

An Intelligent Peer-to-Peer Multi-Agent System for Collaborative Management of Bibliographic Databases

Hager Karoui, Rushed Kanawati, and Laure Petrucci

LIPN, CNRS UMR 7030, Université Paris XIII
99, avenue Jean-Baptiste Clément
F-93430 Villetaneuse, FRANCE
{hager.karoui, rushed.kanawati, laure.petrucci}@lipn.univ-paris13.fr

Abstract. This paper describes the design of a peer-to-peer system for collaborative management of distributed bibliographical databases. The goal of this system is twofold: firstly, it aims at providing help for users to manage their local bibliographical databases. Secondly, it offers the possibility to exchange bibliographical data among like-minded user groups in an implicit and intelligent manner. Each user is assisted by a personal agent that provides help such as: filling in bibliographical records, verifying the correctness of information entered and more importantly, recommendation of relevant bibliographical references. To do this, the personal agent needs to collaborate with its peers in order to get relevant recommendations. Each agent applies a case-based reasoning approach in order to provide peers with requested recommendations. The paper focuses mainly on describing the recommendation computation approach.

Keywords. Peer-to-Peer systems, Recommender systems, Case-based Reasoning, Bibliographies data sharing.

1 Introduction

Maintaining an up-to-date annotated bibliographical database is a central activity of research teams. However the multiplication of document sources (e.g. workshops, conferences, journals, etc.) as well as the on-line availability of most documents have contributed in making the task more complex and more time-consuming. Actually, in addition to the classical information overload problem, researchers have now direct access to papers that are seldom coupled with the complete bibliographical data. It is frequent now to start a new search session in order to find the exact reference of an interesting paper that was found previously. Luckily, researchers usually work in like-minded teams. It is highly possible that information obtained or known to one or more users can be useful to another. In addition, colleagues may have useful hints about the quality of papers and what to read if interested in a given topic. It is obvious that

sharing bibliographical knowledge could not only enrich each member knowledge but also reduce time and effort needed to manage personal databases. However, lessons learned for groupware design studies have shown that for collaborative tools to be successful, they should meet a number of requirements [4,5]. Mainly, the effort required for using a collaborative tool should be as much as the same of that needed for using an individual tool that provides the same service. In addition, users will likely be willing to use a collaborative tool if they are well rewarded. In our example, a user that spends time in feeding a shared bibliographical base without getting any or few valuable recommendations will abandon the search.

Based on these remarks, we propose a new peer-to-peer multi-agent system for collaborative management of distributed bibliographical databases. Each user is assisted by a personal assistant software agent. An assistant agent observes the user interactions with a local personal bibliographical database. It tracks the user current hot topics and finds out information missed in the local database (e.g. the location of a cited conference, the number of pages of a given paper, etc.). Agents communicate with each other in order to find missing information but also to recommend their associated users with references, related to their hot topics, that have been found relevant by other colleagues. Agents can verify also the correctness of local references by comparing these with peer's records. The only additional effort required by the users is to accept or to reject provided recommendations. Since recommendations are made in the context of each user's hot topics, we expect that users will be willing to examine these recommendations.

A more detailed description of the system architecture and services is given in section 2. The paper's main focus is the recommendation computation technique. This is done by applying a case based reasoning (CBR) approach. The CBR methodology has been used successfully by a number of recommender systems [3]. It allows for learning in an incremental manner by dynamically associating similar users. The applied reasoning cycle is detailed in section 3. Related work is discussed in section 4 and finally we conclude and give directions for future work in section 5.

2 System Description

Figure 1 illustrates the overall system architecture where each user maintains a personal bibliographical database locally. The user manages the local database using a management module that provides the classical management functions such as adding, deleting, editing and searching. In addition, it provides functionalities for selecting a list of references to add to an ongoing work (e.g. building a report's bibliography) and exporting lists into different formats (e.g. BibTeX, html, pdf, etc.). In order to ease the exportation/transformation operations the references are stored in an XML database. Each reference r is described by a record that contains the following information:

