A Multi-agent System for Collaborative Bookmarking

Rushed Kanawati[1], Maria Malek[2]

[1]LIPN-CNRS UMR Q 7030 , 99 Av. J. B. Clément, F-93430 Villetaneuse
rushed.kanawati@lipn.univ-paris13.fr
[2]LAPI-EISTI, Av. du Parc, F-95011 Cergy
maria.malek@eisti.fr

Abstract. In this paper we describe a multi-agent system, called *COWING* that aims at enabling an organised group of people to share their results of information searching on the World Wide Web. Users normally save relevant sites they found in their private collections of bookmarks. The goal of the proposed system is to enable users to share their bookmarks, in an *implicit*, *secure* and *effective* way. By implicit we mean that users are not required to spend extra effort in order to use the system. Secure sharing refers to the capacity of each user to control who knows what about her/his own bookmark collections. Finally effectiveness is ensured by recommending users with relevant bookmarks that are computed by applying a distributed collaborative filtering algorithm. Recommended bookmarks are automatically inserted in the most appropriate bookmark folder in the user's local collection. When accessing a folder the user can evaluate (accept or reject) provided recommendations. Our system is a multi-agent system where each user is associated to an assistant agent, called a *WING* agent. A *WING* agent observes the user's behaviour when managing her/his own bookmark collection. It learns how the user classifies her/his bookmarks. A hybrid neural/case-based reasoning supervised classifier is used for this purpose. *WING* agents exchange bookmarks according to a defined collaboration protocol that protects associated users privacy.

Keywords: Collaborative Information Agents, Bookmark, Hybrid Neural/CBR classification.

1. Introduction

The most intelligent *agents* that browse and index sites on the World Wide Web (the Web hereafter) still to be *human*s. Consequently, an effective way to locate relevant information on the Web is to locate people who are likely to know where to find the searched information [25]. Users usually store addresses of relevant Web sites in a bookmark directory. Almost all Web browsers available today provide users with some *bookmaking* facility. Typically a bookmark is a record that holds the information like: the site address (i.e. URL), the site title (i.e. the title of the indexed page), and some other data such as bookmark creation date, last visit date and user-provided description of the indexed page. The bookmark set becomes a personal web space that helps the user in *land-marking* the huge information space composed of the currently available web pages. A major reason behind the popularity of bookmarks is their ease of use. In most browsers a simple click is enough to save the currently visited web page into the list of bookmarks. Later, another simple click on the saved bookmark takes the user directly to the required site. In order to enhance access time to saved bookmarks, most existing bookmarking tools allow users to organise their collections in a hierarchy of folders. While this may effectively enhance the access time, it introduces new problems: "users must continually trade-off the cost of organising their bookmarks and remembering which bookmark are in which folder versus the cost of having to deal with a disorganised set of bookmarks" [1]. Finding the appropriate folder for a given bookmark is not an easy task. The same bookmark may fit in more than one folder. A bookmark may require creating a new folder, splitting existing ones, etc. [17]. Currently, saving a bookmark in a given folder is not as easily supported, by current tools, as the creation of a bookmark. However, in this paper we make the assumption that users do organise their bookmarks in hierarchy of folders.

A user's bookmark collection can be a valuable information source for other users. Bookmarks are valuable for two main reasons. First of all, bookmarks are often the results of tedious and hard information searching process. If users can access others bookmark collections they can spare the required information searching effort. Lastly, because bookmarks are explicitly and intentionally added by the user, they give a precise evidence about the information interests of that user. By applying a collaborative filtering algorithm [21], communities of users that share the same interests can be identified. This can help to establish alternative information searching tools that are community-centred.

Recently, a number of academic and commercial systems have tackled the problem of building collaborative or shared bookmark repositories. Systems such as *Groupfire* (www.groupfire.com) and *MyLynx* (www.mylynx.com) allow users to save their bookmarks on a remote server. Users can select whether they wish to make some of their bookmarks public. The *WebTagger* system [13] makes the bookmark pool sharable and searchable by a group of users. The *GAB* system can automatically merge different user's bookmark lists in a single and a seamless hierarchy [25]. *RAPP* provides users with personalised help for bookmark classification and group-based bookmark recommendation service [5]. Other systems allow to build and to identify communities of interest by analysing user's bookmark collections. Communities are manually defined in *Pharos* [3], semi-automatically in *GroupMark* [19] and computationally in *KnowledgePump* [7].

