Recommenders in a Personalized, Collaborative Digital Library Environment

Henri Avancini, Leonardo Candela, and Umberto Straccia

Istituto di Scienza e Tecnologie dell'Informazione – C.N.R.
Via G. Moruzzi,1 I-56124 Pisa, ITALY.
{avancini,candela,straccia}@isti.cnr.it

Number: TR-2004-XX

May 27, 2004

Abstract

We envisage an information source not only as an information resource where users may submit queries to satisfy their daily information need, but also as a collaborative working and meeting space of people sharing common interests. Indeed, we will present a highly personalized environment where not only users may organize (and search into) the information space according to their individual taste and use, but provides advanced features of collaborative work among the users. It is up then to the system to discover interesting properties about the users' interests, relationship between users and user communities and to make recommendations based on preference patterns of the users, which is the main topic of this paper.

ACM Categories and Subject Descriptors: H.3.2 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval - *Information filtering*; H.3.4 [**INFORMATION STORAGE AND RETRIEVAL**]: Systems and Software - *User profiles and alert services*;

Keywords: Digital library, collaboration, personalization, recommendation

1 Introduction

It is widely recognized that *Digital Libraries* (DLs) [15], will play an important role in the next future not merely in terms of the 'controlled' digital information they allow access to, but in terms of the *services* they provide to the information society at large. Informally, DLs can be defined as consisting of collections of information (usually, heterogeneous in content and format), which have associated services delivered to users and user communities using a variety of technologies. The services offered on such information can be various, ranging from content operations to rights management and can be offered to *individuals* as well as to *user communities*.

Even though DLs have evolved rapidly over the past decade, typically, they still are limited to provide a search facility to the digital society at large. Indeed, they are oriented towards a generic user, as they answer queries crudely rather than, w.r.t. learn the long-term requirements of a specific user, *i.e.* the so-called *user profile*, which is, informally, a (machine) representation of the user's needs through which, for instance, an information seeking assistant, should act upon one or more goals based on that profile and autonomously, pursuing the goals posed by the user (even irrespective of whether the user is connected to the system). In practice, users use the same information resource over and over and would benefit from automatization: the time consuming effort that the user put in searching documents and possibly downloading them from the DL is often forgotten and lost. This requires a repetition of the manual labor in searching and browsing to find the documents just like the first time.

As DLs will become more commonplace, there is a need for them to move from being *passive* with little adaptation to their users, to being more *proactive* and *personalized* in offering and tailoring information for individual users. Nowadays, in several DLs some personalization functionalities are provided. Mainly they fall into the category of personalized *alerting services* (*e.g.*, see [5, 13, 14, 22, 26]), *i.e.* services that notify a user (usually, by sending an e-mail), with a list of references to newly available documents in the DLs and deemed as relevant to some of the (manually) user specified topics of interests. Typically, such alerting services are provided by scientific journal publishers. Some other DLs, in addition, support the users in being able to organize their information space they are accessing to according to *their own subjective perspective* (*e.g.*, see [14]). This is important as not necessarily all the information provided by a DL may be of interest to an user, but just some 'slices' of it. Users and communities of users might well profit from being able to organize the information space in a personalized fashion both in terms of restricting the information space in which to search into as well as in terms of organizing it not necessarily in the way a the DL manager thought would be well suited for anyone.

Seldom [1], DLs can also be considered as *collaborative meeting place* of people sharing common interests. Indeed, our vision is that DLs may be viewed as a *common working places* where users may become aware of each other, open communication channels, and exchange information and knowledge with each other or with experts. In fact, usually users and/or communities access a DL in search of some information. This means that it is quite possible that users may have overlapping interests if the information available in a DL matches their expectations, backgrounds, or motivations. Such users might well profit from each other's knowledge by sharing opinions or experiences or offering advice. Some users might enter into long-term relationships and eventually evolve into a community if only they were to become aware of each other. Hence, we are moving from services supporting an individual user towards services supporting *groups* of users as well.

A major service in a personalized and collaborative DL is the *recommendation* of items to a user based not only on preference patterns of the user itself, but also on those of other users. The use of opinions and knowledge of other users to predict the

[1] [12] is an exception.

relevance value of items to be recommended to each user in a community is known as *Collaborative* or *Social Filtering* (*e.g.*, see [6, 4, 17, 19, 25]). These methods are built on the assumption that a good way to find interesting content is to find other users who have similar interests, and then recommend items that those similar users like.