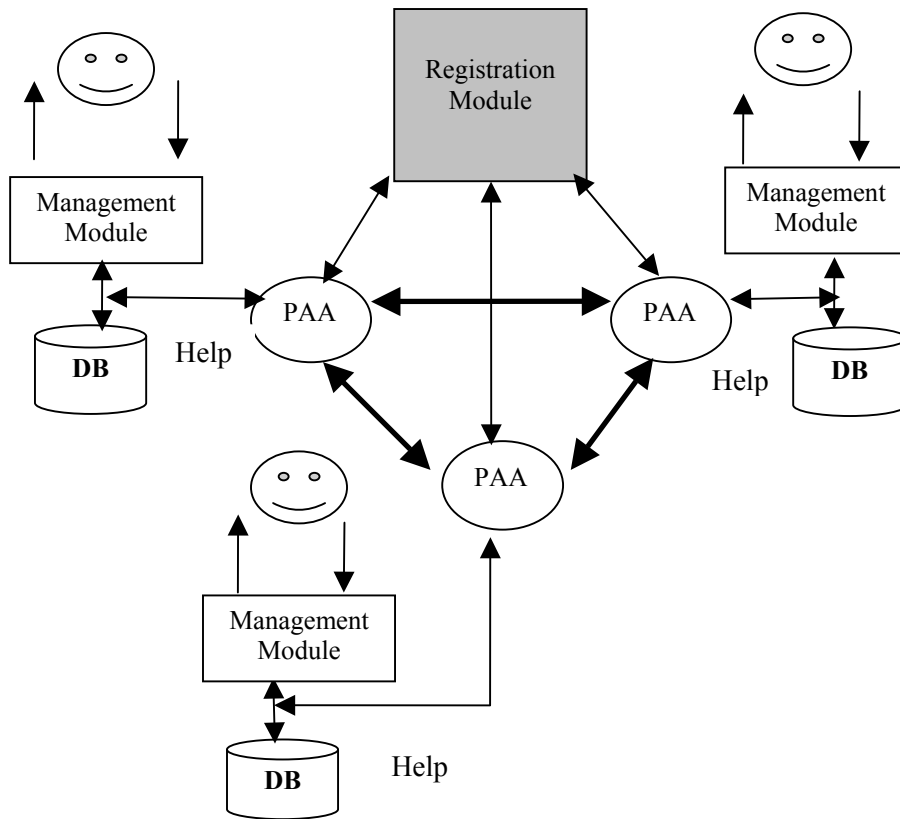


Fig.1. The system architecture

- Bibliographical data (denoted *r.biblio*): the classical data composing a bibliographical reference such as the type (e.g. Article, In Proceedings, Report, etc.), list of authors, title, etc.
- Keywords (denoted *r.keywords*): a list of keywords describing the reference.
- Topics (denoted *r.topics*): a list of topics the reference is related to. The same topic map is used by all users. In this work, we use a tree topic hierarchy. It is reasonable to say that the same research team uses the same topic trees to index the bibliographical references (e.g. using the ACM topic hierarchy in computer science research teams). However, we stress that the same hierarchy will be used differently by different users. Hence, the same reference can be related to different topics by different users. For example one may index all CBR-related papers to the same topic, e.g. CBR, while another user may index the same papers differently: some related to memory

organization in CBR systems and others for CBR case maintenance. A third may index the same references as all related to lazy learning.

- Evaluation (denoted `r.eval`): an evaluation made by each user indicating the quality of the reference from the user's point of view. Evaluation is made on a 4-degree scale starting from very interesting going down to rubbish (if any)!!
- Annotation (denoted `r.annotation`): a free-text field.

Each user is assisted by a personal agent that helps in managing her/his own database. Different services are provided by the local assistant:

- Editing references: when the user is editing a reference, the local assistant can help her/him with filling some data fields (e.g. when user types an acronym of conference name, the assistant provides her/him with the complete name and can fill the other related information such as the conference date, location, etc.)
- References correctness: the correctness verification service is started when references are added or edited in the bibliographical database. After the data fields (e.g. title of conference, name of author, ...) concerning a reference are filled and validated, the system checks the correctness of the input by starting the corresponding agent. A first check of the data correctness is done by matching first the local database of the user. If the data is correctly written then the task is confirmed. Otherwise, if errors are found, the agent suggests a correction, or, if some fields are empty the agent suggests relevant data if available. If the data is not found in the local base, the agent collaborates with its counterpart agents in order to find them.
- Recommendation: this service aims at sharing bibliographic knowledge between the users.

