A Multi-Agent Recommendation System Based on the Sharing of Knowledge Owned by Communities

Youssef ACHBANY[*], Tai NGUYEN[*], Stéphane FAULKNER[+],
Manuel KOLP[*], Yves WAUTELET[*]

{achbany, nguyen, kolp, wautelet}@isys.ucl.ac.be
stephane.faulkner@fundp.ac.be

[*] IAG – Institut d'Administration et de Gestion, ISYS – Unité de Systèmes d'Information,
Université Catholique de Louvain, Place des Doyens 1, 1348 Louvain la Neuve, Belgium

[+] FUNDP - Information Management Research Unit
University of Namur, Rue de Bruxelles 61, 5000 Namur, Belgium

Abstract. To find information of quality from multiple heterogeneous sources is increasingly difficult. This problem finds its origin in the fast growth of the number of information available. The various formats used to store the data can also be problematic for the recommendation process. Nevertheless, for a community of people with similar interests, quality of results can be improved exploiting also the experience of the community. This experience can be shared and strengthen by all members of the same community to respond to their expectations by furnishing the best information on a specific topic. We propose an agent-based recommendation system for supporting communities of people in their searching task.

1. Introduction

The use of Internet as principal source of information among multiple heterogeneous sources was quickly spread, in particular thanks to its impressive quantity of information placed at the disposal of the users. The consequence of this situation consists in a constantly increasing number of hosts and information on the Internet. The pages become appreciably bigger, returning the search for required information more complex. A survey on user search behaviour data [7] show that user rarely goes further than the second page of results provided by a search engine. It means that web search tools should improve quality of the results on the first two pages.

An important issue currently not addressed by most of search engines is the consideration of *user profiles* during the searching process. Indeed, most of search engines base their searching process from the user keywords, without regard to the meaning of theses keywords for the users (i.e., the context where theses keywords are significant for the users). Yet, this could lead to a high score for a page simply because it contains the given keywords. However this page

could be valueless for the user because of the context. The difficulty for users to find what they are looking for, among the provided selection of links, is especially raised when searched words are overloaded with multiple meanings. The context must be taken into consideration to determine a narrower range of search results. Unfortunately, since users typically do not or cannot provide such a context, most of search engines can only rely on the occurrence of user-provided keywords.

A solution to cope with these issues (i.e., quantity of information and consideration of the user profile) is to filter the information on the basis of experiment held by a community of people (whence the collaborative filtering term) sharing the same interest. We define the concept of people community to gather experiences having the same semantic (i.e., experiences that can be beneficial for every community members) together.
So, to find information of quality, a recommendation system has to combine the creation of communities and the use of collaborative-filtering methods on the basis of the community experiences and the user profile.

However, such recommendation system is difficult to implement when the collection of information to analyze is too significant and the collaborative-filtering methods are too heavy (i.e., most of collaborative-filtering methods need an important computation capacity). We need to choose an approach that is distributed, flexible and capable of certain autonomy.
Distributed system is a collection of autonomous entities connected which enable the coordination of theirs activities and the sharing of the resources. Contrary to centralized systems, distributed approach provides an important computation capacity essentially thanks to tasks distribution. Given this definition and to be efficient in term of computation capacity, we need to select a distributed architecture (especially by using collaborative-filtering methods) for our recommendation system.

Flexible and autonomous requirements refer to a system with a high degree of adaptability and capable of autonomous action in order to meet its design objectives. A recommendation system with these requirements can take decisions himself (e.g., which members of the community ask for a recommendation, how to ranks the results …) and adapts its behaviour according to user profile, community experiences and so on.
These three characteristics (i.e., distributed architecture, flexibility and adaptability) are found naturally in the multi-agent paradigm.

In this perspective, multi-agent systems (MAS) seem to be popular to build robust and flexible application (Wooldridge and Jennings, 1994) by distributing responsibilities among autonomous and cooperating agents. An agent is a computational entity that can be viewed as perceiving and acting upon its environment and that is autonomous in that its behaviour at least partially depends on its own experience. These requirements make possible for a MAS to cope with problems (e.g., the processing of huge amounts of data, or of data that arises at geographically distinct locations, the requirement of more autonomy,

…) that conventional systems (majority being systems centralised) were not able to solve.

Agents and multi-agent systems applied to search area are reported in literature. The main proposal is to use an agent that assists its user during web search [4,8,12]. The agent can track user browsing or it can form user profiles in different areas in order to anticipate items of interest. Multi-agent systems aimed to help user during web search implement various approaches. It can be alliance of several agents providing user with result [10]. It is also possible to apply auction protocol and reward mechanism to agent collaboration [13]. Other authors [9,3,14] propose personal agents acting on behalf of their users, collaborating with each other and having the goal to improve their user' browsing. In some of the systems considered so far user is supposed to perform an extra work during search, e.g. he/she needs to specify the areas of his/her interest or to analyze a lot of results of searches similar to the current one. Sometimes there are also restrictions like ability to use only certain part of pre-defined knowledge or ontology.