In this paper we first briefly present the CYCLADES system [2] and stress its 'personalization' and alerting features. A major distinction of CYCLADES is that it is indeed a DL environment supporting collaboration and personalization at various level. Users and communities may search, share and organize their information space according to their own personal view and where the system generates recommendation of various types based on user and community profiles. We then extensively present the recommendation algorithms used in CYCLADES. Worth noting is that they not only recommend to users, documents deemed as relevant, but also user, communities and information sources.

The outline of the paper is as follows. In the next section we recall the main features of CYCLADES, while in Section 3 we report our experimental results of the recommendation algorithms adopted within the final version of CYCLADES. Section 4 concludes.

This work concludes a series of works on the subject by the authors [1, 2, 11, 24].

2 A brief overview of CYCLADES

The objective of CYCLADES is to provide an integrated environment for users and groups of users (communities) that want to use, in a highly personalized and flexible way, 'open archives', *i.e.* electronic archives of documents compliant with the Open Archives Initiative [3] (OAI) standard. Informally, the OAI is an agreement between several digital archives providers in order to provide some minimal level of interoperability between them. In particular, the OAI defines an easy-to-implement gathering protocol over HTTP, which give *data providers* (the individual archives) the possibility to make the documents' metadata in their archives externally available. Indeed, the agreement specifies that each document of an archive should posses a *metadata* record describing the documents properties and content. In particular, the format of the metadata records should be DublinCore [4]. The metadata record consists of several attributes describing author, title, abstract, etc. of documents. The protocol allows then to gather these metadata records (in place of the real documents). A link to the 'real' document is also present if the document is accessible. A metadata record may be understood as a statement of existence and short description of a document. The document may be then accessible to a user according to the access policies of the archive, which owns the document. To date, there is a wide range of archives available (more than one hundred registered archives) in terms of its content, forming a quite *heterogeneous and multidisciplinary information space* (*e.g.*, covering the fields of biology, mathematics, computer science, physics, etc.).

[2] http://www.ercim.org/cyclades
[3] http://www.openarchives.org
[4] http://dublincore.org

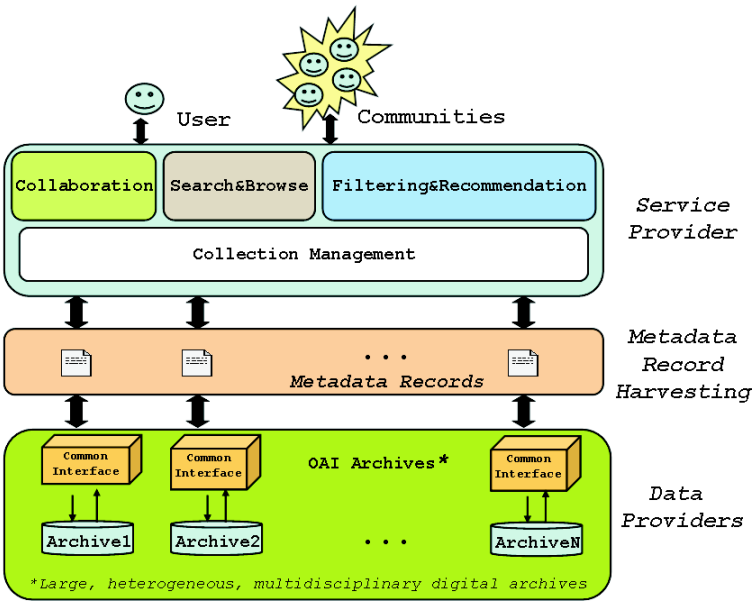


Figure 1: Logical view of CYCLADES functionality.

The availability of the metadata records from the OAI compliant archives makes then it possible for *service providers* to build higher levels of functionality. In this sense, CYCLADES allows the access to the metadata provided by these archives, as it gathers these records and through them provides access to the referenced complete documents (if they exist and their access is allowed). On top of them, CYCLADES acts as an OAI service provider (see Figure 1.) providing functionality for:

1. advanced search in *large, heterogeneous, multidisciplinary digital archives*;
2. collaborative work;
3. information filtering;
4. various forms of information recommendation; and
5. management of records grouped into so-called *collections* (see later on) [5].

These functionality are available in several environments, which are described below.

The main principle underlying CYCLADES is the *folder paradigm* (see Figure 2). That is, users and communities of users may organize the information space into their own folder hierarchy, as *e.g.* may be done with directories in operating systems,

[5]In CYCLADES , there is one collection of metadata records, for each OAI archive. But, additionally, users may define their own sets.

