

# An Architecture for a Multiagent Recommender System in Travel Recommendation Scenarios

Fabiana Lorenzi and Ana L.C. Bazzan and Mara Abel<sup>1</sup>

**Abstract.** This paper presents an architecture of a multiagent recommender system applied to the tourism domain. The system environment has several agents working in the travel packages recommendation process, where each agent has its own knowledge base and is responsible for a task of the process such as booking flights, hotels and the attractions. The agents have a reasoning maintenance element that helps to keep the integrity of their knowledge bases.

## 1 Introduction

Recommender systems are being used in e-commerce web sites to help customers to select products more suitable to their needs and preferences [6]. The recommendation can be done using information about the user, about other users, about the previous recommendations performed or even information got from the internet.

However, the information available from the internet is unorganized and distributed on server sites all over the world. The tourism domain, for example, consists of several tour operators services and lots of data sources from the travel agencies distributed over the internet. Many services are not always available and sometimes information can be ambiguous or erroneous due to the dynamic nature of the information sources and potential information updating and maintenance problems.

The motivation of this work came from the tourism domain problem in recommend best travel packages to users. In the regular process of travel recommendation, the traveler goes to a travel agency to get help to plan her/his trip. The result of such a planning is a travel package. This is not an easy task because it involves different flights, booking hotels, timetable from attractions and further information. For example, if the traveler asks for 15 days in Rio de Janeiro and Salvador, the travel agent needs to compose the whole travel package, including flights, the hotels in both cities, and the possible attractions to visit in each place.

The major problem is that the travel agent needs knowledge about all these items to compose the travel package according to the user's preferences. Normally, the knowledge about all these items is distributed in different sources and it needs an extensive work to search the different pieces and put them all together. The problem of locating information sources, accessing, filtering, and integrating information to compose a travel package has become a very critical task.

In order to deal with these problems we propose a multiagent recommender system. Agents can be used to retrieve, filter and use information relevant to the recommendation. Multiagent systems can divide specialized task knowledge, avoiding unnecessary processing,

and can be built to deal with dynamic changes in the information source.

This paper shows the architecture of a multiagent recommender system where each agent plays the role of an expert of some component of the travel package and helps to assemble the whole travel package. Each agent has its own knowledge base but they are not specialized in some component. For this reason, the agents are able to exchange information among themselves when needed. In this scenario, each agent is responsible for one piece of information (e.g. one travel package's component) and after the whole search, we can expect the system to have a travel package to recommend to the user. Agents work in a distributed and cooperative way, sharing and negotiating knowledge and recommending the best product to the user.

In this kind of distributed multiagent scenario, the maintenance of the agents' knowledge bases is important and the recommendation efficiency depends on it. To deal with the problem of keeping the integrity of the knowledge base of each agent, the system is based on the idea of the beliefs revision - a way of reviewing the assumptions of each agent.

Section 2 shows the related work and section 3 describes the proposed architecture to the multiagent recommender system. Section 4 shows a case study in the tourism domain. Finally, in section 5 we discuss some advantages and some open issues related to the development of the proposed architecture.

## 2 Related Work

The growth of e-Commerce has brought the need of providing a personalized attendance to a huge number of users through the Internet. Several domains are been used in recommender system development. The tourism domain, for instance, has attracted the attention of several researches in the recommendation field. One example is DieToRecs [7], a travel recommender system that suggests both single travel services (e.g. hotel or an event) and complete travel plans comprising more than one elementary service. DieToRecs is able to personalize the current recommendation based on previously stored recommendation sessions (view as cases).

The travel recommendation is a tricky task and it still has some open issues such as different information sources distributed over the internet and the specific knowledge needed to compose a travel. In order to deal with these issues, a new kind of recommender systems has been developed: the multiagent recommender system.

The multiagent recommender system uses agents to help in the solution process, trying to improve the recommendation quality. The agents cooperate and negotiate in order to satisfy the users, interacting among themselves to complement their partial solutions or even to solve conflicts that may arise.