This paper presents a multi-agent recommendation system based on the concepts of knowledge sharing. The system is a generalization of Collaborative Filtering that is a technique by which the user interest for information is predicted from the knowledge of the other user profile. We compute similarities between users to cluster them into groups with similar interest (i.e., community). To compute the similarities, we use an innovative collaborative filtering method based on the Markov-chain model, the random-walk model

The paper is organized as follows. The concepts of agent and multi- agent Systems are described in Section 2. Section 3 gives description of the architecture of our recommendation system. The search process is also explained in this Section. Section 4 details the collaborative-filtering method based on the Markov-chain model. Section 5 details the external search process. Finally, in Section 6, we give the conclusion.

2. Agent and Multi-Agent Systems

An agent defines a system entity, situated in some environment that is capable of flexible autonomous action in order to meet its design objective.

Three key concepts support this definition:

- Situatedness: an agent receives input from the environment in which it operates and can perform actions, which change the environment in some way;

- Autonomy: an agent is able to operate without direct, continuous supervision, it has full control over its own actions;

- Flexibility: an agent is not only reactive but also pro-active. Reactivity means that it has perceptions of the world inside which it is acting and reacts to change in quasi real-time fashion. Proactiviness means that behavior is not exclusively reactive but it is also driven by internal goals, i.e., it may take initiative.

From this, a multi-agent system can be defined as an organization composed of autonomous and proactive agents that interact with each other to achieve common or private goals.

MAS may be either cooperative or competitive agents. In cooperative MAS, the system has a global goal (or set of goals) and the agents that compose the MAS cooperate, possibly by performing diverse tasks, in order to achieve the global goal. This kind of systems is typically adapted to perform distributed problem solving. There is a unique high-level goal decomposed recursively into parallel activities to be performed by a set of agents.

In competitive MAS, each of the component agents has its own set of goals that may or may not meet those of other agents. In this case the MAS is an architecture that allows agents to interact, each one to pursue personal goals and defend its own interests. This kind of systems meets typically engineering requirements of e-commerce, information retrieval applications, web services or peer-to-peer networks. In such environments, every agent generally represents either a client, who wants to obtain some resources or have some service accomplished, or a provider, who wants to sell resources or services at a certain (not necessarily financial) cost. Each agent pursues the goals of the (human or system) actor it represents, and these goals can usually be in conflict.

In order to reason and act in an autonomous way, agents are usually built on rationale models and reasoning strategies that have roots in various disciplines including artificial intelligence, cognitive science, psychology or philosophy. An exhaustive evaluation of these models would be out of the scope of this paper or

even this research work. Agent models are proliferating; some include learning capabilities, others intelligent agendas based on statistics, others yet are based on genetic algorithms and so on. However, a simple yet powerful and mature model coming from cognitive science and philosophy that has received a great deal of attention, notably in artificial intelligence, is the Belief-Desire-Intention (BDI) model [Brat88]. This approach has been intensively used to study the design rationale of agents and is proposed as a keystone model in numerous agent-oriented development environments such as Jack or Jade. The main concepts of the BDI agent model are (except the notion of agent itself we have just explained):

Beliefs that represent the informational state of a BDI agent, that is, what it knows about itself and the world; Desires (or goals) that are its motivational state, that is, what the agent is trying to achieve; Intentions that represent the deliberative state of the agent, that is, which plans the agent has chosen for possible execution.

In more detail, a BDI agent has a set of plans, which defines sequences of actions and steps available to achieve a certain goal or react to a specific situation. The agent reacts to events, which are generated by modifications to its beliefs, additions of new goals, or messages arriving from the environment or from another agent. An event may trigger one or more plans; the agent commits to execute one of them, that is, it becomes intention.

Plans are executed one step at a time. A step can query or change the beliefs, performs actions on the external world, and submits new goals. The operations performed by a step may generate new events that, in turn, may start new plans. A plan succeeds when all its steps have been completed; it fails when certain conditions are not met.

3. Architecture and process descriptions

3.1. Architecture

This section describes the general architecture and process of the system. The aim of the system is to help a community of people to find information that is well suited to their expectation. To attain this goal, we propose a multi-agent system composed by a set of personal agents working within a community to exploit a great amount of knowledge.

A personal agent is assigned to each user of the system and its task is to assists him in his work. Firstly, it helps the user in his searching by finding the good information using the community knowledge and external resources (e.g. search engine like Google). Secondly, it builds a user profile based on his past actions. The user past actions are essentially the information (links) proposed by his personal agent and accepted by him. The most a link is accepted by a user, the most the link become important for the information requested by the user. Finally, the personal agent must compare the profiles of the community members to calculate the similarities between them and its user. Knowing the similarity that exists between the user profile and the others, the personal agent will be able to give an appreciation and a priority to the information which it receives from them. Intuitively information received from a member having a profile rather close to the user has more chance to be accepted by this last than information sent by another with a profile much more distant.

In the design phase, two choices were made concerning the architectures of the global system and of the personal agents. Theses architectures must make it possible for the system and its agents to fulfil their mission of assistance and recommendation in an optimal way. To find the best architecture, we use the organisational styles oriented-agent provided by *Tropos* [20,21] methodology. Organisational styles guide the development of the organizational model for a system.