Amazingly, almost all existing collaborative bookmarking tools seems to ignore major lessons learned in the last decade by the Computer Supported Collaborative Work (CSCW) [12, 20]. In this paper we outline an original collaborative bookmarking system called *CoWING* (for COllaborative Web IndexiNG), that enables an *organised* group of users to share their bookmarks, in an *implicit*, *secure* and an *effective* way. By implicit we mean that users are not required to do extra work in to sharing their bookmarks. The only additional work to do is to define other's access rights on their own repositories. A role-based access control service is provided in order to ease this task [23]. Secure sharing refers to the capacity of each user to control who knows what about her/his own bookmark collections. Finally effectiveness is ensured by recommending users with relevant bookmarks that are computed by applying a distributed collaborative filtering algorithm. The proposed system is implemented as a multi-agent system. Each user is associated with a learning-assistant agent called a *WING* agent (for Web IndexiNG). A *WING* agent observes the user behaviour and learns how the user classified her/his own bookmarks. Each *WING* interacts with peer agents in order to identify communities of interests. Each community is *centred* on a local bookmark folder. Identified communities are then used to recommend the user with new bookmarks that are likely to interest her/him. When the user access a bookmark folder, recommended items relative to that folder are displayed. The user can then accept or reject some or all of delivered recommendations. Thus recommendation evaluation is made in an appropriate context rather than being intrusive. *WING* agents can apply different interaction protocols in order to share their knowledge and compute recommendations. In this paper we discuss the most basic interaction protocol that allow agents to exchange raw date only (i.e. bookmark folders)

The reminder of this paper is organised as follows. The *COWING* system is detailed in section 2. First a quick overview is given in section 2.1. Some notations are introduced in 2.2 then the three main services implemented in the systems are discussed: the access control service, yser's bookmark classification learning and recommendation computation service. Related work is briefly presented in section 3. Finally we conclude in section 4.

2. The COWING System

2.1 System overview

The overall architecture of the *COWING* system is illustrated on figure 1.

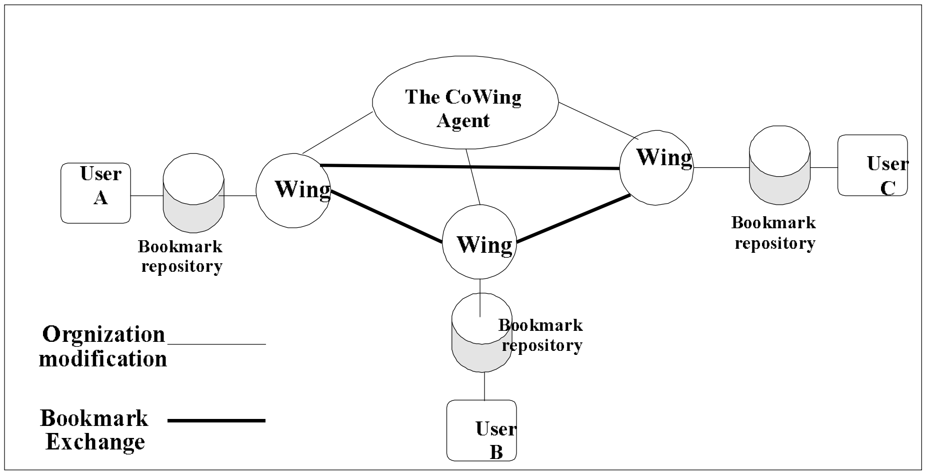

Fig. 1. The *CoWing* system architecture involving three users: A, B and C.

The system is composed of a central *CoWing* agent and a *Wing* agent per registered user. The *CoWing* agent acts as *Wing* agents registry. It provides *Wings* with each other addresses. In addition it provides *Wings* with a description of the users organisation hierarchy (see section 2.3). Each user manages her/his own hierarchy of bookmarks just as in single-user settings. However, users are required to set access rules that define which personal bookmarks to share with whom. An easy to use role-based access control service is provided for that purpose (see section 2.3). A *Wing* agent observes the associated user behaviour in order to *learn* how the user organises her/his bookmarks. In this goal, each *Wing* agent implements a hybrid incremental supervised Neural/CBR (Case-based Reasoning) classifier [18] (see section 2.4). The used classifier learns user's organisation policy from both positive and negative examples. For each user *u*, the set of positive examples is composed of the bookmarks explicitly added by the user to a given folder or bookmarks recommended by the system and accepted by the user. Negative examples are bookmarks that are deleted, moved from one folder to another or rejected by the user after being recommended by the system.

