However, it only has cases about hotels in Paris, Rio de Janeiro, and Lisbon. Thus, it does not have information to provide a recommendation for the user. In this case, a_i has the option of communicating with other agents in the community in order to find the information.

Usually, agent would broadcast its request for all agents in the community. However, this could jeopardize the response time of the system. For this reason, a_i will communicate only with agents it trusts (represented in the figure by $\{ a_j, \ldots, a_n \}$.

The trust degree is used by the agents to avoid unnecessary communication, improving the exchange information process. Thus, the recommendation process becomes efficient because agents accept to share information just with trusted agents.

The proposed trust model is based on the observation that the trust between agents depends on the context. This means that, for instance, a_1 can trust a_2 when is requested to perform tasks of type t_1 but does not trust a_2 for tasks of type t_2.

$$T^{t_k}_{a_i,a_j} = \frac{\sum_{\theta=1}^{\eta} v_\theta^{(t_k)} \times e^{-\tau\theta}}{\sum_{\theta=1}^{\eta} e^{-\tau\theta}} \quad (1)$$

The trust degree of agent a_i for agent a_j with respect of tasks of type t_k is given by $T^{t_k}_{a_i,a_j}$. It is updated through equation 1, where:

- $v_\theta^{(t_k)}$ is the average of the positive evaluations of the user for the task solved at time θ. The user evaluates each attribute and the average of the positive evaluations generates $v_\theta^{(t_k)}$ that ranges between 0 and 1.
- τ is a constant that weights the value $v_\theta^{(t_k)}$ when updating the trust degree;
- η is the number of days past since the evaluation.

$T^{t_k}_{a_i,a_j}$ returns a value in $[0, 1]$, where 0 represents the minimum trust degree of a_i in a_j and 1 represents the maximum trust degree of a_i in a_j.

τ must be defined according to the number of days (θ) that the evaluation is available and it should be a value that tends to 0 in the equation $e^{-\tau\cdot\theta}$.

For example, $\theta = 50$ represents the fact that the evaluation should last for 50 days, and we need a τ value in order to achieve nearly $e^{-\tau\cdot\theta} = 0$. Thus, with $\theta = 50$ we tested the value that is closest to 0 and we use $\tau = 0.088$.

Each agent has a XML file that stores the trust degree in other agents and the confidence degree for each travel service. Figure 6 shows an example of an XML flight service trust file of a_2. We can see that regarding the flight service, the trust degree of a_2 in a_1 is 0.9375, the trust degree in a_3 is 0.82, and the trust degree of a_2 in a_{10} is 0.5. The agent stores also information about the number of positive evaluations (*intpos*). This information are used in the experiments to validate the trust model.

2) Confidence Degree: Agents may accomplish a task of type t_k searching for information in their own knowledge bases. When this happens, they do not communicate with other agents. In this case the evaluation received from the user, a_i may increase or decrease its *confidence degree* for type of task t_k.

If the evaluation of the user was positive, then the confidence degree will be increased which means that the agent is becoming more expert in the type of task t_k. On the other hand, if the evaluation was negative, the confidence degree will be decreased.

The task evaluation is then used in the agent confidence degree computation: the agent increases the confidence in a task type when it solves correctly these tasks. Equation 2

Figure 6. XML file of trust in flight service of agent a_2

```
<trustFlight>
 <trustAg1>
   <trust>0.9375</trust>
   <intPos>4</intPos>
 </trustAg1>
 <trustAg3>
   <trust>0.82</trust>
   <intPos>3</intPos>
 </trustAg3>
 ...
 <trustAg10>
   <trust>0.5</trust>
   <intPos>1</intPos>
 </trustAg10>
</trustFlight>
```

shows how the confidence degree is updated in each agent. It is the same updating used in the trust degree, however here is the trust of a_i in itself.

$$T^{t_k}_{a_i,a_i} = \frac{\sum_{\theta=1}^{\eta} v_{\theta}^{(t_k)} \times e^{-\tau\theta}}{\sum_{\theta=1}^{\eta} e^{-\tau\theta}} \quad (2)$$

Table I shows an example of the confidence degree of a_3 in task of flight. We can see that after the first performed task (that had positive evaluation from the user) the confidence degree of a_3 in task of flight increased. However, the second performed task had a negative evaluation and then the confidence degree decreased to 0.5. From the third to the eighth performed task, the confidence degree of a_3 varied according to the evaluations received. However, we can noticed that from the fifth performed task the confidence degree started to grow due the positive evaluations received by the agents.