While basing on *Tropos* organisational styles, our choice has naturally preferred the co-optation at the highest level (the global system). The co-optation is a style that involves the incorporation of representatives of external systems into the decision-making or advisory structure and behaviour of an initiating organization. By co-opting representatives of external systems, organizations are, in effect, trading confidentiality and authority for resource, knowledge assets and support. The initiating system has to come to terms with the contractors which is being made on its behalf, and each co-optated actor has to reconcile and adjust its own views with the policy of the system it has to communicate. Taking into consideration this definition, it is easy to conceive the whole system like a multi-agent system composed by the personal agents representing the external systems.

Figure 2.1 represents the system with a co-optation structure. The personal agents of the system cooperate with each others to share community's experience and to provide a support for collaborative searching. A different external system (external search engine API) is used by the agents to provide new links relatively to the community knowledge.

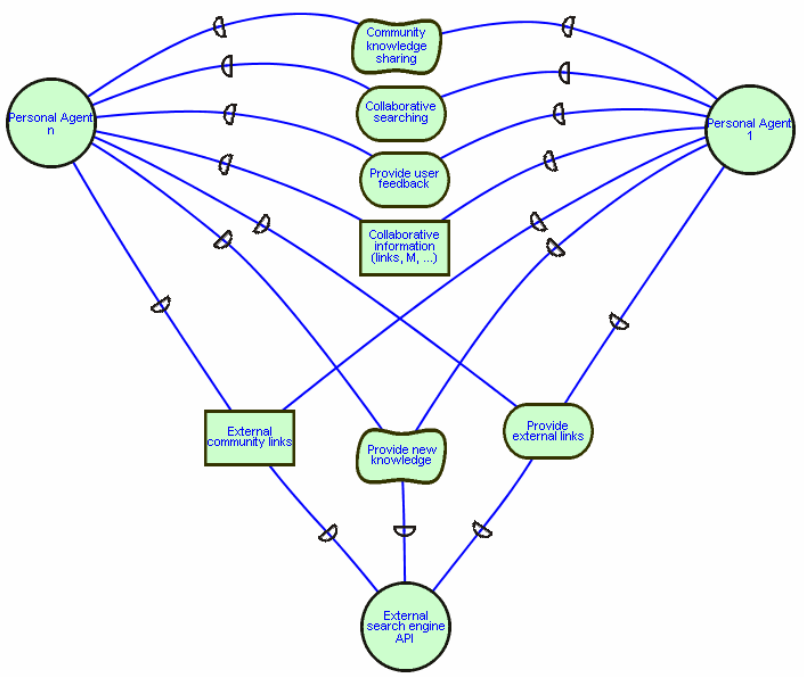

Figure 3-1 : The co-optation style is used to model the interactions of the personal agents at the highest level (i.e., at the platform level).

At a lower level, the personal agents are too complex to be seen like simple system agent. To control this complexity, these agents must themselves be implemented like multi-agent systems. Regarding the personal agent goals and behaviours, our choice has naturally preferred the joint venture. This style is used to model how business stakeholders (individual, physical or social systems) coordinate in order to achieve common goal. Theses alternative styles are developed by organizational theory and strategic alliances literature. The joint venture is a decentralized style that involves an agreement between two or more principal partners in order to obtain the benefits derived from operating at a larger scale and reusing the experience and knowledge of the partners. Each principal partner can manage and control itself on a local dimension and interact directly with other principal partners so as to exchange, provide and receive services, data and knowledge. However, the strategic operation and coordination is delegated to a *Joint Management* actor, who coordinates tasks and manages the sharing of knowledge and resources.

Figure 2.2 shows the instantiation of the joint venture structure of our system. This instantiation is realised using the social patterns provided by Tropos ontology. Unlike organizational styles, they focus on the social structure necessary to achieve one particular goal, instead of the overall goals of the organizations. A social pattern defines the actors (their roles and responsibilities) and the social dependencies that are necessary for achievement of the goal. We use two social patterns, the broker and the mediator respectively for the public and the private interface role. Tropos defines a broker like an arbiter and intermediary who has access services of an actor (Provider) in order to satisfy the request of a consumer. The second pattern, the mediator, mediates interactions among different actor. It has acquaintance models of colleagues and coordinates the cooperation between them.

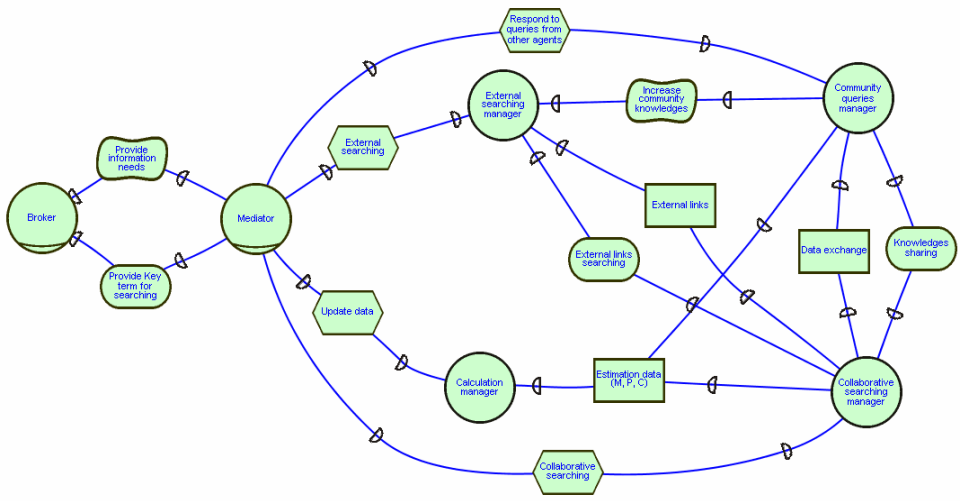

Figure 3-2 : The joint venture style is used at a lower level to model the goals, behaviours, etc. of a personal agent