Roughly, bookmark recommendations are computed as follows. Each *Wing* agent asks peer agents to feed him with new bookmarks. When a *Wing B* receives a request from a *Wing A*, the former computes the *view A* has on its own repository. An agent view is composed of the set of bookmark folders and bookmarks for which the agent has the read access right. The agent *B* sends back to *A* bookmark folders that constitute *A*'s view on *B*'s repository. For each received folder *f*, *A* uses its classifier, switched to the classification mode, in order to classify bookmarks contained in *f*. If the *majority* of these bookmarks are classified in a same local folder f_l then *A* recommends to add all bookmarks contained in *f* into f_l. When the user consult the bookmark folder f_l S/he can confirm or reject the agent proposition. Depending on the user decision (i.e. confirm or reject) recommended bookmarks will be treated either as positive or as negative examples.

Next, we introduce some notations that will be used in describing the functioning of the *CoWing* System. Then we detail each of the three main services implemented in *CoWing*: the access control service, the bookmark classifier and the bookmark recommendation mechanism.

2.2 Notations and work hypothesis

A bookmark b_i described as a vector $b_i = < A_i, C_i, L_i>$ where:

- A_i is the address (i.e. URL) of the document indexed by the bookmark. We choose to model an address by a couple $A_i = <SA_i, FP_i>$ where SA_i is the server address on which the indexed document is located. FP_i is the file path of that document on the server machine. Notice that we only consider here static HTML documents.
- C_i is a vector composed of the *k* most significant words describing the content of the document indexed by the bookmark. *k* is a system parameter.
- L_i is the list of hyperlinks embedded in the document indexed by the bookmark. We note $L_i =\{l_j\}$. Each link l_j is described by a *couple* $l_j=<anc_j, dest_j>$ where anc_j is the link anchor and $dest_j$ is the address of the destination document of that link.

Bookmarks are organised in hierarchy of folders. Each folder may contain a set of bookmarks and a set of sub-folders. A bookmark b_i can belong to only one folder at one time (aliases and copies are not considered). This hypothesis is a restrictive one but it simplifies the implementation of the classifier system (section 2.3)

We note f^u_k the kth bookmark folder defined by user *u*. A bookmark folder f^u_k is defined as a couple $f^u_k = <B_k, f^u_{kf}>$ where B_k is the set of bookmarks contained in the folder f^u_k, and f^u_{kf} is the identifier of the super folder of f^u_k. Information about bookmark and bookmark folders are obtained by continuously monitoring the user's bookmark file similarly to the procedure described in [4].

Bookmarks similarity. Given two bokmarks b_i and b_j. the function $Sim(b_i, b_j)$ measures the similarity between b_i, b_j.. This function takes as arguments the description of both bookmarks and returns a numeric value between 0 and 1. 1 for describing maximum similarity (i.e. identity) and 0 for denoting extreme dissimilarity. This similarity function is an aggregation of basic similarity functions defined on each of the bookmark attributes. Next we give similarity functions defined over the three bookmark attributes.

Address similarity. Given two addresses *a* and *b*, we define an address similarity function *SimAdr* as follows:

- $SimAdr\ (a,b) = 0$ *if* $a.SA \neq b.SA$ /* two different web sites */

- Otherwise: $SimAdr(a,b) = 1- h(a.FP, MSCA(a.FP,b.FP) + h(b.FP,MSCA(a.FP,b.FP) /h(a.FP,root) +h(b.FP,root)$

Where the function *h()* returns the number of links between two nodes in the documents tree and *MSCA()* returns the most specific common ancestor of two nodes in a tree. This similarity measure is based on the hypothesis that two documents that are placed in the same directory on the same server are similar to each other. More the directory is deep in the server hierarchy more the documents are related to each other.

Content similarity. Given two keyword vectors *u* and *v* we define a content similarity function *SimCont* as follows: $SimCont(u,v) = Card(u \cap v) / Card (u \cup v)I$ where *card()* is the cardinality function. A similar function is applied to measure embedded links similarity.