TABLE I.
EXAMPLE OF CONFIDENCE DEGREE EVOLUTION IN a_3 FOR TASK OF FLIGHT THROUGH 8 PERFORMED TASKS

Tasks	Evaluation	Trust
1	1	1
2	0	0.5
3	1	0.65
4	0	0.49
5	1	0.58
6	1	0.64
7	1	0.67
8	1	0.70

During the recommendation cycles, the more cases an agent solves, the bigger its knowledge base become. It means that an agent could become an expert in a type of task t_k and could improve its recommendations for tasks of type t_k.

The confidence degree is also used by the agent to choose the task to perform. For example, if an agent has always solved flights tasks, it would likely have better results in performing another flight task rather than finding a good hotel service.

IV. EXPERIMENTS AND SIMULATIONS

Some experiments were performed in order to validate the proposed trust model. A community of 10 agents was created. A knowledge acquisition step was done in a real travel agency and 40 cases (which represent 120 solved tasks, i.e., three tasks for each case) were acquired and stored in the agents knowledge bases as follows:

- Each agent received 12 solved tasks;
- *Agent*$_5$ and *Agent*$_3$ received more cases of flight tasks;
- *Agent*$_2$ received more cases of hotel tasks;
- The rest of cases was distributed randomly in the other agents;
- *Agent*$_7$ received 10 fake cases, with not existent flights and attractions.

The travel agency also provided 20 completed cases (queries and users evaluations) to be run as new queries in the simulations. These 20 cases represents 60 tasks because each case was composed by flight, hotel and attractions tasks.

Two different trust strategies were run in simulations:

1) **WTrust**: Agents considered their trust degrees during the recommendation process;
2) **NTrust**: Agents did not consider their trust degrees to recommend.

The 60 tasks were run in both simulations. To facilitate the understanding of the results, they were grouped into 6 ranges of 10 tasks each (the sequence of the performed tasks was preserved).

While the simulations were being performed, a log file was created to record the trust degrees in each recommendation cycle. The values were used to observe the behavior of the agents. As the simulations were run offline, i.e., without users, each recommendation provided by the agents were evaluate by another human expert.

We measured the number of positive evaluations obtained by the agents after solving the tasks. These values were compared with the evaluations of the recommendations done by the human travel agent.

A. Simulation 1: strategy NTrust

In the first simulation agents were not able to use their trust degrees. This means that they choose randomly the tasks to perform and the agents they exchange information with.

TABLE II.
TASK EVALUATION - AGENTS DO NOT USE THE TRUST MODEL - NTRUST

Ranges	Number of positive results (rate="Liked it")			
	Flight	Hotel	Attraction	Total
1 - 10	28	30	9	67
11 - 20	36	26	8	70
21 - 30	24	34	9	67
31 - 40	24	18	10	62
41 - 50	36	18	9	63
51 - 60	28	26	10	64

Table II shows the results of this experiment. Columns 2, 3 and 4 show the total number of positive evaluations for flight, hotel and attractions tasks. As we mentioned in section III-A, each travel agent is composed by a set of attributes (flight=4, hotel=5 and attraction=1). Thus, the number of positive evaluations in each travel service corresponds to the number of attributes that received the rate "I liked it" from the user. The last column presents the sum of all types of tasks.

We can see in table II that the positive results increased a little bit during the second set of queries. However, from the third set of queries the results start to decrease and in the last queries the total number was not good as it could be.

These results were expected for two reasons: 1) agents did not update their confidence degrees and they did not use the confidence degrees as they solve the tasks. If an agent chooses tasks at random then the number of cases of a task type will grow larger that others and, consequently, they do not become experts in this type of task, and 2) agents did not use the trust degrees to select other agents to exchange information when needed which means that the number of communication among agents increases.

Analyzing the logs with all the solved tasks, we saw that:

- *Agent*$_5$, *Agent*$_3$ and *Agent*$_2$ chose tasks that they were not experts, generating a high number of communication between them and other agents.
- *Agent*$_7$, who has fake cases, appeared in several communications among agents. This happened because agents did not use trust degrees, i.e., they did not update the trust degree in in Agent$_7$ after receiving negative evaluations. This means that Agent$_7$ was considered reliable during the whole recommendation process.
- The number of communication among agents increased considerably. As mentioned above, when agents do not use trust degrees they try to get the information for their tasks with all agents from the community.