The role of the *collaborative searching manager* is to answer the user request by carrying out 3 tasks. Firstly, it must check if the user did not already bring an action on same information into the past. If that is the case, it will provide to the user the most relevant links accepted during the preceding requests. For recall, the relevance of a link is measured by the number of acceptance made by the user on this link. Secondly, it must propagate the request near the other members of the community so that the experiment sharing can be done. The propagation will be done to the members having a profile close to that of the initiator of the request. Finally, to allow the introduction of new knowledge (represented by links) within the community, this manager will also have to propagate the request towards an external search engine to the system. Once these three tasks finished, before returning the results to the user, the received links will have to be filtered in order to eliminate the doubles (it is of course possible that the same links are obtained several times for the same

request) and they will have also to be sorted according to their degree of relevance and their source.

The main task of the *external searching manager* is to provide external links to the community by using an external search engine (e.g. Google). Its role is primordial because it supplies the community experience with new knowledge.

The *community queries manager* is responsible for the requests coming from the other community members, and not from its user. Its responsibility extends on two dimensions. Firstly, it must answer the request by using its local experiment, i.e. the links accepted by its own user. Secondly, it must propagate the request near the members who are closest to his profile. The sharing of knowledge within the community is done mainly thanks to this manager. Thus each member will be able to bring his experiment while answering the requests coming from the others.

Finally, the *calculation manager* deals with the calculation of the various data (M, C, P…) necessary to the creation of the profiles, the similarities between profiles and the scores concerning the links. By dedicating a manager specifically for all these calculations, the system will be capable to answer to requests while carrying out these calculations.

3.2. Process Description

In this part, we will describe the process of search for our recommendation system. Following the request of a user, this process provides relevant links towards Web pages. For that, research is based on the keywords of the user, but also on its profile and the knowledge held by the community to which this user belongs.

Taking into account only the knowledge of the user and his community, the system will not be able to answer the requests containing unknown keywords (i.e. keywords for which the user and the members of his community never made request).

To cope with this issue, the result of research must also include links coming from an external source. This external entity will answer a request while being based only on the keywords and without regard to any other information.
Notice that when the keywords of a request are known of a community, it is nevertheless necessary to use an external entity to obtain new links. Indeed, obtaining these new links will increase the user knowledge (and thus the knowledge of his community) by presenting for example links to pages recently created.

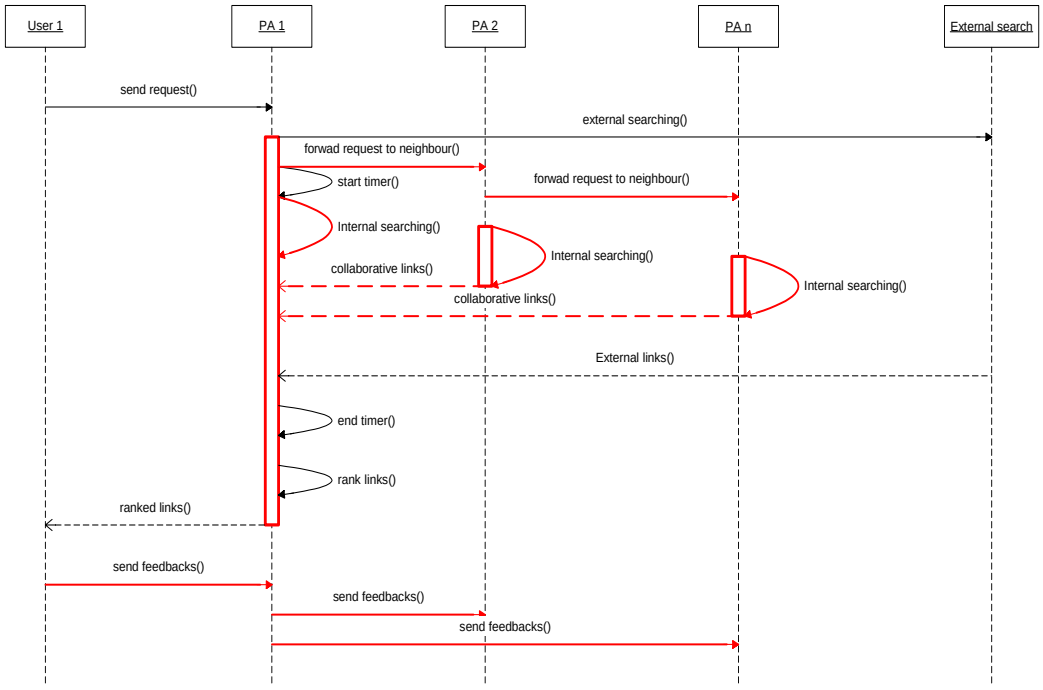

Figure 3-3 : The sequence diagram of the searching process including the internal, collaborative and external search

We will now describe more deeply the searching process. We will separate this process in three different levels of research: intern, collaborative and external (Figure 3-3).