<sup>1</sup> Universidade Luterana do Brasil, Canoas, Brasil, email: lorenzi@ulbra.tche.br and Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, email: lorenzi,bazzan,marabel@inf.ufrgs.br

In [9] the authors investigated the viability 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. Some points of this application can be discussed: 1) the market correlates the agents' internal valuation and the user's valuation of the recommendations by invoking a bidding and a rewarding regime. If, for some reason, the user does not evaluate the recommendations, maybe this valuation will not be done correctly; and 2) auctions are used to decide the recommendation winners and the agents who provided good recommendations (chosen by the user) receive some reward in return. Those agents become richer and are able to get their recommendations advertised more frequently than the methods whose recommendations are infrequently chosen by the user. This behavior avoid the serendipity and maybe the system will never surprise the user with unexpected recommendation.

Another example of multiagent case-based recommender system is CASIS [3]. The authors proposed a metaphor from swarm intelligence to help the negotiation process among agents. The honey bees' dancing metaphor is applied with case-based reasoning approach to recommend the best travel to the user. The recommendation process works as follows: the user informs his/her preferences; the bees visit all cases in the case base and when they find the best case (according to the user's preferences) they dance to that case, recruiting other bees to that case; and the case with the most number of bees dancing for it is the one recommended to the user.

The advantage of this application is that the bees always return something to recommend to the user. Normally, case-based recommender systems use pure similarity to retrieval the best cases. The recommendation results depend on the defined threshold and sometimes the system does not find cases that match the threshold. Despite the controversy that sometimes is better not recommend instead of recommend wrong products, this is not true in the tourism domain. CASIS system always return some recommendation, which is specific for the current season, given the dynamic nature of the approach).

### 3 An architecture for a multiagent recommender system

Recommender systems have providing alternatives to deal with the problem of information overload in the internet [8]. The user describes his/her preferences and the system recommends the best product according to what was described by the user. However, in some domains, it is possible that a single information source does not contain the complete information needed for the recommendation.

For example, in a travel recommender system the travel package is composed by several components such as flights, hotels and attractions. These components cannot be found in just one information source in the travel agency. Normally, the flights come from the flight companies systems (like *Sabre* from United Airlines or *Amadeus* from Varig); the hotels have their own websites and each attraction such as a museum also has its own website where its timetable is published. In order to recommend some travel package to the user, it is necessary to search for each information in many different sources and further assemble all the components together.

The travel package recommendation was modeled using the ar-

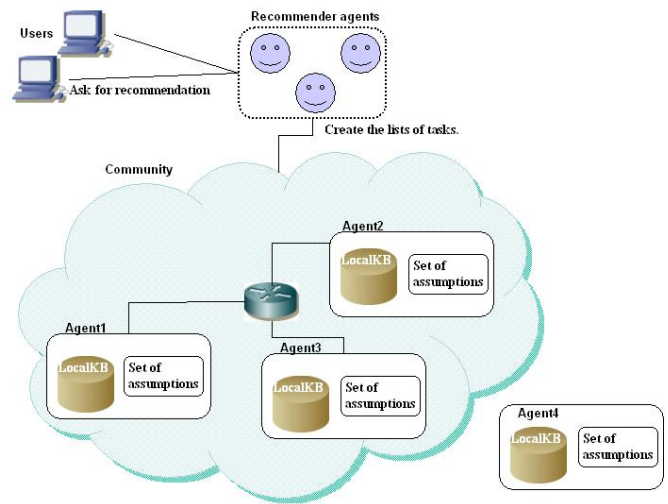


Figure 1. The architecture of the multiagent recommender system

chitecture shown in figure 1. The multiagent recommender system proposed in this work has a group of agents with a common global goal (the recommendation) and separate individual goals (the component that each one has to search). The agents are distributed over the network (in the travel agency for instance) and there are two types of agents:

- The **Travel Agent** (TA) agent is the agent responsible for the final recommendation. When the user asks for some recommendation, the TA agent starts the process. It creates a list of tasks (the components of the travel package) according to the user's preferences and communicates it to all the agents within the community. The *community* is composed by the agents who are within a defined range of action. Even within this range, we assume that failures may happen sometimes. Besides, the agents can be too far to hear the requests or cannot provide an answer.
- The **Components** (CP) agents are responsible for searching the components of the travel package that will be recommend to the user. Each agent picks a task in the list.