2.3 Access control service

Although privacy protection is a central issue in collaborative information systems [14], few existing systems provide an adequate protection model. For instance, existing collaborative bookmarking systems either do not provide a protection system, either implement a primitive protection policy where user's can distinguish between private and public data. These simple protection schemes mismatch collaboration requirements [6,10,12,20]. In real world settings users need to express *fine grained* access control rules. A user u_i may wish to share a bookmark folder with some user u_j but not with user u_k. Moreover a user u_i may wish to share with user u_j a given bookmark folder *f* but not some specific bookmarks that are saved in *f*. Role-based access control models has been proposed in order to allow fine-grained, easy to use access control specifications [24]. In *COWING*, we implement a modified version of the role-based access control model described in [10, 11]. The implemented model is described here after.

A role R_i is an object that contains of a set of *access rules*. An access rule AR_i is defined as a triple $AR_i = < [+|-], o, a>$. The leading sign determines whether the right *a* granted over object *o* is positive or negative. Negative rights are introduced in order to ease access right specification [27]. The object *o* can be a single bookmark *b* or a bookmark folder *f*. An access right *a* can be one of the following rights: *read*, and *modify*. The modification right implies the read one.

Two types of role objects can be distinguished:

- *Organisational role*: These are roles that describe abstract positions in the user's community. For example, in a academic research group we can define the following abstract roles: researcher, student, Ph.D. candidate, Administrative assistant, Web master, librarian, etc.
- *User roles*: each user has her/his own user role. This describes access rules granted to the user.

The set of organisational roles form what we call the *organisation model*. A hierarchical structure is defined over the organisation model. This gives the organisation model a structure of a direct acyclic graph (DAG). The set of user's roles has a flat structure. Each user role object is linked to one or more organisational roles. Figure 2.a illustrates an example of the relations between user roles and the organisation model. We call the whole set of roles (organisational and user roles) the *extended organisation model.*

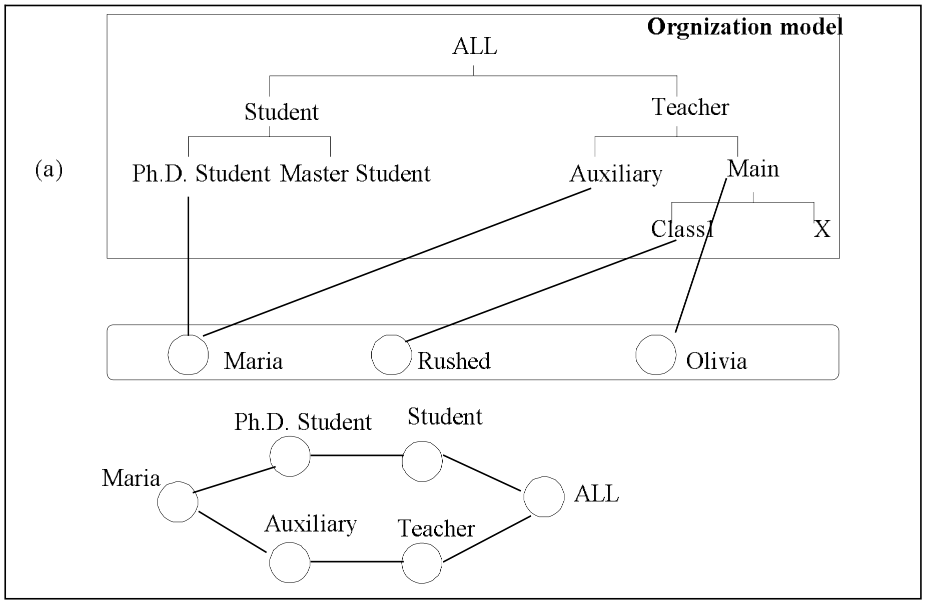

Fig. 2. The access control model components

The organisation model is defined and administrated by the application administrator. Each Wing agent holds a copy of the extended organisation model. All *WING* agents share the same structure however, access rules associated with each role (user or organisational one) differ from one *WING* to another. When agent *B* needs to compute agent *A* view on its local repository, it constructs what we call A's access right *DAG* (noted *AC/DAG*). This graph is constructed by climbing up links that relate the user role A to the organisation model. Figure 2.b illustrates the *AC/DAG* of the *WING* agent associated with the user named Maria. All folders in B's repository for which *A* has at least the read right are added to the computed view. The general idea of the evaluation algorithm is to evaluate A's read right in terms of access rules contained in A's *AC/DAG*. Starting by A's user role, if no explicit answer, rejection of confirmation, is obtained then the evaluation function consider rules contained in next role in the *AC/DAG* graph. The graph is explored in a depth first way. This exploration rule is necessary to avoid ambiguity in evaluating the access rule. More details about the access control model can be found in [10].