3.2.1. Internal search

To answer the requests of a user, internal search uses knowledge specific to this user. This knowledge was created by using the links accepted by this user in the past. This research is based on the assumption that links accepted by a user in previous requests, can be still accepted by this user in next requests.

3.2.2. Collaborative search

Whereas internal research used only knowledge specific to a user, collaborative research will take into account the set of knowledge specific to a community.

Indeed, this research will use a collaborative-filtering method (this method is presented in the following section) which exploits knowledge of the users having a profile similar to the profile of the search initiator (i.e., the community members of the initiator).

This research is based on the assumption that knowledge of a community can be used by the system to suggest the members of this community the interesting links. In other words, the relevant links for a major part of the community members can be also relevant for the other members of this same community.

3.2.3. External search

Contrary to the two previous modes of research, external search is not based on any knowledge acquired as a preliminary by the user or the community. This research requires an external entity which launches a search using only the user keywords. In our case, this external entity is an external search engine (e.g., Google).

As remind, this external search is of primary importance for our recommendation system. It provides results even for keywords which have never been mentioned in a request. In addition, it increases knowledge by adding new links considered interesting by the user.

4. A Markov-chain model of MAS architecture

4.1. Definition of the weighted graph

A weighted graph G is associated with a MAS (Multi-agents system) architecture in the following obvious way: agents correspond to nodes of the graph and each interaction between two agents is expressed as an edge connecting the corresponding nodes.

In our case, this means that each instantiated agent (i.e. personal agent) corresponds to a node of the graph, and each information exchange about keywords is expressed as an edge connecting the corresponding nodes.

The weight $w_{ij}>0$ of the edge connecting node i and node j (say there are n nodes in total) should be set to some meaningful value, with the following convention: the more important the relation between node i and node j, the larger the value of w_{ij}, and consequently the easier the communication through the edge. Notice that we require that the weights be both positive ($w_{ij}>0$) and symmetric ($w_{ij} = w_{ji}$). The elements a_{ij} of the adjacency matrix A of the graph are defined in a standard way as

$$a_{ij} = \begin{cases} w_{ij} \text{ if node } i \text{ is connected to node } j \\ 0 \text{ otherwise} \end{cases}$$

where A is symmetric.

Because of the way the graph is defined, user agents who have the same keywords, and therefore have similar interest, will have a comparatively large number of short paths connecting them. On the contrary, for user agents with different interests, we can expect that there will be fewer paths connecting them and that these paths will be longer.

4.2. A random-walk model on the graph

The Markov chain describing the sequence of nodes visited by a random walker is called a random walk. We associate a state of the Markov chain to every node; we also define a random variable, $s(t)$, representing the state of the Markov model at time step t. If the random walker is in state i at time t, then $s(t) = i$.

We define a random walk with the following single-step transition probabilities

$$P(s(t+1)=j \mid s(t)=i)=\frac{a_{ij}}{a_{i.}}=p_{ij} \quad \text{where } a_{i.}=\sum_{j=1}^{n} a_{ij}$$

In other words, to any state or node $i = s(t)$, we associate a probability of jumping to an adjacent node $j = s(t+1)$, which is proportional to the weight w_{ij} of the edge connecting i and j. The transition probabilities only depend on the current state and not on the past ones (first-order Markov chain). Since the graph is totally connected, the Markov chain is irreducible, that is, every state can be reached from any other state. If this is not the case, the Markov chain can be decomposed into closed sets of states which are completely independent (there is no communication between them), each closed set being irreducible.

If we denote the probability of being in state i at time t by $x_i(t) = P(s(t) = i)$ and define P as the transition matrix with entries $p_{ij} = P(s(t+1) = j \mid s(t) = i)$, the evolution of the Markov chain is characterized by

$$\begin{cases} x_i(0)=x_i^0 \\ x_i(t+1)=P(s(t+1)=i \mid s(t)=j)x_j(t)=\sum_{j=1}^{n} p_{ij}x_j(t) \end{cases}$$

Or, in matrix form, $\begin{cases} x(0)=x^0 \\ x(t+1)=P^T x(t) \end{cases}$

where T is the matrix transpose. This provides the state probability distribution $x(t) = [x_1(t), x_2(t), ..., x_n(t)]^T$ at time t once the initial probability density x^0 is known. It is well-know (see [15]) that such a Markov chain of random walk on a graph is time-reversible ($\pi i P i j = \pi j P j i$) with stationary probabilities given by

$$\pi_i=\frac{\sum_{j=1}^{n} a_{ij}}{\sum_{i,j=1}^{n} a_{ij}}=\frac{a_{i.}}{a_{..}}$$

This value is the probability of finding the Markov chain in state $s = i$ in the long-run behaviour.

This has important implications. For instance, it is known that all the eigenvalues of the transition matrix of a reversible Markov chain are real and distinct (see for instance [16])

For more details on Markov chains, the reader is invited to consult standard textbooks on the subject (e.g., [17], [18]).

4.3. Association of a weight to the keywords and the edges

In our recommendation system, the weight of a keyword in a profile represents the importance of this word for the user compared to all the other keywords.