B. Simulation 2: strategy WTrust

In this experiment, agents use their trust degrees. Before running this validation it was necessary to set the parameters involved in the trust model:

- The confidence degree of the agent for each task type t - $(T^t_{a_i,a_i})$ was defined as 0.5. It means that agents at the beginning are not considered experts of solving a particular task;
- The confidence degree of the agent in other agents - $(T^t_{a_i,a_j})$ was defined as 0.5, i.e., agents do not have a particular opinion about other agents in the initial stage of the simulation.
- τ was set as 0.058; this implies that the evaluations was lasting 90 days for all the types of tasks t. $(e^{-0.058 \cdot 90} = 0)$

Several simulations were run and the results were not considered good comparing to the human expert solution. Analyzing the logs generated by the agents, we noticed a weakness of the model: agents were very severe about the trust on other agents. If a_i did not trust a_j, even if a_j was an expert in the type of task t, a_i did not consider a_j to share information when necessary. For example, a_1 does not trust in a_3 for t_1 because in the past it shared information that resulted in poor recommendation, but in this moment a_3 has became an expert in t_1

In order to solve this issue, we improved the method defining three probability values to be used in choosing who the agent will communicate with. A configuration file was created with three probability values to a_i communicates with a_j:

- *Trusted* agents: it represents the agents a_i trusts in the moment;
- *New* agents: it represents the agents a_i has not communicate yet which means that a_i does not have opinion about these agents yet;
- *Old* agents: it represents the agents a_i has already trusted in previous recommendation cycles but due negative evaluations, it does not trust these agents in the moment.

With these values, the agent will not always choose agents it trusts but eventually it might choose agents it does not know yet or agents it already considered reliable in the past.

Figure 7. XML file with the probability values

```
<object.ConfigurationXML>
  <percTrustedAgents>60</percTrustedAgents>
  <percNewAgents>30</percNewAgents>
  <percOldAgents>10</percOldAgents>
</object.ConfigurationXML>
```

Figure 7 shows the configuration file example and the values set: 60% of trusted agents, 30% of new agents and 10% of old agents. We got to these ideal values after run more queries.

Table III illustrates the sum of positive evaluations obtained from agents in this second experiment. We can see that in the second set of queries the results started to increase but in the third set of queries the quality decreased again. However, as grow the number of positive evaluations, the results improved and from the forth set of queries the results started to increase.

TABLE III.
USERS EVALUATIONS WITH TRUST - WTRUST

Ranges	Number of positive evaluations (rate="Liked it")			
	Flight	Hotel	Attraction	Total
1 - 10	28	31	8	67
11 - 20	40	27	6	73
21 - 30	30	28	9	67
31 - 40	40	25	10	73
41 - 50	40	26	9	75
51 - 60	40	31	10	81

Figure 8 shows the results obtained from NTrust and WTrust strategies compared with the evaluations obtained from the human expert solution (the total values obtained

by the human expert in each group of recommendations were 76, 72, 70, 71, 73 and 78). This comparison is very interesting because we can see that in the first group of recommendations the agents had good evaluation and in the second one the results decreased a little bit.

However, from the third group we can see clearly that agents with the trust model improved their results. It means that the communication with trusted agents improved the quality of the information exchanged between them.

The Kruskal-Wallis test showed that the results are significant (p-value 0), comparing NTrust, WTrust and the human expert solution.

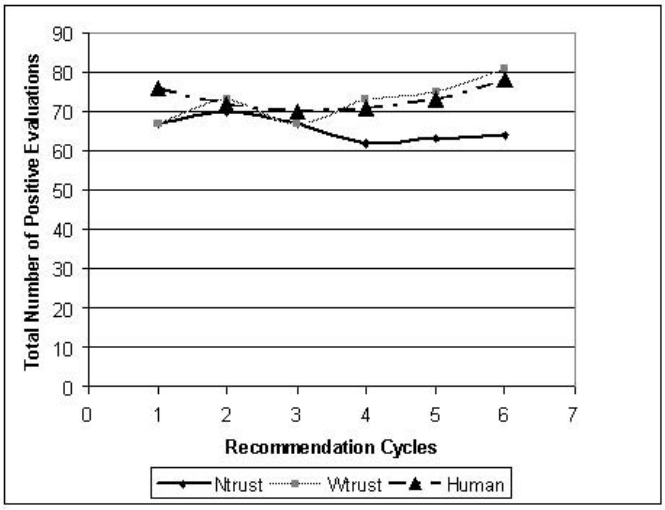

Figure 8. Comparison among results (NTrust x WTrust x Human)

