Computing Social Networks for Information Sharing: A Case-Based Approach

Rushed Kanawati[1] and Maria Malek[2]

[1] LIPN – CNRS UMR 7030, 99 Av. J.B. Batiste F-93430 Villetaneuse
rushed.kanawati@lipn.univ-paris13.fr
[2] LAPI – EISTI, 11 Av. du Parc F-95011 Cergy
maria.malek@eisti

Abstract. In this paper we describe a peer-to-peer approach that ails at allowing a group of like-minded people to share relevant documents in an implicit way. We suppose that user save their documents in a local user-defined hierarchy. the association between documents and hierarchy nodes (or folders) is used by a supervised hybrid neural-CBR classifier in order to learn the user classification strategy. This strategy is then used to compute correlations between local folders and remote ones allowing to recommend documents without having a shared hierarchy. Another CBR system is used to memorize how good queries are answered by peer agents allowing to learn a dynamic community of peer agents to be associated with each local folder.

Keywords: Collaborative Document Sharing. Peer To Peer, Case-based reasoning; Community identification.

1 Introduction

Social networks based information searching and dissemination approaches have gained recently an increasing attention as a promising approach to handle the problem of information searching on the Internet. The idea is to provide a computer-mediated support that allow organized group of like-minded people to share their experiences in information searching. Actually, it is reasonable to expect enhancing the outcome of a personal information searching process by providing the user with some access to searching experiences of other like-minded users [4, 7]. Useful experiences include: a) relevant founded documents, b) interesting information sources as well as c) usage information indicating how to better use existing searching tools. In this work we focus on the problem of documents sharing among an organized group of like-minded people. Examples are shared bookmark systems [10, 11, 12, 13], and collaborative bibliographical data management systems [2, 6, 9]. The problem here is to recommend users that are interested in some topic T with documents that are effectively relevant to T by exploring the set of judgments made by the group's members about the relevancy of documents to interest topics. Relevancy judgment can be made explicitly by users or inferred, in an implicit way, by observing user's actions.

Interests topics are usually organized in a hierarchy. The goal is to ease document indexing and recall processes. Hierarchies can either be user-defined as it is the case of individual web bookmarking tools, or system-defined such as the case of using some domain ontology to index the documents. According to the sharing degree of the interests topic' hierarchy among the users we classify document sharing approaches into three classes:

1. *Class 1*: Systems where users share the same topic hierarchy with the same interpretation of each topic. Most existing systems fall into this class. Such systems have often a centralized architecture [1, 3]. Users register, in an implicit or in an explicit way, relevancy judgment into a central repository. The problem here is to compute (or to predict) how relevant would be a document *d* for a user *u* knowing the relevancy of this document to other users [3].
2. *Class 2*: Systems where users share the same hierarchy of interest' topics but each has her/his own interpretation of topics. A document is judged relevant to a topic T by one user may be judged relevant to other topics (related to T) by others. In addition to the problem of predicting document relevancy for each user, systems belonging to this class should also handle the problem of heterogeneity of topics interpretation [9]
3. *Class 3:* Systems where each user manages her/his own topic' hierarchy. This class can be seen as a generalization of the previous one. In addition to the above mentioned problems, systems from this class should find a mapping between the different topics hierarchies used by different users.

In this paper we describe a collaborative case-based reasoning (CBR) approach for implementing systems from the third class. The basic idea of the proposed approach is to associate with each user a CBR classifier that learns how the user classifies documents relevant to her/his own hierarchy of interest topics. The use of a CBR classifier is motivated by the incremental learning capacities that can handle dynamic changing in user's classification strategies. Mapping topics defined by user *U1* to those defined by another user *U2* is computed by classifying documents provided by *U1* using the classification knowledge (i.e. cases) related to *U2*. In other words the similarity measure between two topics (defined by two different users) is itself computed by a CBR system. This mapping function is used to infer, for each user and for each topic, the most appropriate social network to fed the local topic with relevant documents. The reminder of this paper is organized as follows: The document recommendation approach is described in section 2. Subsection 2.1 gives a general overview of the approach. Then the three main components of the approach : Learning to classify, learning to recommend and community learning are describes inn subsequent subsections. Ann example of applying the recommendation approach for implementing a bookmark recommendation approach is given with first experimental results are given in section 2.5. Related work are reported in section 3 and a conclusion is given in section 4.