We will note k_{ij} the weight of the keyword j in the user profile i. This weight must take into account the number of request made by the user on this keyword, as well as the number of visited links. So, the computation of the weight for a keyword j in the user profile i is given by

$$k_{ij} = f_{ij} + l_{ij}$$

where f_{ij} represents the number of request done by the user i on the keyword j, and l_{ik} corresponds to the number of links for the keyword j contained in the profile of the user i.

Now that we have defined the meaning of weight for a keyword, we can specify the concept of weight for an edge in our recommendation system.

The weight of an edge between two users (i.e., their personal agents) represents the similarities degree which expresses the resemblance of their profiles. The profiles of two agents are strongly similar, if the users related to these agents are interested by the same keywords and accept the same links for theses keywords.

We will note c_{ij} the similarities between the profiles of user i and user j, and the computation of this weight is characterized by

$$c_{ij} = sim(Pf_i, Pf_j) = \frac{\sum_{k=1}^{r} k_{il*} k_{jl}}{\sqrt{\sum_{l=1}^{r_i} k_{il}^2} * \sqrt{\sum_{l=1}^{r_j} k_{jl}^2}}$$

where Pf_i represents the list of the keywords contained in the user profile i and r corresponds to the number of keywords that Pf_i and Pf_j have in common.

4.4. Probability of absorption and average first-passage time/cost

In this section, we review three basic quantities that can be computed from the definition of the Markov chain, that is, from its transition probability matrix: the probability of absorption, the average first-passage time, and the average commute time. We also introduce the average first-passage cost which generalizes the average first-passage time. Relationships allowing to compute these quantities are derived in a heuristic way (see, e.g., [17] or [18] for a more formal treatment).

4.4.1. The probability of absorption

The **probability of absorption** $u(k|i)$ is the probability that a random walker starting from some initial state i enters for the first time state $k \in S^a$ (S^a is a subset of states) before reaching any other state belonging to S^a. The states of the set S^a are "absorbing states" in the sense that, once the random walker reaches one of them, it stops walking. Thus, once an absorbing state has been reached, the probability of staying in it is 1. A recurrence relation allowing to compute the probability of absorption can be obtained by elementary probability theory (a proof is provided in [19]):

$$\begin{cases} u(k \mid i) = p_{ik} + \sum_{j=1, j \neq k}^{n} p_{ij} u(k \mid j), \text{ for } i \notin S^a \text{ and } k \in S^a \\ u(k \mid k) = 1, \text{ for } k \in S^a \\ u(k \mid i) = 0, \text{ for } i, k \in S^a \text{ and } i \neq k \end{cases}$$

This system of linear equations can be solved by iterating the relations (the relaxation method, see [13]). The probability of absorption can also be obtained by algorithms that were developed in the Markov-chain community (see for instance [17]), or by using the pseudoinverse of the Laplacian matrix of the graph (see [19]).

4.4.2. The average first-passage time and average first-passage cost

A similar relationship can be established for the **average first-passage time** $m(k|i)$, which is defined as the average number of steps that a random walker, starting in state $i \neq k$, will take to enter state k for the first time [18]. More precisely, we define the minimum time until hitting state k as $T_{ik} = \min (t \geq 0 \mid s(t) = k \text{ and } s(0) = i)$ for one realization of the stochastic process. The average first-passage time is the expectation of this quantity, when starting from state i: $m(k|i)=E[T_{ik}|s(0) = i]$.

In a similar way, we define the **average first-passage cost**, $o(k|i)$, which is the average cost incurred by the random walker starting from state i to reach state k. The cost of each transition is given by $c(j|i)$ (a cost matrix) for any states i, j. Notice that $m(k|i)$ can be obtained as a special case where $c(j|i) = 1$ for all i, j.

In [19] we derive recurrence relations for computing $m(k|i)$ and $o(k|i)$ by first-step analysis:

$$\begin{cases} o(k \mid k) = 0 \\ o(k \mid i) = \sum_{j=1}^{n} p_{ij} c(j \mid i) + \sum_{j=1, j \neq k}^{n} p_{ij} o(k \mid j), \text{ for } i \neq k \end{cases}$$

For $m(k|i)$, we obtain

$$\begin{cases} m(k \mid k) = 0 \\ m(k \mid i) = 1 + \sum_{j=1, j \neq k}^{n} p_{ij} m(k \mid j), \text{ for } i \neq k \end{cases}$$

These equations can be used in order to iteratively compute the first-passage times [18] or first-passage costs. The meaning of these formulae is quite obvious: in order to go from state i to state k, one has to go to any adjacent state j and proceed from there. Once more, these quantities can be obtained by algorithms developed in the Markov-chain community (see for instance [17]) or by using the pseudoinverse of the Laplacian matrix of the graph, as shown in [19].

5. External search

This search provides external links to the community by using an external search engine. Its role is primordial because it supplies the community experience with new knowledge. This research requires an external entity which launches a search using only the user keywords. In our case, this external entity is GOSIS [22], an aGent-Oriented Search Information System.