Another relevant result is that WTrust strategy had better results (from the third group of recommendations) than the human expert. We showed the results to the human expert and after analyzing the results we reached a conclusion: when there is and exchange of information between agents the results are better. The human expert told us that when more than one travel agent work in a recommendation the result is better because the knowledge of all them is assembled in the solution presented to the customer.

Analyzing the logs that were created during the recommendation cycles it was possible to grasp some more specific details about the agents behavior:

- $Agent_5$ initially chooses tasks of hotel but as new queries came in it became an expert in flight tasks;
- $Agent_7$ gradually disappeared from the communication among agents. It means that agents realized they did not receive good evaluations when they got information from $Agent_7$;
- Agents only communicated with trusted agents.

V. Conclusion and Future Work

This paper presented a trust model applied to a multiagent recommender system where agents choose which agents to communicate in order to exchange information based on how much they trust them.

The proposed trust model has two important features: 1) agents become experts in specific types of tasks during the recommendation cycles, using the computed confidence degree, and 2) the communication process between agents is improved using the trust degree, i.e., avoiding unnecessary communication.

Although the trust method is straightforward, the trust degree is based on previous interactions between agents and it may increase or decrease over time. This is what happen in real scenarios, as in a travel agency.

Moreover, the confidence degree brings an interesting feature for the recommender system: the agents specialize in solving particular types of tasks. These features help the multiagent recommender system to improve the results presented to the user.

As future work we are studying how a network of trustful agents may help an agent. For example, if agent a_i trusts agent a_j but agent a_j does not have the information a_i needs then it may recommends another agent from its trust. With this feature agents will have more agents to share information which will decrease the probability of searching for the information into the web.

References

[1] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings ACM Conference on Computer-Supported Cooperative Work*, 1994, pp. 175–186.

[2] A. Goy, L. Ardissono, and G. Petrone, "Personalization in e-commerce applications," in *The Adaptive Web*. Springer Berlin / Heidelberg, 2007, pp. 485–520.

[3] Y. Z. Wei, N. R. Jennings, L. Moreau, and W. Hall, "User evaluation of a market-based recommender system." *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 2, pp. 251–269, 2008.

[4] M. Witkowski, A. Artikis, and J. Pitt, "Experiments in building experiential trust in a society of objective-trust based agents," in *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies*. London, UK: ACM, 2001, pp. 111–132.

[5] L. Drumond, R. Girardi, and A. Leite, "Architectural design of a multi-agent recommender system for the legal domain," in *ICAIL '07: Proceedings of the 11th international conference on Artificial intelligence and law*. New York, NY, USA: ACM, 2007, pp. 183–187.

[6] D. Zhang, S. Simoff, S. Aciar, and J. Debenham, "A multi agent recommender system that utilises consumer reviews in its recommendations," *International Journal of Intelligent Information and Database Systems*, vol. 2, no. 1, pp. 69–81, 2008.

[7] S. Macho, M. Torrens, and B. Faltings, "A multi-agent recommender system for planning meetings," in *Workshop on Agent-based recommender systems*, 2000.

[8] F. Ricci, "Travel recommender systems," *IEEE Intelligent Systems*, vol. 17, no. 6, pp. 55–57, 2002.

[9] D. Camacho, J. M. Molina, D. Borrajo, and R. Aler, "Mapweb: Cooperation between planning agents and web agents," *Information & Security: Special issue on agent-based Technologies*, vol. 8, no. 2, pp. 209–238, December 2002.

[10] M. Torrens, B. Faltings, and P. Pu, "Smartclient: Constraint satisfaction as a paradigm for scaleable intelligent information systems," *CONSTRAINTS: an International Journal*, vol. 7, no. 1, pp. 49–69, 2002.