2 Our Approach

2.1 General Description

The goal of the proposed document sharing approach is to allow a group of like-minded people to share their documents in an implicit way [7]. Let U be the group of users. $U = \{u_1, u_2, \ldots, u_n\}$ where n is the number the group members. The sharing system functions as follows: each user u_i manages her/his own document collection. We assume that each user u_i organizes his/her documents in a collection of folders $F_{ui}= \{f_{ui}^j\}$. The same document d can be saved to more than one local folder. Each user u_i is associated with a personal software assistant agent A_{ui}. The role of this agent is to learn to map, if possible, each local folder f_{ui}^j with folders managed by other users. In other words the task of agent A_{ui} is to learn a mapping function $M_{ui} : F_{ui} \rightarrow \prod_j F_{uj}\, j \neq i$. Computing this mapping function requires computing a correlation measure between local and remote folders. A local folder f_{ui}^j will be mapped to highly correlated remote folders. The huge number of available documents does not allow to use simple folder correlation measure such as $correlation(f_1, f_2) = |f_1 \cap f_2| / |f_1 \cup f_2|$ since two effectively highly correlated folders may have a low correlation degree computed by such a function. A document similarity function could be used to cope with this problem. Correlation between two folders could be expressed by how much documents in f_1 are similar to those in f_2. Another problem to cope with is that folders do not necessarily express a class of documents. Folders are basically defined by users as a document organization entity. Different users apply different document classification criteria. Therefore using a mere document similarity function ay lead to poor folder correlation detection. In our approach we propose to compute folder correlation as a function of their usage similarity. More precisely, a correlation degree between folder f_{ui} (created by user u_i) and folder f_{uj} (created by another user u_j) is given by the ratio of documents in folder f_{uj} that would be classified in folder f_{ui} according to user u_i classification scheme. Hence in order to compute the mapping function we first need to model each user classification scheme. This is performed by using a incremental supervised classifier. Classes to be recognized are defined by the set of folders created by the user. The proposed classifier is described in next section. A collaboration protocol is described in section IV that allow each agent to compute correlation between its local folders and remote ones. Documents added to remote folders that are highly correlated to a local folder f will be recommended to be added to f. From the classification learning point of view, each document recommendation will constitute either as a positive example (if the recommendation is accepted by the user), or either as a negative example (if the recommendation is rejected by the user). The classification learning and the older correlation computation are described in the next two following sections.

2.2 Learning to Classify

In order to learn the user document classification scheme, each personal agent implements an incremental hybrid neural/case-base reasoning classifier. This classifier called PROBIS and initially proposed in [14] 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, 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 characterized by:

1. *The prototype's center*. This is given by the co-ordinates in the m-dimensional space (each dimension corresponding to one parameter), these co-ordinates are the center of the prototype.
2. *The prototype's influence region*. This is determined, by the region of the space containing all the examples represented by this prototype.
3. *The class* to which belongs the prototype. In our application the class is the folder identifier in which the document is saved.

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. Documents that belongs to more than one folder will be typically saved in this memory zone. 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 organization helps to accelerate the classification task as well as to increase the system's generalization 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. 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 initialized 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 [5] to fit better the new example (this operation is called accommodation). The vector representing the prototype co-ordinates and memorized in the weights of the links going from the input layer to this prototype is modified as follows:

$$W_{pro}(t+1)= W_{pro}(t)+g(t)*Sim\ (b_i- W_{pro}(t))$$

where b_i is the vector representing the document to classify, g(t) is a decreasing series which tends to 0, and *Sim* is the document 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 any) is activated again and one of the three previous conditions is right.