An important feature of this architecture is that the CP agents are not specialized in a specific component of the travel package. They can pick different components in different recommendation cycles. This allows the CP agents to exchange information among themselves. The CP agents cooperate in the recommendation process, communicating with each other (through broadcast) and sharing knowledge.

Each CP agent has two important elements: its own knowledge base (called LocalKB) where they store the information they search in the internet; and a reasoning maintenance system that includes a mix of an Assumption based Truth Maintenance Systems (ATMS) [1] and a Distributed TMS [2]. The agent keeps a set of assumptions in this element [4]. These assumptions come from the exchange information between the agent and the others agents. An important aspect of this architecture is that the agents must be able to assess and maintain the integrity of the communicated information and their KBs.

Another important feature of our architecture is the communica-

tion time consumed by the agents. In this distributed scenario, agents can be in the same or in different computers and the required information can be in the travel agency network or in the internet. For this reason we defined three different levels of information search: 1) the CP agent looks for the information in its own knowledge base; 2) the CP agent looks for the information in the community, asking for the agents of the community if some of them has the information it needs; 3) the CP agents go to the internet to search for the information.

These different levels lead to a good performance of the system, avoiding unnecessary searches because most of the requirements can be solved in the two first levels of communication. However, in the beginning of the process, when the KBs are empty, the agents will search more frequently in the internet.

#### 4 Case Study - A Travel Recommendation

In this section, we present a case study with the application of the multiagent recommender system architecture in the tourism scenario. The system recommends travel packages to the user. Each travel package is composed by flights, accommodation and attractions. The agents work in a cooperative way to recommend the best travel package to the user. They can communicate with each other and are able to share their knowledge.

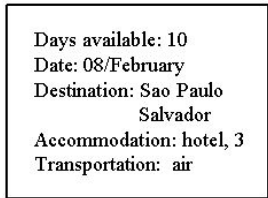


Figure 2. Users' preferences

The recommendation cycle starts when the user specifies his/her preferences (as shown in figure 2). After these preferences are given, a list of tasks is created by the TA agent. Each preference chosen by the user becomes an item (also called a component of the travel package) in the list. The CP agents then pick a task from this list. When a CP agent selects a task, it is marked as "already allocated" and no other agent can select it (in the future we will study whether the system's performance could be improved if we allow more agents to carry out the same task; possibly there will also be a cost regarding inconsistencies). According to the user's preferences example shown in figure 2, the list was created with 3 tasks: 1) destinations and attractions; 2) type of transportation; and 3) type of accommodation.

Each CP agent knows its identification, its own knowledge base (LocalKB) and its set of assumptions (information exchanged with other agents). Initially, all the agents have their LocalKBs empty and these LocalKBs are incremented according to the utilization of the system.

The communication among the agents is done through exchange of messages. The first communication protocol used in this architecture is the *Choosing tasks* Protocol, shown in Figure 4. The TA agent sends a broadcast message to the community indicating that the list is available. To pick a task from the list, each CP agent has to start a communication with the TA agent. The TA agent sends a list with the available tasks to the CP agent. It chooses (randomly) its task