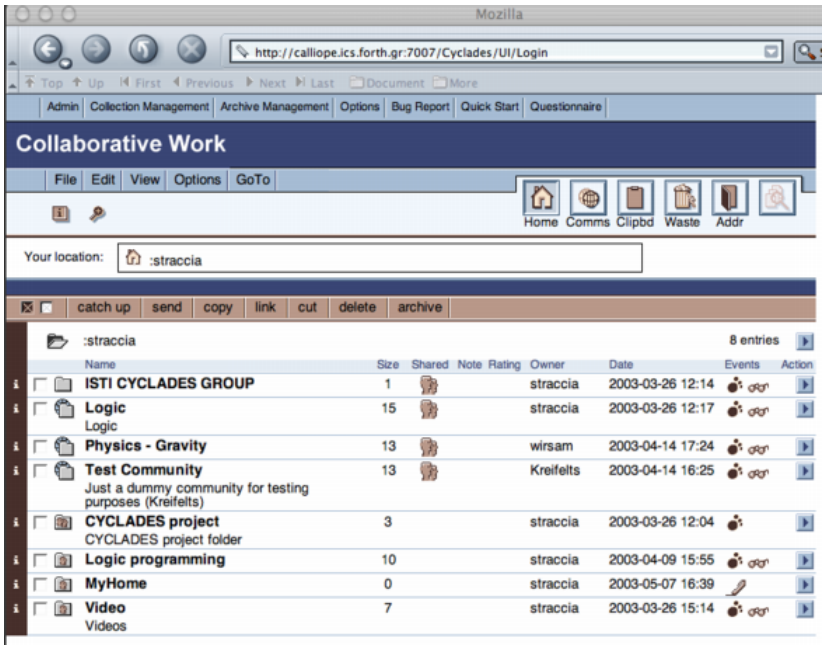


Figure 2: User interface: a user home folder.

bookmark folders in Web browser and folders in e-mail programs. Each folder typically corresponds to one user related subject (or discipline, or field), so that it may be viewed as a thematic and usually semantically related repository of data items.

Collaborative Work Environment. This is an extension of the BSCW environment (Basic Support for Collaborative Work) [3] and provides the folder-based environment. The collaborative work environment allows the management of various data items, such as metadata records, queries, collections, external documents (*i.e.*, documents which can be uploaded by the user to the folder), ratings and annotations.

There are two types of folders: (*i*) *private folders*, *i.e.* a folder owned by one user only. This kind of folder can only be accessed and manipulated by its owner. They are invisible to other users; and (*ii*) *community folders*, which can be accessed and manipulated by all members of the community that owns the folder. Community folders are used to share data items with other users and to build up a common folder hierarchy. Community folders may also contain *discussion forums* where notes may be exchanged in threaded discussions (similar to news groups). For instance, Figure 2, shows the home (top level) folder of a user. It contains several sub-folders. Among them, there are some (shared) folders belonging to communities (created by someone) to which the user joined to, like the 'Physics-Gravity' folder (community), while others are private folders

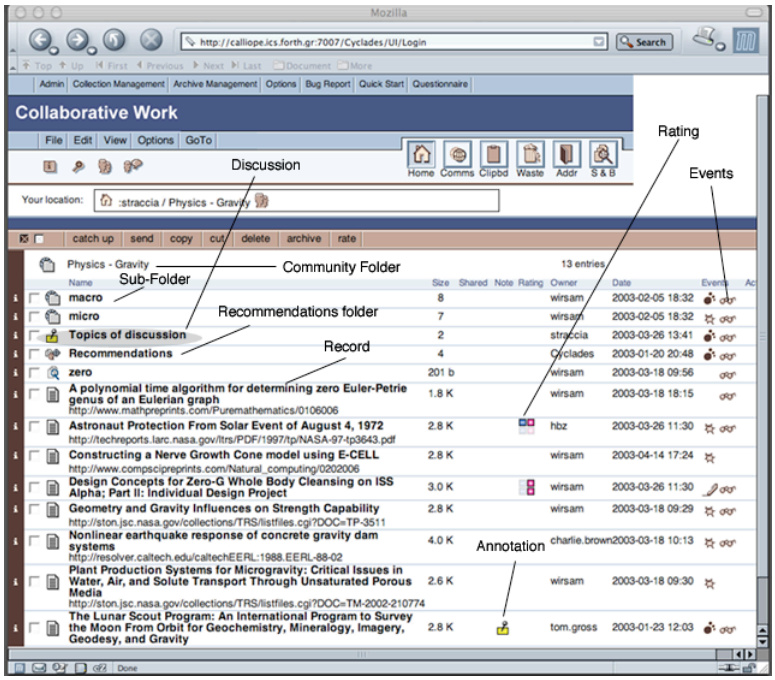


Figure 3: User interface: folder content

and have been created directly by the user, *e.g.* the 'Logic Programming' folder. These folders contain community or user collected OAI records relevant to some topics (*e.g.* gravity and logic programming, respectively). Figure 3 shows the content of a specific user folder, in our case the 'Physics-Gravity' folder of the community of physicists. In order not to lose shared activity in the collaborative DL environment, mutual awareness can be supported through event icons displayed in the environment. Activity reports that are daily received by email are also be possible. Users can also view the list of all existing communities and can join a community directly if this is allowed by the community's policy, or contact the community administrator in order to be invited to the community. In the collaborative work environment, the access policies can be set-up, as well as the notification (alerting) modalities.