The goal is to take advantage of past experiences of a single user or even a group of users for recommending more relevant references. A CBR approach is used to compute the different recommendations in a cooperative way, i.e. the different assistant agents must collaborate with each other to obtain various and relevant recommendations from several agents. We want the local assistant to suggest various and interesting recommendations to its user according to her/his current activity. The user can choose to either accept or refuse the recommendation proposed.

3 Recommendation Computation

3.1 Informal description

In order to provide users with relevant recommendations, each assistant agent applies the following computation cycle: first it computes the list of the most hot topics that interest the user. For each hot topic the agent sends a recommendation request to a set of peer agents that are likely to have references related to the topic. A recommendation request message contains a topic identifier and a list of keywords that describe the set of references saved under that topic in the local database. The agent may receive from each contacted agent, two lists of references: recommended references and unrecommended ones. The later list is formed of references that match

the request but are badly evaluated by the associated user. The assistant agent merges and filters the received results in order to remove duplicated references as well as references that are already stored in the local database (hence known to the user). The highly ranked references are then recommended to the user. The latter can accept to add all or some of these references to the indicated topic or to other topics. Users can also reject some or all provided recommendations. When receiving a recommendation request an agent searches in the local database for references that match the received query. The search process starts from the topic indicated in the request. Recall that all users share the same topic hierarchy. When there is an insufficient number of references the agent continues the search in sub-topics following a depth-first search strategy. If not enough references are found the agent continues the search in super-topics. The rationale is to prefer references related to sub-topics that are supposed to be more focused to those related to super-topics that are supposed to be more general. The search process ends when enough relevant references are found or if the similarity between the request topic and the searched topic falls below a given threshold. Relevancy of a reference is computed by a similarity measurement that takes into account the similarity between topics (requested and searched one) as well as the match between the reference keyword description and the request keyword list.

3.2 Hot topics computation

As stated before all users share the same topic hierarchy that has a tree structure. All topics are not equally interesting for each user. Moreover, for each user the set of interesting topics changes over time. It is crucial for any recommender system to provide recommendations in the area that interests the user most. Recommendations providing should not be intrusive. Furthermore, if users are really interested in the recommendation area they will be more willing to evaluate these recommendations. In order to compute a user's hot topics list we apply a simple algorithm that measures the temperature of each topic. Topics that have a temperature above a given threshold σ will be labeled as hot topics. Initially, all topics have a zero temperature. Each action executed by the user involving a topic t will increase the temperature of that topic by a specified amount. Typical actions that modify a topic's temperature are: adding, editing or searching for a reference related to the topic. Different actions add different values to the current topic temperature. A cooling function is also applied in order to decrease the temperature of topics not used by the user. As the topic hierarchy can be used differently by different users, we need to determine the most specific topics (e.g. deepest topics) the user's activity is centered on. To do so, a temperature propagation function is applied. Starting from the leafs of the the topic tree, each topic propagates its current temperature to the parent topic. Topics with a temperature above a system value σ_r will cease to propagate their temperature and will be added to the list of hot topics. The n most hot topics that have been added to the hot topics lists after visiting the tree in a bottom-up way will be returned. The heuristics is to return the most specific topics which concentrate a given level of the user's focus.

3.3 Committee formation

An important issue in any peer-to-peer system concerns the peer committee formation. The idea is to provide each agent with the ability to select a subset of available peers that are likely to provide the most relevant results (in our case recommendations). The goal is not only to enhance the overall system performances by reducing both the network and the agents load. The aim is also to enhance the quality of recommendations provided by avoiding noisy results. Different approaches are proposed in the literature. Some are based on the notion of agent reputation [1]. Others propose to apply automatic learning techniques in order to enable each agent to determine if it needs to increase the committee of peers and if it is the case which peer agent to invite [9]. While we totally agree about the importance of that issue, we have decided to employ a naive approach consisting of broadcasting recommendation requests to all available peers as an initial approach. We suppose that all agents could be assigned the same trust degree.