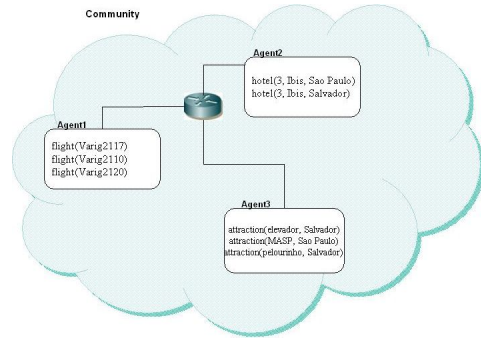


Figure 3. Agents in the 1st cycle of recommendation

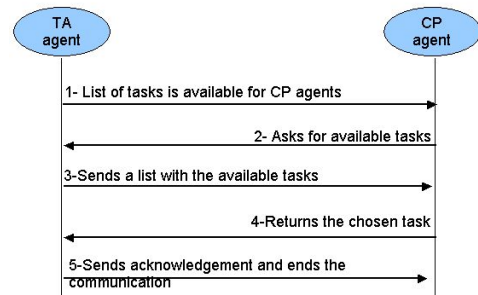


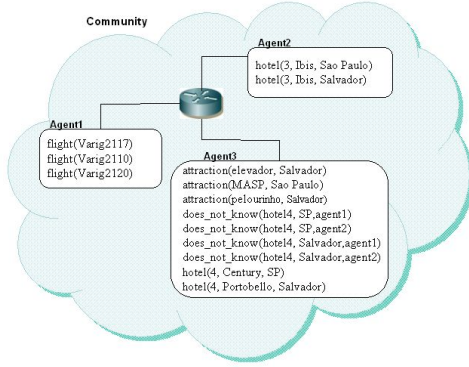
Figure 4. Communication protocol - Choosing tasks

from the list, informing the chosen task to the TA agent. The TA agent sends an acknowledgement and ends the communication. This communication protocol must be atomic to avoid inconsistencies in the list of tasks. Figure 3 shows that in this example, only three CP agents were in the community in the moment that the TA agent has created the list of tasks.

In the example, *Agent1* has selected the task *type of transportation* and it was responsible for the flights to the destination chosen by the user. In the web search, *Agent1* found three possible flights. *Agent2* has selected the *accommodation* task and found information about three-star hotels in Sao Paulo and Salvador. *Agent3* is responsible for the attractions. Due to a failure or due to its position (outside the range), *Agent4* is outside the community and it did not listen to the request. That is why it does not appear in figure 3.

If the user criticizes the recommendation and, for example, changes the type of wished accommodation, the recommendation cycle restarts. The list of tasks is recreated only with this new preference. In this new cycle, the agents can choose different tasks from the list (even because the list is dynamic so the tasks change according to the new preferences defined by the user). Now the user asked four-star hotels and s/he expects a new recommendation. In this cycle, the agents already have information in their LocalKB and they search first in their own KB. If they do not find the information needed, they start the communication with other agents.

The communication protocol in this case is called *Assistance Protocol* and it involves the CP agent that is searching for information and the community. The CP agent send a broadcast message to the community telling that it needs information *i* to perform its task *t*.



**Figure 5.** Agents in the 2nd cycle of recommendation

The agents from the community answer if they have or not the information. As we can see in figure 5, *Agent3* was responsible for searching information about accommodation. It did a broadcast to the community asking for information about four-star hotels. *Agent1* and *Agent2* answered the request, telling that they do not have the information and *Agent3* store this fact in its LocalKB.

Let us assume that the user criticized the recommendation one more time and now s/he changed the destination, choosing Sao Paulo and Rio de Janeiro. The agents choose their tasks from the list and now *Agent4* is responsible for the four-star hotels. This agent has not been introduced to the other agents yet because in the previous cycles it was not in the boundaries of the broadcast. So, it introduces itself to the community and gets to know the other agents. It learns that *Agent3* has information about four-star hotels and they exchange information. However, it learned now that the information is about four-star hotels in Sao Paulo and Salvador. *Agent3* has not information about four-star hotels in Rio de Janeiro.