Search and Browse Environment. It supports the activity of searching records in the various metadata record collections accessible from within CYCLADES as well as to search into the shared folders or private folders a user owns. Users can issue a query and are allowed to store selected records within their folders and community folders they have access to. Essentially, three types of search are supported:

1. *ad-hoc search*, where a user specifies a query and the system looks for

relevant records within a specified collection;

2. *filtered search* is like the usual ad-hoc search, except that the user specifies, additionally to a query (*e.g.* 'zero'), also a target folder (*e.g.* 'Physics-Gravity'). The goal of the system consist then to find documents not only relevant to the query, but also relevant to the topic of the target folder (in our example, the request is something like 'find records about zero gravity'); and
3. *what's new, on-demand*, where the user specifies a target folder, without specifying a query, and the goal of the system consists in finding all records, relevant to the target folder, which where become available to the system since the last time the user asked for this request. This corresponds roughly to the functionality provided by alerting services, except that the profile is build implicitly from the folder content, and that records are delivered to the user on-demand.

Filtering and Recommendation Environment. It supports the personalized search (*i.e.* "filtered search" and "what's new, on-demand") and provides the recommendations functionalities. All recommendations are specific to a given user folder (topic of interest), *i.e.* they have always to be understood in the context not of the general interests of the user, but of the specific interests (topic) of the user represented by a folder. A user may get recommendations of *metadata records* (suggesting to the user to access to relevant documents), *collections* (suggesting to the user to search within a relevant information space), *users* (suggesting to the user to enter in relationship with a user or give a look to the publicly available documents of the recommended user), and *communities* (suggesting to the user to join the community) issued to users based on user and/or community profiles. For instance, Figure 4 shows the recommendations related to the 'Physics-Gravity' folder, deemed by the system as relevant to this folder.

Collection Management. It manages collections (*i.e.* their definition, creation, and update) of metadata records. Its aim is to allow a dynamic partitioning of the information space according to the users' interests, where to search. Usually, a collection is meant to reflect a topic of interest of a user or a community, *e.g.* the collection of records about 'Information Retrieval'. Informally, a collection specification is the definition of a not materialized view over the information space. It is up to the system to automatically determine the "physical" collections in which to search for relevant records (this is accomplished by means of a technique called *automated source selection* see, *e.g.* [8, 16, 21, 23]).

As pointed out, filtering and recommendation play an important role in making CYCLADES a personalized and collaborative environment. In the following section, we detail the algorithms used for metadata record recommendation, user recommendation and collection recommendation. We also report experimental results of their effectiveness. The recommendation of communities, *i.e.* the suggestion to join a community is realized as for the case of collections. Indeed, a community is identified with

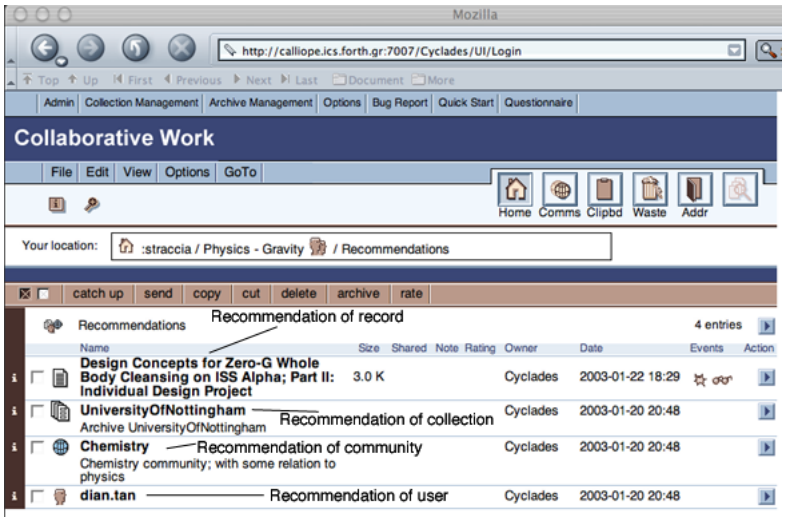


Figure 4: User interface: folder content and recommendations.

its community folder, *i.e.* a set of metadata records. As such, a community can be seen as a collection of records and, thus, the recommendation of a community can be implemented like the recommendation of a collection. Therefore, we do not address it further.