3.4 Delivering recommendations

A recommendation request message is composed of a triple: $R = \langle A, T, L \rangle$ where A is the sender agent identifier, T is a target topic and L is a list of keywords that is computed from the set of keywords lists describing references related, directly or indirectly to the topic T . A reference is indirectly related to a topic T if it is related to a topic T' more specific than T .

When receiving a request, an agent starts to search in its local database for references that match the couple (T, L) . Informally, the keyword list contained in a request will be treated as a query, the designated target topic T indicates the starting point of the document search in the local database. The agent will retrieve from the local database references that match the received query. Reference/query matching is evaluated by a simple similarity function $\text{Sim}_{\text{Ref}}(R, r)$ that measures the similarity between a request R and a reference r . A reference r matches a request R if their similarity is above a given specified threshold σ_r . The similarity function is a weighted aggregation of two basic similarity functions: topic similarity ($\text{Sim}_{\text{Topics}}$) and keywords similarity ($\text{Sim}_{\text{Keywords}}$). Formally we we have :

$$\text{Sim}_{\text{Ref}}(R, r) = \alpha \text{Sim}_{\text{Topics}}(R.T, r.\text{topics}) + \beta \text{Sim}_{\text{Keywords}}(R.L, r.\text{keywords}) .$$

where α and β are the basic similarities weights. Obviously, we have $\alpha + \beta = 1$. The keyword similarity function used is a simple function measuring the number of common words between two lists. Formally:

$$\text{Sim}_{\text{Keywords}}(A, B) = (A \cap B) / (A \cup B) .$$

The topics similarity measure uses the topics underlying hierarchical structure. The applied heuristic is the following: The similarity between two topics depends on the length of the path that links the two topics and on the depth of the topics in the

hierarchy. Recall that in a tree there exists only one path between any two nodes. Moreover, a match with specific nodes closer to leaf nodes results in a higher similarity than nodes matching at higher levels of the tree. Formally we have:

$$\text{Sim}_{\text{Topics}}(T_1, T_2) = 1 - (\text{path}(T_1, \text{MSCA}(T_1, T_2)) + \text{path}(T_2, \text{MSCA}(T_1, T_2))) / (\text{path}(T_1, \text{root}) + \text{path}(T_2, \text{root})) .$$

where $\text{path}(a,b)$ returns the path length between nodes a and b , root is the topic's tree root and $\text{MSCA}(a, b)$ returns the most specific common ancestor of nodes a and b in the topic tree.

Figure 2 shows a simple example of topic tree where the root is *Computing Methodologies*. We consider here four topic levels. Each topic has a list of associated references.