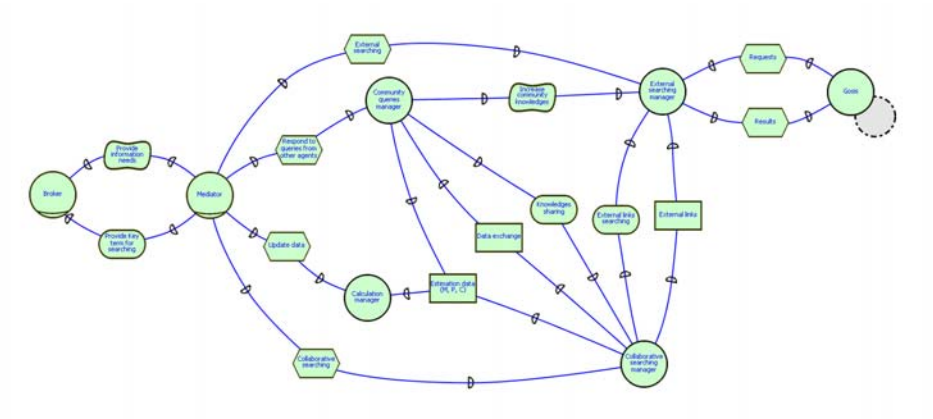

Figure 5-1 : The complete architecture of personal agent

GOSIS is a typical search engine supporting the creation of information sources that facilitate the integration of information coming from different heterogeneous sources. The sources may be conventional databases or other types of information, such as collections of Web pages. Figure 5-2 shows the GOSIS architecture in joint-venture in which the Broker plays the role of the joint-venture's management public interface, the Mediator plays the role of the joint-venture's management private interface and the associated actors are Matchmaker, Wrapper, Monitor and Multi-criteria Analyzer.

The Broker interacts with users and provides them ain interface for searching information. The Mediator coordinates the tasks and controls the resource's share between the associated actors. Each associate can control himself on a local dimension and interact directly with others to exchange resources, data and knowledge.

When a user wishes to send a request, it contacts the broker agent, which serves as an intermediary to select one or many mediator(s) that can satisfy the user information needs. Then, the selected mediator(s) decomposes the user's query into one or more sub-queries to the sources, synthesizes the source answers and returns the answers to the broker.

If the mediator identifies repetitive user information needs, the information that may be of interest is extracted from each source, merged with relevant

information from other sources, and stored as knowledge by the mediator. Each piece of information stored constitutes a materialized view that the mediator will have to maintain up-to-date.

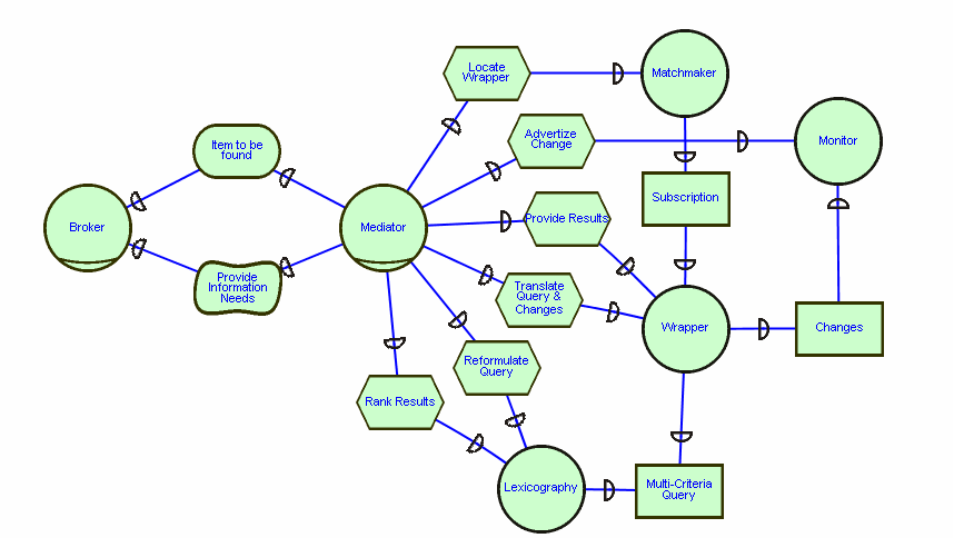

Figure 5-2 : The GOSIS architecture in joint-venture

Connected to each information source is a wrapper and monitor agent. The wrapper is responsible for translating the sub-query issued by the mediator in the native format of the source and translating the source response in the data model used by the mediator.

The monitor is responsible for detecting changes of interest (e.g, change which affects a materialized view) in the information source and reporting them to the mediator. Changes are then translated by the wrapper and sent to the mediator.

It may also be necessary for the mediator to obtain information concerning the localization of a source and its connected wrapper that are able to provide current or future relevant information. This kind of information is provided by the matchmaker agent, which then lets the mediator interact directly with the correspondent wrapper. The matchmaker plays the role of a “yellow-page” agent. Each wrapper advertises its abilities by subscribing to the yellow pages. The wrapper that no longer wishes to be advertised can request to be inscribed.

Finally, the multi-criteria analyzer can reformulate a sub-query (sent by a mediator to a wrapper) through a set of criteria in order to express the user preferences in a more detailed way, and refine the possible domain of results.

We have developed WrapperSQLServer, WrapperText and WrapperGoogle for different types of information sources: SQL Server (structured information), text file (semi-structured information) and Internet (non-structured information,

with the help of the Google search engine). By using this search engine, the external links are diversified and the community experience increases with these links. The recommendation system can collect the information from different and heterogeneous sources. That leads to a significant increase in the efficiency of the system.