3 Recommendation algorithms and experimental evaluation

We are going to describe our recommendation algorithms and test for their effectiveness. We first introduce some notation, describe the data corpus and then proceed with the description and effectiveness tests of the algorithms for record, user and collection recommendation. For each recommendation type (record, user, collection) we provide the objective, the test set, the algorithm, the evaluation method, and the result analysis.

3.1 Preliminaries

In the following, consider

- a set of users u_k;
- a set of folders F_i (of the users); and
- a set of available metadata records d_j.

For ease, we consider a metadata record as an unstructured piece of plain text (of course, more sophisticated algorithms can be devised by taking into account the metadata structure). Metadata records belong to folders and each user may rate a document within a folder he has access to. A record may belong to multiple folders. With r_{ijk} we indicate the rating value given by a user u_k to record d_j, which is stored in folder F_i. We further assume that whenever a data item d_j belongs to a folder F_i of a user u_k, an *implicit* default rating r_{ijk} is assigned. Indeed, a record belonging to a folder of a user is an implicit indicator of being the record relevant to the user folder. Given a folder F_i and a record d_j belonging F_i, we compute an average rating from all the ratings given to record d_j in folder F_i, *i.e.* we average the ratings r_{ijk} relative to the same record–folder pair (i, j) and indicate it as

$$r_{ij} = \text{mean}_{k \geq 1}\{r_{ijk}\} \ . \tag{1}$$

(see matrix (c) in Table 1). The rating r_{ij} indicates the average rating given by the users to record d_j in folder F_i. This allows us to re-conduct our three dimensional recommendation setting (user-folder-record) to an easier to manage two dimensional setting (folder-record) to which we apply our recommendation algorithms. Of course, in place of the mean, other measures may be applied to compute r_{ij} as well. All records in the folders are indexed according to the well-known vector space model [27]. With

$$d_j = \langle w_{j1}, \ldots, w_{jm} \rangle$$

we indicate its indexed representation, where $0 \leq w_{jk} \leq 1$ is the 'weight' of term (keyword) t_k in the record d_j (see matrix (a) in Table 1).

For each folder, we compute its *folder profile*, which is a machine representation of what a folder is *about*. For a given folder F_i, for ease the folder profile (denoted f_i) of F_i is computed as the *centroid*, or average, of the records belonging to F_i, *i.e.*

$$f_i = (1/|F_i|) \sum_{d_j \in F_i} d_j \ ,$$

and, thus, it is represented as a vector of weighted terms as well, *i.e.* $f_i = \langle w_{i1}, \ldots, w_{im} \rangle$ (see matrix (b) in Table 1). Note that the folder profile does not take into account the ratings associated to the records, but only the records' text content.

Similarly, the *user profile* of a user u (denoted p_u) is built as the centroid of the folder profiles the user has access to and is an indicator of the interests of an user, *i.e.* if $\mathcal{F}_u$ is the set of folders the user u has access to, then

$$p_u = (1/|\mathcal{F}_u|) \sum_{F_i \in \mathcal{F}_u} f_i \ .$$

Like folder profiles, the user profile is represented as a vector of weighted terms as well, *i.e.* $p_u = \langle w_{u1}, \ldots, w_{um} \rangle$.

The data we represent is summarized in Table 1. Matrix (a) represents the records, matrix (b) represents the folder profiles, matrix (c) represents the average ratings associated to the records belonging to folders, while matrix (d) represents the user profiles.

Table 1: (a) The records matrix. (b) The folder profile matrix. (c) The folder-record rating matrix. (d) The user profile matrix.

	...	t_k	...
d_1	...	w_{1k}	...
...	...	...	...
d_j	...	w_{jk}	...
...	...	...	...
d_n	...	w_{nk}	...

(a)

	...	t_k	...
f_1	...	w_{1k}	...
...	...	...	...
f_i	...	w_{ik}	...
...	...	...	...
f_v	...	w_{vk}	...

(b)

	...	d_j	...
F_1	...	r_{1j}	...
...	...	...	...
F_i	...	r_{ij}	...
...	...	...	...
F_v	...	r_{vj}	...

(c)

	...	t_k	...
p_1	...	w_{1k}	...
...	...	...	...
p_u	...	w_{uk}	...
...	...	...	...
p_h	...	w_{hk}	...