2.3 Learning to Recommend

The goal of an assistant agent is to gather form peer agents documents that can be relevant to be added to a local folder. Given a target local folder *f,* an agent applies the following steps for computing documents relevant to *f* : First, the agent computes summery of *f*. Actually, the summery is a keyword list containing the k-most frequent keywords that are listed in descriptions of documents stored in *f.* Then the agent applies the community formation algorithm, described in section 2.4, in order to get the list of peer agents that are likely to provide most relevant documents. A recommendation request is sent to each member of the computed community. A recommendation request contains the following informations: the sender agent identifier, the target folder identifier, the folder summery and the list of document descriptions of documents in *f*. Selected peer agents respond to a recommendation request by sending back : a list of relevant folder's identifiers and a list of recommended agents. Local folders whose correlation degree with the received folder is above a given threshold *t* are said to relevant to the recommendation request. The correlation between folder *f* and a local folder *g* is given by:

$$\texttt{correlation(f,g)} = |\texttt{d}_\texttt{i} \texttt{ in f : Class(d}_\texttt{i}\texttt{) = g} | / |\texttt{f}|$$

Where `Class(x)` is a function giving the predicted class, according to the local classifier, of document $\texttt{d}_\texttt{i}$. With each sent folder, the agent associate the computed correlation degree.

The list of recommended agents is computed by applying the local community formation algorithm (see section 2.4) using the received folder summery. This facility allow to propagate among agents of the system the *expertise* of the different agents. Notice that either of the both answer lists (recommended folders end agents) can be empty. Upon receiving answers for the selected peer agents, the initiator agent used the lists of relevant folders with their correlation degrees for updating a local folder correlation matrix (FCM). The FCM is a *m X n* matrix where *m* is the number of folders in the local repository and *n* the number of peer agents known to the agent. An entry *FCM[i, j]* is a set of couples $<f_{jk}, cor_{ij}>$ where f_{jk} is a folder identifier maintained by user u_j and cor_{ij} is the correlation degree between the folder f_{jk} and the folder f_{ik} maintained by local agent.

The *FCM* matrix is used by an assistant agent in order to determine which remote folders are highly correlated to a given local folder *f*. A *folder request message* is then sent for agents that have the selected remote folders. Upon receiving a folder request message, an agent send back all documents contained in the requested folders. The initiator agent merges the list of received documents (in case it requests downloading more than one remote folder) and the top K-documents will be presented to the associated user when accessing the local folder *f*. *K* is a user defined parameter that allow the user to limit the number of new documents to be recommended at once when accessing a local folder. This two step document recommendation computation process aims at reducing the network traffic by transmitting over the networks only documents to be effectively recommended to the user.

Received lists of recommended agents are used by the community formation module, described in the next section, in order to complete its information about the peer community associated to a given local folder. The user evaluates the recommended documents by simply accepting or refusing each of these documents. Recommendations provided by a given agent are evaluated using the classical precision and recall criteria defined as follows:

$$\text{Precision (summery, } a_i) = |\text{ accepted recommendations provided by } a_i| \,/\, |\text{ recommended documents }|$$

$$\text{Recall (summery, } a_i) = |\text{ accepted recommendations provided by } a_i| \; |\text{ accepted recommended documents }|$$

The user feed back is used, on one hand, to update the the local agent classification knowledge as described in section 2.2. On another hand, it is used to evaluate the recommendations provided by peer agents. This evaluation is used by the community computation module, as described in the next section.

2.4 Community Computation

In order to avoid sending recommendation requests to all known peer agents, we provide each assistant agent with learning capability that allow to compute for each local folder a set of peer agents that are most likely to provide relevant documents. We call these agents, the *folder community*.

We use a second CBR reasoner in order to compute folder's communities. A source case is classically composed of a problem part and a solution part []. In this subsystem, the problem part is given by a keyword list that summarize a folder content while the solution part is composed of the list of peer agents identifiers. With each agent identifier is associated a the highest correlation degree obtained from this agent and, if it exists, the user evaluation of the recommendations provided by this remote agent answering a recommendation request using the folder summery provided in the problem part.