2.4 Learning to classify

Each *WING* agent uses a hybrid neural/ case-based reasoning (CBR) classifier in order to learn the user's bookmark classification strategy [18]. CBR is a problem solving methodology that is based on reusing past experiences in solving problems in order to solve new problems. A case is classically composed of two parts the problem part and the solution part. To solve a new problem the system retrieves from its *memory* (called also the case base) all cases that the problem part is *similar* to the problem to solve. Solutions proposed by retrieved cases can be adapted to propose a solution to the new problem. In our application, the problem part of a case is composed of set of the attributes of a bookmark (see section 2.2), the solution part is the folder identifier in which the bookmark is filed by the user.

The used classifier memory model, called *PROBIS*, is based on the integration of a prototype-based neural network and a flat memory devised into many groups, each of them is represented by a prototype. *PROBIS* contains two memory levels (see figure 3), the first level contains prototypes and the second one contains examples. The first memory level is composed of the hidden layer of the prototype-based neural network. A prototype is characterised by :

1. The prototype's co-ordinates in the *m*-dimensional space (each dimension corresponding to one parameter), these co-ordinates are the *centre* of the prototype.
2. The prototype's influence region, which is determined, by the region of the space containing all the examples represented by this prototype.
3. The class to which belongs the prototype (i.e. a bookmark folder)

The second memory level is a simple flat memory in which examples are organised into different zones of similar examples. These two levels are linked together, so that a memory zone is associated with each prototype. The memory zone contains all examples belonging to this prototype. A special memory zone is reserved for atypical examples. These are examples that do not belong to any prototype.

The classifier system operates either in *learning* mode or in *classification* mode. The system can switch from one mode to another at any moment. Before the *first learning phase*, the system contains neither prototypes nor zones of examples. Examples for training are placed initially in the atypical zone. Prototypes and associated zones are then automatically constructed. An incremental prototype-based neural network is used to construct the upper memory level. Particular and isolated examples are kept in the atypical zone whereas typical examples are transferred to the relevant typical zones. This memory organisation helps to accelerate the classification task as well as to increase the system's *generalisation capabilities*. In addition adding a new example is a simple task, the example is added in the appropriate memory zone and the associated prototype is modified.

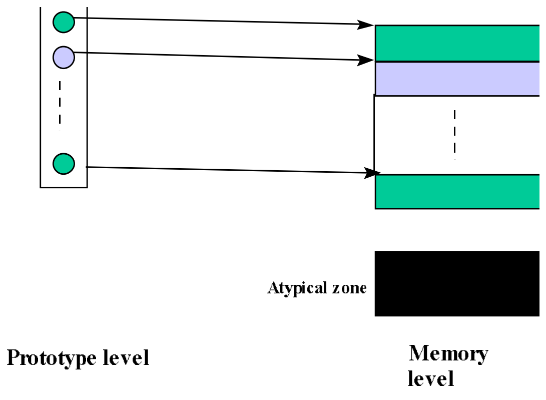

Fig. 3. The memory is composed of two levels: prototypes and stored examples

The learning procedure is the following:

1. If the new example does not belong to any of the existing prototypes, a new prototype is created (this operation is called *assimilation*). This operation is accomplished by adding a new hidden unit to the neural network. The co-ordinates of this prototype and the *radius* of the influence region is initialised to a maximal value (this is a system parameter). A new memory zone is also created and linked to the prototype. The new example is added to the new memory zone.
2. If the new example belongs to a prototype whose class value is the same as the example, the example is added to the associated zone of the second level memory. The prototype co-ordinates are modified according to the *Grossberg* learning law |10] to fit better the new example (this operation is called *accommodation*). The vector representing the prototype co-ordinates and memorised in the weights of the links going from the input layer to this prototype is modified according: $W_{pro}(t+1)= W_{pro}(t)+g(t)*Sim\ (b_i- W_{pro}(t))$ where b_i is the vector representing the bookmark to classify, $g(t)$ is a decreasing series which tends to 0, and *Sim* is the bookmark similarity function.
3. If the new example belongs to a prototype whose class value is not the same as the example, the radius of this prototypes is decreased in order to exclude the new example of this prototype (this operation is called *differentiation*). The new example is introduced again to the neural network and the most similar prototype (if there is any) is activated again and one of the three previous conditions is right.

Build prototypes approximate the folders in the bookmark repository. Atypical examples correspond to bookmarks that can be classified in more than one folde.