(d)

By relying on matrix (a) of Table 1, the correlation (similarity) among two rows establishes a similarity between records. Similarly, in matrix (b), the correlation among two rows establishes a correlation among folder profiles. Of course, the same applies to matrix (d) so that a correlation among users may be computed. Furthermore, as all three matrixes (a), (b) and (d) are term based, we also compute the mixed correlations among them. For instance, we may compute the similarity among folders profiles and user profiles to determine whether the user is interested in the implicit topic described by the folder. All these measures are based on *content* only (no rating are taken into account). The measure used for content correlation among two vectors v_1, v_2 taken from matrixes (a), (b) or (d), denoted $CSim(v_1, v_2)$, is the well-know *cosine*, *i.e.* the scalar product between two row vectors (we assume that the rows are already normalized):

$$CSim(v_1, v_2) = \sum_k w_{1k} \cdot w_{2k} \ .$$

By relying on matrix (c), a correlation among folders can be determined by taking into account the ratings issued by users. This similarity is called *rating similarity* of two folders F_1 and F_2, denoted $RSim(F_1, F_2)$, and is determined using the *Pearson correlation coefficient* [6], *i.e.*

$$RSim(F_1, F_2) = \frac{\sum_j (r_{1j} - \overline{r}_1) \cdot (r_{2j} - \overline{r}_2)}{\sigma_1 \cdot \sigma_2} \ ,$$

where $\overline{r}_i$ is the mean of the ratings $r_{i1} \ldots r_{in}$, and σ_i is their standard deviation.

The combined similarity between two folders is then obtained by taking into account the content similarity (*i.e.*, using $CSim$) and the rating similarity (*i.e.*, using $RSim$). In what follows, the *combined similarity* or simply *similarity*, denoted $Sim(F_1, F_2)$, between two folders F_1 and F_2 will be determined as a linear combination between their content similarity and their rating similarity, *i.e.*

$$Sim(F_1, F_2) = \alpha \cdot CSim(f_1, f_2) + (1 - \alpha) \cdot RSim(F_1, F_2) \ ,$$

where $0 \leq \alpha \leq 1$.

3.2 Corpus

As to date, neither there is yet a significant corpus within the CYCLADES system (build by real users) nor there exists an available corpus from the literature, which fits to our setting, we build a suitable corpus automatically by taking the data from the Internet.

The corpus was selected from the *Open Directory Project* hierarchy (ODP or DMOZ) [6]. ODP is among the largest human-edited directories of the Web. The ODP data includes over 3.8 million sites, about 60,000 editors and over 460,000 categories. ODP powers the core directory services for the Web's largest search engines and portals, *e.g.* Google [7].

Each category in ODP contains a set of Web documents, which have been evaluated by one or more editors for their relevance to the category. Furthermore, to each document within a category, Google assigns a score (using the PageRank [7] algorithm).

We construct our corpus as follows.

1. The set of users is the set editors of ODP.
2. The set of records is the set of documents in ODP.
3. The set of folders is the set of categories in ODP.
4. We consider different environments of 23, 85 and 100 collections of records, respectively. Each set is characterized by a different degree of heterogeneity of the collections (see description below).

To each record d_j in folder F_i, evaluated by user u_k, we set the rating r_{ijk} equal to the PageRank score s_{ij} assigned to record d_j w.r.t. folder F_i. This means that r_{ij}, the average rating over all users rating records d_j in folder F_i, is indeed s_{ij} [8]. We considered all the categories under "Science", together with the involved records and users (1415 folders, 61 users, 18091 records). All the profiles (of the folders and users) has been restricted to the top weighted 100 terms.

The three different collection environments we consider in our tests have been built as follows from the ODP records. In Table 2 we report the characteristics of these three environments of collections.

[6] http://dmoz.org

[7] http://www.google.com

[8] Note that all users rate d_j in F_i equally. But this does not matter us, as in the recommendation algorithm just the mean r_{ij} is required.

Table 2: The experimental environments

Environment	Number Of Collections	Collection size		
		Max	Min	Avg
ScienceI	23	15,722	11	3,027.47
Quasi-random	85	5,660	1	809.67
Random	100	768	621	696.32

ScienceI: each of the 23 collections contains the records that are in the corresponding first-level sub-category of Science into the DMOZ hierarchy. Therefore, the records of a collection share the same topic;

Quasi-random: each of the 85 collections contains a set of records, which have been selected randomly, from a subset of the first-level sub-category of the category Science. In this case, the collections are more heterogeneous than the previous case as the records of a collection belong to a different, but limited, set of categories. That is, a collection is multi-topic;

Random: each of the 100 collections contains a set of records randomly selected from the whole corpus. Therefore, the collections are highly heterogeneous.