[11] S. D. Ramchurn, D. Huynh, and N. R. Jennings, "Trust in multi-agent systems," *The Knowledge Engineering Review*, vol. 19, no. 1, pp. 1–25, 2004.
[12] A. R. C. Hussin, K. Keeling, L. Macaulay, and P. McGoldrick, "A trust agent for e-commerce: Looking for clues," in *EEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 286–289.
[13] N. Nassiri, "Increasing trust through the use of 3d e-commerce environment," in *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2008, pp. 1463–1466.
[14] P. Victor, C. Cornelis, M. De Cock, and P. Pinheiro da Silva, "Gradual trust and distrust in recommender systems," *Fuzzy Sets Syst.*, vol. 160, no. 10, pp. 1367–1382, 2009.
[15] J. O'Donovan and B. Smyth, "Trust in recommender systems," in *IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces*. New York, NY, USA: ACM, 2005, pp. 167–174.
[16] M. Montaner, B. López, and J. L. de la Rosa, "Developing trust in recommender agents," in *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2002, pp. 304–305.
[17] P. Massa and B. Bhattacharjee, "Using trust in recommender systems: an experimental analysis," in *In Proceedings of iTrust2004 International Conference*, 2004, pp. 221–235.
[18] F. Lorenzi, A. L. C. Bazzan, and M. Abel, "Multiagent truth maintenance applied to a tourism recommender system," in *Tourism Informatics: Visual Travel Recommender Systems, Social Communities and User Interface Design*, N. Sharda, Ed. Information Science Reference, 2009, pp. 54–72.

Fabiana Lorenzi is currently a Ph.D. candidate at Universidade Federal do Rio Grande do Sul, Brazil. She received her MS degree in computer science from Universidade Federal do Rio Grande do Sul, in 1998. She is a lecturer at the Universidade Luterana do Brasil, Brazil since 1997 and her research interests include multiagent recommender systems, case-based reasoning and knowledge management.

Gabriel Baldo is currently attending a post-graduate in Project Management with emphasis in Information Technology at the Pontifical Catholic University of Rio Grande do Sul. He received his degree in computer science from the Lutheran University of Brazil, in 2010. His research interests include Project Management and Artificial intelligence focused in multiagent systems and recommender systems..

Rafael Bianchi Pereira da Costa is currently attending a post-graduate in Management and Governance of Information Technology at the Lutheran University of Brazil. He received his degree in computer science from the Lutheran University of Brazil in 2009. His research interests include java environments, multiagent systems and recommender systems.

Mara Abel is graduated in Geology and doctor in Computer Science, Artificial Intelligence. She is a professor in Knowledge Engineering at the Institute of Informatics of Universidade Federal do Rio Grande do Sul (UFRGS), Brazil, where she leads several initiatives in entrepreneurship. Her main focus on research is related to study alternatives for representing visual knowledge, developing applications in petroleum reservoir analysis area. She is also the co-founder of the company ENDEEPER Rock Knowledge System, an spin-off of her research group.

Ana Bazzan received her PhD in 1997 from the University of Karlsruhe, Germany. From 1997 to 1998, she had a postdoc research associate position in the Multi-Agent Systems Laboratory at the University of Massachusetts in Amherst, under the supervision of Prof. Victor Lesser. In 1999 she joined the Institute for Informatics at UFRGS as an Associate Professor and got tenure 3 years later. From April 2006 to March 2007 she had an appointment at the University of Wrzburg (Germany), as a fellow of the Alexander von Humboldt Foundation. Her research interests include: Multiagent Systems, Game-Theoretic Paradigms for Coordination of Agents, Multiagent Learning, Coordination and Cooperation in MAS, Agent-Based Simulation, Iterated Prisoner's Dilemma and other games, Swarm Intelligence, RoboCup Rescue, and Traffic Simulation and Control.

Francesco Ricci is associate professor of computer science in the Database and Information Systems group at the Free University of Bozen-Bolzano, Italy. He received the doctoral degree in Mathematics from the University of Padova in 1983 and from 1984 to 1986, he was a Ph.D. student at Scuola Normale Superiore, Pisa, Italy. Since 1988 he has been at ITC-irst, where he has been responsible for European projects and internal divisions (Expert System group, Knowledge Representation Lab.). His current research interests include recommender systems, constraint satisfaction problems, machine learning, case-based reasoning and software architectures.