6. Conclusion

We described an agent-based recommendation system dealing with the extraction of implicit knowledge from user behavior during web search. The knowledge produced from observations is used in order to suggest links or agents to group of people and to their personal agents. The main idea is that we do not express this knowledge in explicit form but use it for improving quality of further search sessions, including searches performed by new users. Personal agents produce results by asking another personal agent about links. Each agent has the learning capabilities that help to produce results even without interaction. With interaction, when the user performs a search already done by one of the community members, he/she does not do it \from scratch" but exploits his/her and the others' experience. This feature increases the search quality.

Our system is a solution to the problem of finding necessary information from multiple heterogeneous sources. One of the main advantages of our approach is represented by the use of both search engine results and suggestions produced by community members. The multi-agent system mimics natural user behavior of asking someone who probably knows the answer. Finally, the process of producing suggestions is completely hidden from the user and therefore does not force him/her to perform additional actions.

There are possibilities of user profile improvements. Although for the present time it contains information only about acceptance and rejection of links obtained from an external search engine, it is possible to have rules for acceptance of links from the other agents. Finally, one of the further directions is to analyze the social relations and interactions between the users.

References

[1] Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a FIPA-compliant agent framework. In: Software-Practice and Experience, Vol. 31 (2) (2001) 103-128

[2] Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual
Search Engine. In: Computer Networks and ISDN Systems 30 (1998) 107-117

[3] Chau, M., Zeng, D., Chen, H., Huang, M., Hendriawan, D.: Design and Evaluation of a Multi-agent CollaborativeWeb Mining System. In: Decision Support Systems 35 (2003) 167-183

[4] Chen, L., Sycara, K.: WebMate: A Personal Agent for BrowsingandSearch. In: Proceedings of the2ndInternational Conference on Autonomous Agents, ACM Press, New York, NY (1998) 132-139

[5] Degemmis, M., Lops, P., Semeraro, G., Costabile, M.F., Lichelli, O., Guida, S.P.: A Hybrid Collaborative Recommender System Based on User Pro_les. In: Proceedings of the Sixth International Conference on Enterprise Information Systems, Vol. 4. Porto (2004) 162-169

[6] Gori, M., Witten, I.: The bubble of Web visibility. In: Communications of the ACM (2004 In Press)

[7] iProspect Search Engine User Attitudes Survey.:
http://www.iprospect.com/premiumPDFs/iProspectSurveyComplete.pdf

[8] Lieberman, H.: Letizia: An Agent That Assists Web Browsing. In: International Joint Conference of Artificial Intelligent, Montreal (1995)

[9] Blanzieri, E., Giorgini, P., Massa, P., Recla, S.: Implicit Culture for Multi-agent Interaction Support. Proceedings of Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Trento – Italy (2001). Available at: http://dit.unitn.it/~ implicit/

[10] Menczer, F.: Complementing Search Engines With OnlineWeb Mining Agents. In: Decision Support Systems 35 (2002) 195-212

[11] Somlo, G.,Howe, A.: UsingWeb Helper Agent Pro_les in Query Generation. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, ACMPress, Melbourn (2003) 812-818

[12] Turner, R., Turner, E., Wagner, T., Wheeler, T., Ogle, N.: Using Explicit,APriori Contextual Knowledge in an Intelligent Web Search Agent. In: Lecture Notes in Artificial Intelligence 2116 (2001) 343-352

[13] Wei, Y., Moreau, L., Jennings, N.: Recommender Systems: A Market-Based Design. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, ACMPress, Melbourn (2003) 600-607

[14] Yu, Bin, Singh, M.: An Agent-Based Approach to Knowledge Management. In: Proceedings of Eleventh International Conference on Information and Knowledge Management (CIKM02), SAIC Headquarters, McLean, Virginia, USA (2002)

[15] S. Ross. Stochastic Processes, 2nd Ed. Wiley, 1996.

[16] P. Bremaud. Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues. Springer-Verlag, 1999.

[17] Wei, Y., Moreau, L., Jennings, N.: Recommender Systems: A Market-Based Design. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, ACMPress, Melbourn (2003) 600-607

[18] J. Norris. Markov Chains. Cambridge University Press, 1997.

[19] Fouss F., Pirotte A., Renders J.-M, Saerens M.: A novel way of computing dissimilarities between nodes of a graph, with application to collaborative filtering and subspace projection of the graph nodes, 2004

[20] M. Kolp, T. Tung Do and S. Faulkner. Introspecting Agent-Oriented Design Patterns. To appear in S. K. Chang (Ed.) Advances in Software Engineering and Knowledge Engineering, World Scientific, 2004.

[21] M. Kolp, S. Faulkner and T. Tung Do. Analysis Styles for Requirements Enginnering: an Organizational Perspective. To appear in S. K. Chang (Ed.) Advances in Software Engineering and Knowledge Engineering, World Scientific, 2004.

[22] S. Faulkner, M. Kolp T. Nguyen, A. Coyette A Multi-Agent Perspective on Data Integration Architectural Design. Knowledge-Based Information and Engineering Systems (KES), 2004