3.3 Record recommendation algorithm

Objective. The objective of the record recommendation algorithm is, given a user u and a folder F_t belonging to u (indicating a topic the user is interested in), called the *target folder*, to recommend to F_t (and, thus, to the user) records relevant to the topic represented by F_t.

Test set. To create the test set, we considered the set $\bar{D}$ of all records of the corpus, which belong to at least two folders. For each of these records $d_j \in \bar{D}$ (300 records), we randomly choose a folder F_t in which d_j occurs. The set of chosen folders $\bar{F} = \{F_t\}$ (250 folders), the records d_j and the relative average rating r_{tj} forms the test set, *i.e.* the test set in this case is a set of triples (F_t, d_j, r_{tj}), where we check whether our recommendation algorithm recommends record d_j to the *target* folder F_t and how well the computed recommendation score agrees with the rating r_{tj}.

Algorithm. Our recommendation algorithm follows a four-step schema:

1. select a set $MS(F_t)$ of k-most similar folders to the target folder F_t, according to the similarity measures $xSim$ (we can use either $CSim$, $RSim$ or a combination of both and in this latter case we use $\alpha = 0.5$);

2. from this set of similar folders, determine a pool P_D of candidate records, *i.e.* set of records belonging to the similar folders $F_i \in MS(F_t)$;

3. for each of the records $d_j \in P_D$ compute a recommendation score of d_j to F_t, according to content (cosine between target folder and record) and rating, *i.e.*

$$s^R(F_t, d_j) = \overline{r} + \frac{\sum_{F_i \in MS(F_t)} (r_{ij} - \overline{r}_i) \cdot RSim(F_t, F_i)}{\sum_{F_i \in MS(F_t)} \cdot RSim(F_t, F_i)} \ ,$$

where $\overline{r}$ ($\overline{r}_i$) is the mean of the ratings in the target folder F_t, *i.e.* the mean of the F_t row in matrix (c) (mean of $F_i \in MS(F_t)$ rating row);

4. Recommend to folder F_t records having a positive score.

Evaluation method. For each target folder $F_t \in \bar{F}$, we compute the set of recommended records $\{d_j\}$ and consider their recommendation score $s^R(F_t, d_j)$. We measure the *coverage*, which is the percentage of records correctly recommended to the target folders, *i.e.* the system recommends a record to a folder and in the corpus effectively the record belongs to the test set. We also measure the *accuracy* of the system through the widely used *Mean Absolute Error* (MAE)[18, 20]. It is the mean of the absolute error among the prediction $s^R(F_t, d_j)$ of a document $d_j \in \bar{D}$ and d_j's real rating value r_{tj} in F_j. That is, the MAE is the mean of $|s^R(F_t, d_j) - r_{tj}|$ over $F_t \in \bar{F}$, $d_j \in \bar{D}$. The purpose of the MAE is to determine how far away is the system predicted value from the real rating value of a record.

Result analysis. The results of our experiments are summarized below. We considered two different parameters:

1. the strategy to select the similar folders w.r.t. a target folder; and
2. the impact of varying k, the $k - most$ similar folders threshold.

In case 1., we have three different options: either select similar folders by relying on rating similarity only (using $RSim$), or using content similarity only (using $CSim$), or to use both (Sim). In case 2., k took one of the values $10, 100$ and 200.

Figure 5 reports the coverage and MAE in using $CSim$ (left bar), $RSim$ (center bar) only and in using Sim (right bar) for various k. The combined approach Sim is the best performing one.

Note that all cases reach a stability at $k \geq 100$. For $k = 100$, combined approach perform 49% better than relying on $RSim$ only and 10% better than relying on $CSim$ only. In Figure 6 we report the fraction of records, which have been recommended by using both $RSim$ and $CSim$. For instance, consider the "CSim Coverage" figure. The whole bar reports the coverage of recommended records using $CSim$. The part labeled "CSim & RSim" indicates the amount of records that have been recommended both using $CSim$ and $RSim$, while the label "Only CSim" indicates the amount of records that have been recommended using $CSim$ but *can not* be recommended using $RSim$ and, thus, indicates the added value of using $CSim$. The explanation for the "RSim Coverage" is similar. Interestingly, we can observe that a large part of records, that have been recommended using $CSim$ only, can not be recommended using $RSim$ and vice-versa. This confirms that the combination of both strategies can improve the coverage significantly.