It is important to notice that the agents cannot always hear the messages. They can be away from the boundaries of the broadcast. This leads to inconsistencies in their LocalKBs and also in their set of assumptions. That is a serious problem when they try to exchange information. This work helps in the negotiation among the agents, trying to keep the integrity of their KBs. The TMS element of each agent helps to keep the KBs integrity and allows the agents to exchange information in a reliable way.

Another point that our approach considers is the fact that travel package recommendation needs specific knowledge to each component of the package. The information needed can be distributed in different sources. With its own KB and the possibility of exchange information among all the community members, the agents help in the recommendation, working in a distributed and cooperative way.

## 5 Concluding Remarks

Recommender systems are been used in e-commerce to help users to find out better products according to their needs and preferences. Knowledge-based approach tries to exploit the knowledge about the user to improve the recommendation. In this paper, we presented an architecture of a distributed multiagent recommender system where the agents work together to search information that fit to the users preferences.

In this multiagent architecture, each agent has its own knowledge

base and its set of assumptions. All the agents communicate to each other to exchange information, avoiding unnecessary processing because they first check their knowledge bases and the community of agents. Only if they do not find information in their LocalKB or in the community then they search in the internet.

The proposed architecture is under development. Preliminary studies show some positive feedback:

- The use of agents in the distributed recommender system in the tourism domain returns good recommendations, according to the user's preferences. The agents have the knowledge needed and they are able to create the recommendation as the human travel agent would do it;
- The two different kind of agents help in the recommendation process performance. TA agents are responsible for creating the whole recommendations. CP agents are responsible for the searches;
- The TMS element in each CP agent helps to keep the data integrity of the knowledge bases. Each time an agent talks to another one to ask for information, it is necessary to validate the agent's beliefs, otherwise inconsistencies can appear.

As future work, we intend to increase some features in the approach such as including more components in the list of tasks and evaluate what happen if two agents pick the same tasks, maybe some competitive behavior can show up. We want also to include some task allocation approach to improve the process where the CP agents choose their own tasks.

Another point to be investigated is the feasibility to convert the agent's knowledge base into a case base. It means that each agent can have its own case base and tries to recommend using past cases. As shown in [5], distributed case-based reasoning is being used in a variety of application contexts.

## REFERENCES

- [1] J. de Kleer. An assumption-based tms, extending the atms, and problem solving with the atms. *Artificial Intelligence*, 28(2):127–224, 1986.
- [2] Michael N. Huhns and David M. Bridgeland. Multiagent truth maintenance. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1437–1445, November/December 1991.
- [3] Fabiana Lorenzi, Daniela S. Santos, and Ana Lucia C. Bazzan. Negotiation for task allocation among agents in case-base recommender systems: a swarm-intelligence approach. In Esma Aimeur, editor, *Proceedings of the Workshop Multi-Agent Information Retrieval and Recommender Systems - Nineteenth International Conference on Artificial Intelligence (IJCAI 2005)*, pages 23–27, July 2005.
- [4] Omitted. Distributed assumption-based truth maintenance applied to a multiagent recommender system. In *Third European Starting AI Researcher Symposium (STAIRS) - under submission*, August 2006.
- [5] E. Plaza and L. McGinty. Distributed case-based reasoning. *The Knowledge Engineering Review - To appear*, 0(0):00–00, July 2006.
- [6] 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*, pages 175–186, 1994.
- [7] Francesco Ricci, Adriano Venturini, Dario Cavada, Nader Mirzadeh, Dennis Blaas, and Marisa Nones. Product recommendation with interactive query management and twofold similarity. In Agnar Aamodt, Derek Bridge, and Kevin Ashley, editors, *5th International Conference on Case-Based Reasoning*, pages 479–493, Norway, June 2003.
- [8] J. Schafer, J. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2):115–153, 2001.
- [9] Y. Wei, L. Moreau, and N. Jennings. Recommender systems: A market-based design. In *Proceedings Second International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS03)*, pages 600–607, Melbourne Australia, July 2003.