2.5 Learning to recommend

The bookmark recommendation computation is performed as follows. Each WING agent maintains locally two data structures: an *agenda* and a *folder correlation matrix (FCM).*

The agenda is a dictionary structure where keys represent identifiers of peer *WING* agents to contact and values are next contact dates. Hence *Agenda[i]* gives the next contact date with agent *i*.

The *FCM* is a *mXn* matrix where *m* is the number of folders in the local repository and *n* the number of peer agents known to the local agent. An entry *FCM[i, j]* is a couple $<f^{j}_{k}, cor_{ij}>$ where f^{j}_{k} is a folder identifier maintained by user u_j and cor_{ij} is the correlation degree between the folder f^{j}_{k} and the folder f^{i}_{k} maintained by local agent.

Correlation between two folders f_1 and f_2 is given by the number of bookmarks contained in folder f_2 that are classified in folder f_1 divided by the total number of bookmarks in f_2. In the *FCM* matrix, an entry $FCM[i,j]= <f^{j}_{k}, cor_{ij}>$ is computed by taking the folder f_k from the agent *j* repository that have the *maximum* correlation value with folder *i* belonging to the local repository.

Given a *WING* agent A, the bookmark recommendation process is made by executing the following algorithm:

```
1   For each B agent in Agenda do
2     If Agenda[B] is over then          ; it is time to contact
3       send B a bookmark request        ; the B agent
4       receive from B: V and ND         ; V is A’s view on B repository
5       Agenda[B]= ND;                   ; ND the next contact date
6       For each f in V                  ; f folders in view V
7           <i,c>=computeCorrelation(f)  ; i is the local folder with
8           If FCM[i,B].cor < c then     ; highest correlation with f
9               FCM[i,B]= <f,c>          ; c is correlation of i and f.
10      If FCM[i,B].cor > δ then         ; δ is a minimum correlation
           recommend to add bookmarks    ; threshold
           in f to the local folder i
```

Figure 4 illustrates the interaction protocol between two Wing agents. The function *computeCorrelation* (line 7 in above algorithm) finds the folder *i* in the local repository that have the highest correlation value with a folder f as defined above. The function proceeds as follows. For each bookmark b_i in f the local neural/CBR classifier is applied. For each bookmark, the classifier responds by the identifier of a local folder. The folder that has been selected the most will be the returned folder. Notice that the correlation relation is not symmetric since correlation is computed by using local classifiers (the classifier is different from one agent to another) and by using information contained in the local agent view on the repository of the other agent.

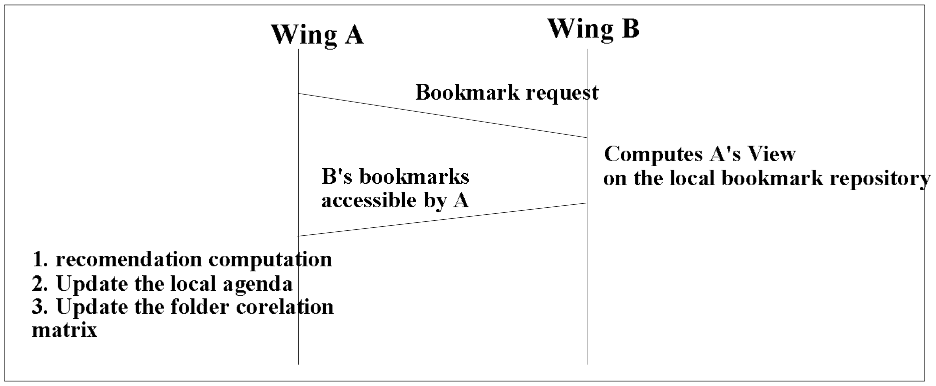
Wing A
Wing B
Bookmark request
Computes A's View on the local bookmark repository
B's bookmarks accessible by A
1. recomendation computation
2. Update the local agenda
3. Update the folder corelation matrix

Fig. 4. Interaction protocol between *WING* agents

2.6 Experimental results

In order to validate our approach we have applied the following experimentation protocol. We start by forming a synthetic collection of bookmarks. The total number of bookmarks is 300. These bookmarks are grouped in 3à folders. The mean number of bookmarks per folder is 10. Starting from this bookmark collection we randomly generated ten other collections by modifying each by up to 35%. Two types of operations are possible in order to modify a folder:

1. delete a bookmark from the entire collection,
2. move a bookmark to another folder.