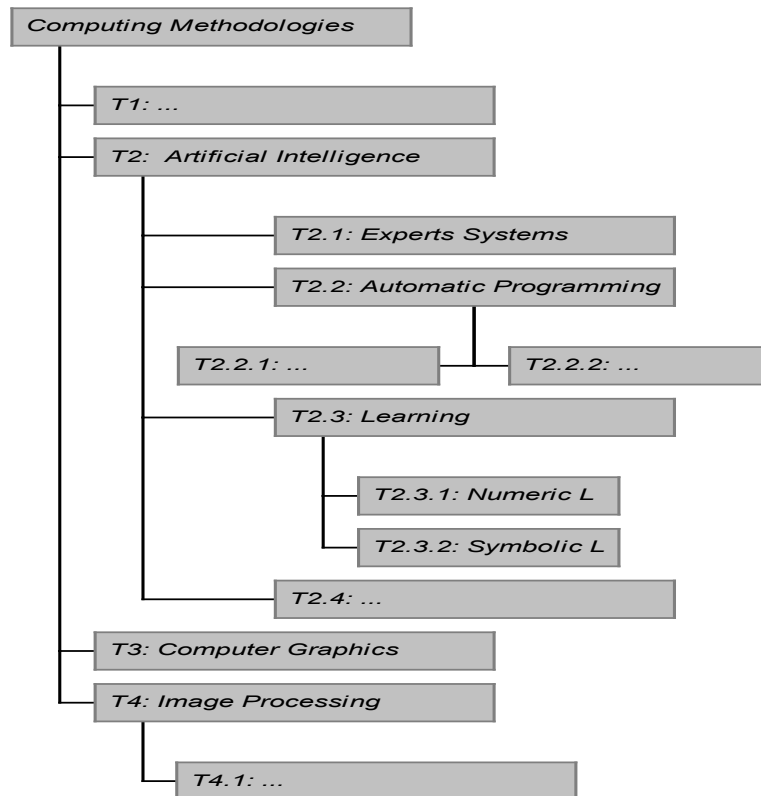


Fig.2. Example of topic tree

Based on the topic tree of Figure 2, we give some examples of topic similarity computation:

$$\begin{aligned} \text{Sim}_{\text{Topics}}(T_{2.3.1}, T_{2.3.2}) &= 1 - ((1+1) / (3+3)) = 2/3 \\ \text{Sim}_{\text{Topics}}(T_{2.3.1}, T_{2.2}) &= 1 - ((2+1) / (3+2)) = 2/5 \\ \text{Sim}_{\text{Topics}}(T_{2.2}, T_{2.3}) &= 1 - ((1+1) / (2+2)) = 1/2 \end{aligned}$$

The topic similarity values are computed by applying the $\text{Sim}_{\text{Topics}}$ function. We note that the value of the $\text{Sim}_{\text{Topics}}(T_{2.3.1}, T_{2.3.2})$ is higher than the value of $\text{Sim}_{\text{Topics}}(T_{2.2}, T_{2.3})$ because they are closer to leaf nodes. In addition, the topics $(T_{2.3.1}, T_{2.3.2})$ are more similar than $(T_{2.3.1}, T_{2.2})$.

Using these similarity functions, the local agent will try to return the m most relevant references that match the received request. Starting from the target topic $(R.T)$ the agent will search for related references that are similar above a threshold σ_r . If not enough references are found, it examines references related to more specific topics, then it moves to more general topics. The retrieval process ends when m relevant references are found or when no more topics are left.

Relevant references that are positively evaluated by the local user form the list of recommended references while those badly evaluated form the list of unrecommended references. Each returned reference is associated with a score representing its similarity with the initial request. The agent that has sent the recommendation request will receive a couple of recommended and unrecommended references from each agent. It merges and filters the received lists, it eliminates firstly duplicated references and those which are already known to the user (those actually stored in the local database). Then it applies a sorting method in order to rank recommended and unrecommended references regardless of their similarity values. In a next step another value presenting the importance of the sender agent will be taken into account. The k highly ranked recommended references will be proposed to the user.

4 Related Work

One interesting work directly related to our approach is the Bibster system [2]. The main goal of Bibster is to allow a group of people to search for bibliographical references in each other personal database. A peer-to-peer architecture is used. However, only exact searching is supported. Our system represents an extension to the Bibster system, where similarity-based searching is used. Moreover, the search is made by software agents instead of being initiated by the users themselves.

Collaborative bookmarking systems address a similar problem [7]. However in peer-to-peer collaborative bookmarking systems [6] we lack a unified hierarchy of topics making the matching evaluation harder to compute. Another similar work is the I-SPY information searching engine [12]. The goal of the I-SPY system is to allow a group of like-minded people to share their search results in an implicit way. The

system is built in a centralized manner where a hit matrix records for each submitted query the documents selected by the user. When a new query is submitted, results that have been selected by other users in response to similar past queries are provided by the system. In our system the set of topics can be viewed as a set of pre-specified queries. The overlap between user queries is much more likely to happen than in the case of open vocabulary queries.