Initially, the case base is empty. As a consequence, the local agent will send the recommendation request to all known agents. Upon receiving recommendations and a the user feedback, a new source case can be added to the case base. The recommendation evaluation criteria are used to compute a *trust degree* in the concerned remote agent. The trust degree is defined as the product of the computed precision and recall. Notice that only agents that have answered a folder request message (see previous section) can be evaluated. All other agents that have answered a recommendation request message will be assigned a neutral trust degree (i.e. 0.25 on a scale from 0 to 1, that to say that both precision and recall are equal to 0.5).

Before sending a new recommendation request with a target folder *f*, the agent computes the new summery of *f*. The summery of a folder changes in function of the actual set of documents that are stored in *f* but also in function of the summery of other local folders. the case retrieval phase computes the similarity between the new folder summery and the problem part of each stored case. Cases with similarity measure above a given threshold t_r will be retrieved. Each retrieved case provides a set of agents to contact. The obtained lists of agents are then merged and the result list is sorted in function of the trust degree. The top K-agent (where K is another system parameter) will form the folder community.

2.5 Application: Collaborative Bookmarking

In oder to illustrate our document recommendation approach, we have applied the proposed approach in the context a collaborative bookmark management system. In this application a document is represented by a bookmark. A bookmark is described by a couple : 1) the address of the indexed web site (i.e. the site URL) and 2) by a set of keywords that summarize the content of the indexed page. Hence the similarity between two bookmarks is defined as a weighted sum of two basic similarities defined overs URL and keyword lists.

$$sim(b1, b2) = \alpha\, URLSim(b1.url, b2.url) + (1-\alpha)\, ContSim(b1.Cont, b2.cont)$$

Where $0<\alpha<1$ is weight of the address similarity. The address similarity *URLSim* function is defined as follows:

$$URLsim(a,b) = 0 \text{ if a and b have different web servers.}$$
$$URLSim(a,b) = 1- h(a.FP, MSCA(a.FP,b.FP) \quad + \\ h(b.FP,MSCA(a.FP,b.FP) /h(a.FP,root) +h(b.FP,root)$$

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.

The content similarity function is defined by : $ContSim(u,v) = | u \cap v | / | u \cup v |$.

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 30 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 classifi cations of examples belonging to the learning set (i.e. local bookmarks used to build the classifier)

- The generalization ratio that measures the precision of recommending a bookmark of 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 generalization 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

A number of document sharing and recommendation systems are proposed in the scientific literature. Most of existing systems fall in the first two classes defined in section 1.

Pharos [1] and *KnoweldgePump* [3] provide users with the possibility to share a centralized document repository. The repository hierarchy is defined by a system administrator. Both systems provide also customization service in order to recommend users with documents 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.

GAB is a shared bookmark system that allows merging different user bookmark repository in a virtual centralized bookmark [16]. No recommendation mechanism is provided. 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 *Power-Bookmarks* systems [11]. *CoWing* in a peer-to-peer (P2P) multi-agent collaborative bookmarking system [7, 8]. *CoWing* is the our first prototype of document sharing system. In this early prototype agents send recommendation request to all known agents and all agents respond directly with the set of documents to recommend. No Community formation function is provided. *Bibster* [2, 6] is a P2P bibliographical data sharing system. In this system, each agent publishes its own expertise for all agents. The published expertise is used to match agents to actual need of recommendation expressed by an agent.

Most related to our approach are the *COBRAS* system [9] and the *REMINDIN* approach [15]. Both systems provide a query routing in a P2P network based on observing queries that are successfully answered by peer agents, these queries are memorized locally. Subsequently, a peer uses this information in order to select others peers to forward request to.