Notice that we assume that a bookmark may not belongs to two different folders at the same time. The generated bookmark collections verify, by construction, this property. The modification percentage (i.e. 35%) ensures a suitable overlapping between the different collections of bookmarks. The system performances are evaluated by two criteria:

- The *learning ratio* that measures for each classifier the precision of good classifications of examples belonging to the learning set (i.e. local bookmarks used to build the classifier)
- The *generalisation ratio* that measures the precision of recommending a bookmark o the right folder. The right folder of a bookmark is the original folder where the bookmark was in the initial collection.

A set of ten different experiences has been conducted. The average obtained learning ratio is 93,3% and the average generalisation ratio 86,2%. While these figures are encouraging, we should admit that these will not be the same is real world settings where overlapping ration among bookmark folders is far below the artificial overlapping threshold we have imposed in our experimental work.

3. Related Work

Few systems are proposed in the literature to cope with the problem of collaborative bookmark management. Almost all-commercial systems are based on implementing a central shared URL repository that allows users to store and retrieve URLs. Some shared URL repositories, such as *MyLynx.com* allow user to define a private section and a public section.

Examples of shared bookmark systems are the *GAB* system [25], *KnowledgePump* [7], *Pharos* [3]. The GAB system offers a service that allows merging different user bookmark repository in a virtual centralized bookmark. However no recommendation mechanism is implemented. It is up to the users to navigate in the merged repository to find bookmarks they are interested in. A comparable approach is also implemented in the *PowerBookmarks* systems [15]. Both *KnoweldgePump* and *Pharos* provide users with the possibility to share a centralised bookmark repository. The repository hierarchy is defined by a system administrator. Both systems provide also customisation service in order to recommend users with bookmarks that are more interesting for them in given folder. Recommendation computation is made by applying a collaborative filtering mechanism that is base on matching the characteristics of bookmarks added and accessed by each user.

Most similar to our work is the *RAAP* system [5]. In *RAAP* the system also learns by using a classical classifier how users classify bookmarks and use this information to recommend people with new bookmarks. However, *RAAP* has the disadvantage of being built on a centralised repository. It provides a poor access control model.

Related also to our work is the *Yenta* system [6]. *Yenta* is presented by its authors as a matchmaking system. It aims at discovering matchmaking between people based on comparing shared interests. The principal of *Yenta* could be easily applied to built a collaborative bookmark system. The accent is put on distributing the computation of the matchmaking function.

4. Conclusion

In this paper we have presented *CoWing*: a multi-agent collaborative bookmark management system. The *COWING* system addresses mainly the resources discovery problem. It provides a mean that allow users to share their bookmarks, in a personalised way without asking users to do extra task except for defining others access control on their own repositories. Each user is assisted by a personal agent, the *CoWing* agent that uses a hybrid neural/CBR classifier that learns the user strategy in classifying bookmarks. The learned classification strategy is used to construct associations between bookmark folders belonging to other users.

Experiments made on *synthetic data* show that our approach is valid. However, we believe that some enhancements should in order to make the system operational in real work settings. One important issue concerns the cold start problem [11, 25]. The applied recommendation computation approach makes the hypothesis that users have organised their bookmarks in a hierarchy of folders. Each folder has some semantic sense. While lot of users do use hierarchical bookmark structures, some still using flat organisation structures [2]. Another related problem is the witnessed low overlapping between different user bookmark repositories [4]. We are working on proposing solutions to these two problems. Future work concerns also the extension of the system to handle the two other problems of bookmark maintenance and organisation.