In [8], McGinty and Smyth describe a collaborative case base reasoning CCBR architecture, which allows problem solving experiences to be shared among multiple agents by *trading* cases. This approach was applied in personalised route planning and it promises a solution by allowing a given user agent to borrow cases from similar agents that are familiar with the target territory. There is a tradeoff between agent similarity and their case base coverage. That is, a remote agent is useful to a target agent if it has a different coverage, but to be considered similar, they must share a set of common problems. Plaza & all investigate in [10] possible modes of cooperation among homogeneous agents with learning capabilities. They present two modes of cooperation among agents: Distributed Case based Reasoning (DistCBR) and Collective Case based Reasoning (ColCBR). In the DistCBR cooperation, an originating agent delegates authority to another peer agent to solve the problem. In contrast, ColCBR maintains the authority of the originating agent, since it decides which CBR method to apply and merely uses the experience accumulated by other peer agents. They prove that the result of cooperation is always better than no cooperation at all. However, these protocols are domain dependent and are the result of a knowledge modelling process. In [11], Plaza and Ontanon present several collaboration strategies for agents that learn using CBR. Agents use a market mechanism (bartering) to improve the performances both of individual and the whole multi-agent system. Two policies are presented: Committee policy and Bounded Counsel policy. In the first collaboration policy, the member agents of a MAC system are viewed as a committee. An agent that has to solve a problem sends it to all the other agents of the committee. The final solution is the class with the maximum number of votes. The next policy is the Bounded Counsel where an agent tries to solve a problem by himself and if it fails to find a “good” solution, then it can ask counsel to other agents in the MAC system. They conclude that the Committee policy is better than the Isolated and Bounded Counsel, however, this precision has a higher cost since a problem is solved by every agent. Because of the bias of the examples of an agent which decreases the accuracy of the system, they propose the collaboration strategy based on bartering cases to improve the performance of the MAC system by diminishing their individual case base bias. The mechanism of case bartering is based on bartering agreement where the result of the interchange diminishes the agents’s individual case bases bias.

5 Conclusion and Future Work

This paper describes work in progress that aims at providing a group of like-minded people with an intelligent and an implicit way to share their bibliographical knowledge. Currently, we are working on implementing the first version of this system. The CBR approach described in this paper is limited to the use of similarity

guided searching. However we are working on defining a deeper CBR approach that would enable agents to speed up their response time by exploiting similarities between current and past recommendation requests. Another important problem to handle is the committee formation problem: how to learn to form the optimal committee for each recommendation request? This is still an open question. Another research direction that we will consider in the future is the application of collaborative CBR schemes in order to enhance the individual recommendation quality of each assistant agent.

References

1. M. Ammar, M. Gupta and P. Judge. A reputation system for peer-to-peer networks. In 13 international workshop of networks and operating system support for digital and video, Montreal, 2003
2. J. Broekstra, M. Ehrig, P. Haase, F. Harmelen, M. Menken, P. Mika, B. Schnizler, and R. Siebes. Bibster -a semantics-based bibliographic peer-to-peer system. In Proceedings of SemPGRID'04, 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing, pages 3-22, New York, USA, may 2004
3. Robin Burcke. Hybrid recommender systems with case-based reasoning. In Advances in Case-based Reasoning, 7th European Conference, ECCBR 2004, LANI 3155, pages 91-105, 2004
4. J. Grundin. Why cscw applications fail: Problems in the design and evaluation of organisational interfaces. In Proceedings of the first ACM Conference on Computer Supported Cooperative Work, 1998
5. R. Kanawati. Groupware: Architectural and control issues. PhD thesis, Institut National Polytechnique de Grenoble, 1997
6. M. Malek and R. Kanawati. Cowing: A collaborative bookmark management system. In Proceedings of CIA'02: International workshop on cooperative information agents, LANI 2182, pages 34-39, 2001
7. M. Malek and R. Kanawati. Informing the design of shared bookmark systems. In Proceedings of RIAO'2000: Content-based Multimedia Information Access, 28 April 2000, Paris, 2000
8. Loraine McGinty and Barry Smyth. Collaborative case-based reasoning: Applications in personalised route planing. In ICCBR, pages 362-376, 2001
9. Santiago Ontanon and Enric Plaza. Learning to form dynamic committees. In Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pages 504-511, Melbourne, Australia, 2003. ACM Press, NEW YORK, USA
10. Enric Plaza, Josep Lius Arcos, and Francisco Martin. Cooperation modes among case-based reasoning agents, 1996
11. Enric Plaza and Santiago Ontanon. Cooperative Multiagent Learning. Springer, London, March 2003
12. B. Smyth and E. Balfe. Case-based collaborative web search. In 7th European Conference on Advances in Case-based Reasoning, Madrid, pages 489-503, 2004