4 Conclusion

In this paper we've proposed a new approach for allowing a a group of like-minded people to share documents in an implicit and intelligent way. Users are not required to do extr effort to get document recommendations from others. They are only required o maintain their personal documents in a hierarchy of folders (as most do). Personal software agents observes users in order to learn document classification rules. This knowledge is used to compute a correlation degree between local folders and remote ones. In addition, agents uses feed-back provided by the user when accepting or rejecting recommendations in order to learn to associate a community of peer agents for each local folder. Recommendation request concerning a local folder is only sent to the community avoiding broadcasting the request to all known peers.

References

1. Bouthors, V., Dedieu, O.: Pharos, a Collaborative Infrastructure for Web Knowledge Sharing. In: Abiteboul, S., Vercoustre, A.-M. (eds.) ECDL 1999. LNCS, vol. 1696, pp. 215–233. Springer, Heidelberg (1999)
2. Brockstra, J., et al.: Bibster: A semantic-based bibliographic P2P syste. In: Proceedings of the second workshop on semantics in P2P & grid computing, New York, pp. 3–22 (May 2004)
3. Glance, N., et al.: Making recommender systems work for organization. In: proceedings of PAAM'99, London (April 1999)
4. Delgado, J., Ishii, N., Ura, T.: Intelligent Collaborative Information Retrieval. In: Coelho, H. (ed.) IBERAMIA 1998. LNCS (LNAI), vol. 1484, pp. 170–182. Springer, Heidelberg (1998)
5. Grossberg, S.: Competitive learning: From interaction activation to adaptive resonance. Cognitive Science 1, 23–63 (1987)
6. Hasse, P., Ehrig, M., Hotho, A., Scnizler, B.: Personnalized Information Access in a bibliographic Peer-to-Peer System. In: Staab, S., Stuckenschmidt, H. (eds.) Semantic web and Peer-to-Peer, Decentralized management and exchange of knowledge and information, pp. 144–157. Springer, Heidelberg (2006)
7. Kanawati, R., Malek, M.: Informing the design of shared bookmark systems. In: proceedings of RIAO'2000: Content-based Multimedia information access, Paris (2000)
8. Kanawati, R., Malek, M.A.: multi-agent system for Collaborative Bookmarking. In: proceedings of fourth international Workshop on Agent-oriented Information System (AOIS'02), CEUR , vol. 59, Bologna, pp. 84–97 (July, 2002)
9. Karoui, H., Kanawati, R., Petrucci, L.: COBRAS: Cooperative CBR System for Bibliographical Reference Recommendation. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 76–90. Springer, Heidelberg (2006)
10. Keller, R.M., Wolf, S.R., Chen, J.R., RabinowitzJ., L., Mathe, N.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)
11. Li, W., Vu, Q., Agrawal, D., Hara, Y., Takano, H.: PowerBookmarks: A system for Personnalizable Web Information Organization, Sharing and Management. In: proceedings of the 8th International World Wide Web Conference (WWW'8), Toronto, Canada (May 1999)
12. Lim, J-G.: 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 (1994)
13. Maarek, Y.S., Ben Shaul, I.Z.: Automatically Organizing Bookmarks per Contents. In: Proceedings of the 5th International World Wide Web Conference, Paris (May 6-8, 1996)

14. Malek, M.: Hybrid approaches Integrating Neural Networks and case based reasoning: from Loosely Coupled to Tightly Coupled Models. In: Sankar, K.P., Tharam, S.D., Daniel, S.Y. (eds.) Soft Computing in Case-based Reasoning, pp. 73–94. Springer, Heidelberg (2000)
15. Tempich, C., Staab, S.: Semantic Query Routing in Unstructured Networks Using Social Metaphors. In: Staab, S., Stuckenschmidt, H. (eds.) Semantic web and Peer-to-Peer, Decentralized management and exchange of knowledge and information, pp. 107–123. Springer, Heidelberg (2006)
16. Wittenburg, K., Das, D., Hill W., Stead, L.: Group Asynchronous browsing on the World Wide Web. In: Proceedings of the 6th International Conference on the World Wide Web (WWW'6) (1997)