References

1. D. Abrams, R. Baecker, and M. Chignell. (1998). Information Archiving with Bookmarks: Personal Web space Construction and Organization. *In Proceedings of ACM Conference on Human Computer Interactions (CHI'98),* Los Anglos, 18-23 April pp. 41-48.
2. Abrams, D. (1997) *Human Factors of Personal Web Information Spaces*. MS Thesis, Department of Computer Sciences, University of Toronto, 1997, Also available at http://www.dgp.torento.edu/~abrams
3. Bouthors V., and Dedieu O. (1999). Pharos, a Collaborative Infrastructure for Web Knowledge Sharing. *In Proceedings of the third European Conference On Research and Advanced Technology for Digital Libraries (ECDL'99)* Abiteboul S., and Vercoustre A. Eds), LNCS No 1696, Paris September, 1999, pp. 215-233
4. A. Cockburn, B. McKenzie. What Do Web Users Do? An Empirical Analysis of Web Use. In International Journal on Human-Computer Studies, 2000.
5. J. Delgado, N. Ishii and T. Ura. Intelligent Collaborative Information Retrieval. In proceedings of Progress in Artificial Intelligence, IBERAMIA'98, LNAI 1484, pp. 170-182.
6. Foner, L.N. (1999) Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System, Ph.D. Thesis, Massachusetts Institute of Technology, June 1999.
7. Glance, N., Arregui, D., and Dardenne M. (1999) Making Recommender Systems Work for Organizations. *In Proceedings of PAAM'99*, London April 1999.
8. S. Grossberg, Competitive learning: From interaction activation to adaptive resonance, Cognitive Science, n°1, 1987, pp. 23-63.
9. Grudin, J. (1994) Groupware and Social Dynamics: Eight challenges for developers. Communication of the ACM 37,1 (January 1994), pp. 92-105.
10. R. Kanawati. Groupware: Control and Architectural issues. Ph.D. Thesis, Institut National Polytechnique de Grenoble, November 1997. 172 pages (In French).
11. Kanawati, R. (1998) COLT: Yet Another Integrated Collaborative Environment. *In Proceedings of the third International Conference on the Design of Cooperative Systems (Dareses, F., and Zaraté, P. Eds)* Volume II Cannes 26-29 May 1998, pp. 99-102
12. R. Kanawati, M. Malek. Informing the design of shared bookmark systems. In proceedings of RIAO'2000, Paris, 2000.
13. R. M. Keller, S. R. Wolf; J. R. Chen, J. L. Rabinowitz and N. Mathe. A Bookmaking Service for Organizing and Sharing URLs. In proceedings of the 6th International Conference on the World Wide Web, Santa Clara, CA, April 1997.
14. T. Lau, O. Etzioni D.S. Weld. Privacy Interfaces for Information Management. Communications of the ACM 42(10), April 1999. pp. 88-94
15. W.S. Li, Q. Vu, D. Agrawal, Y. Hara and H. Takano. PowerBookmarks: A system for Personalizable Web Information Organization, Sharing and Management. *In proceedings*

of the 8th International World Wide Web Conference (WWW'8), Toronto, Canada. May 1999.

16. Lim, J-G., (1994) Using Cool-lists to Index HTML Documents in the Web, *In Proceedings of the 2nd International Conference on the World Wide Web* (WWW'2) Chicago, IL, 1994 pp. 831-938.
17. Maarek, Y.S., Ben Shaul, I.Z. (1996), Automatically Organizing Bookmarks per Contents. *In Proceedings of the 5th International World Wide Web Conference*, Paris, 6-8 May.
18. M. Malek. Hybrid approaches Integrating Neural Networks and case based reasoning: from Loosely Coupled to Tightly Coupled Models. In K.P. Sankar, S. D. Tharam and S. Y. Daniel (Eds) Soft Computing in Case-based Reasoning. Spinger, 2000 pp.73-94
19. Mathe, N. and Chen, J. R. (1998); Organizing and Sharing Information on the World-Wide Web using a Multi-agent System; *In proceedings of ED-MEDIA'98 Conference on Educational Multimedia and Hypermedia,* Freiburg, Germany, June 1998.
20. Markus, M. L., and Connolly, T. (1990) Why CSCW Applications Fail: Problems in the Adoption of Interdependent Work Tools. *In Proceedings of the International Conference on Computer Supported Collaborative Work (CSCW'90),* New York 1990.
21. Resnick P.(1997) Recommender Systems, *Communications of the ACM* 40(3), 1997
22. Rucker J., and Marcos J. P., (1997) Siteseer: Personalized Navigation for the Web. *Communications of the ACM* 40(3), pp. 73-75. 1997
23. R. Sandhu, E. Coyne, H. Feinstein, C.Youman. Role-Based Access Control Models. *IEEE Computer; 20(2),* 1996 pp 38—47,
    URL: http://citeseer.nj.nec.com/sandhu96rolebased.html
24. H. Shen, P. Dewan. Access Control for Collaborative Environments. *In proceedings of the ACM Conference on Computer-Supported Cooperative Work (J.Turner and R. Kraut eds) (CSCW'92),* Torento, Canada, 1992. pp 51-58
25. Wittenburg, K., Das, D., Hill W., and Stead L. (1997) Group Asynchronous browsing on the World Wide Web. *In proceedings of the 6th International Conference on the World Wide Web (WWW'6),* 1997