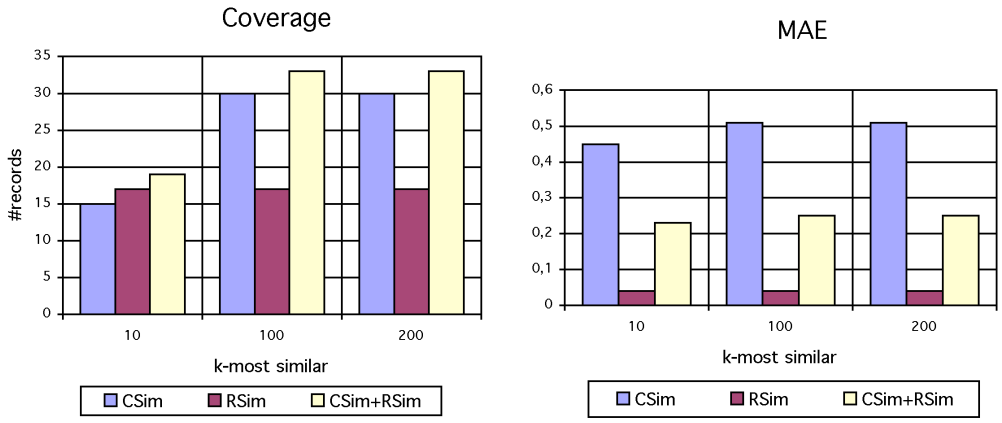


Figure 5: Record recommendation: coverage and accuracy results

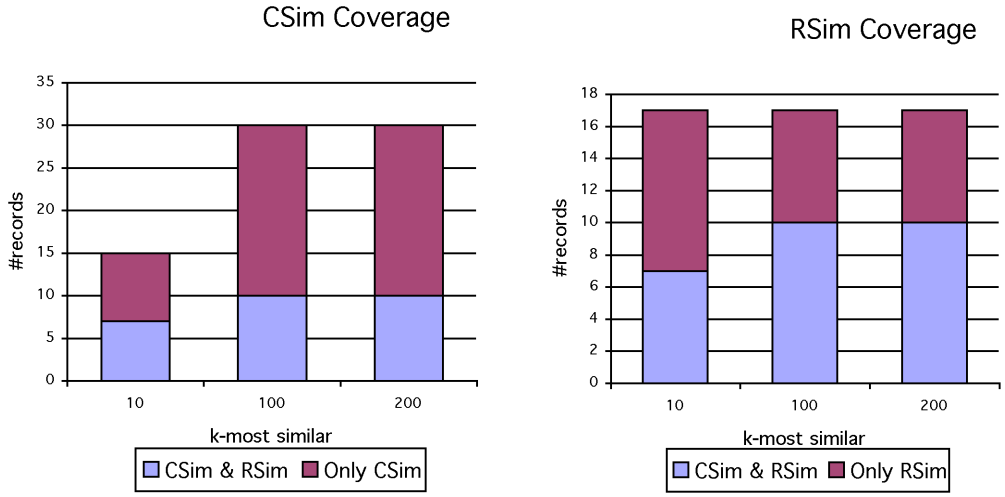


Figure 6: Record recommendation: *RSim* vs. *CSim*.

3.4 User recommendation algorithm

Objective. The goal of the user recommendation algorithm is, given a target folder F_t of user u, to recommend to F_t (and, thus, to user u) those users, which by the system are thought to have overlapping interests with the topic addressed by the folder F_t (and, thus, may be related to user u).

Test set. To create the test set, we considered the set $\bar{U}$ of all users of the corpus, which have access to at least two folders. For each of these users $u_k \in \bar{U}$ (61 users), we randomly choose a folder F_t user u_k has access to. The set of chosen folders $\bar{F} = \{F_t\}$ (49 folders) and the users u_k forms the user test set, *i.e.* the test set in this case is a set of pairs (F_t, u_k), where we check whether our recommendation algorithm recommends user u_k to the target folder F_t.

Algorithm. We have analyzed three different algorithms, with increasing level of effectiveness. The first user recommendation algorithm follows a four-step schema, like for the record recommendation case:

1. select the set $MS(F_t)$ of s-most similar folders to F_t, according to a similarity measures. We can use either $CSim$, $RSim$ or the combination of both (Sim), but in [1, 2] we have already observed that Sim ($\alpha = 0.5$) has better effectiveness, so we use it here as well;

2. from this set of similar folders, determine a pool $P_U(F_t)$ of candidate users to be recommended, *i.e.* let $P_U(F_t)$ be the set of users being owners of the folders in $MS(F_t)$;

3. compute the *recommendation score* for each possible recommendable user, *i.e.* for each user $u_k \in P_U(F_t)$ determine the *user hits factor* (where $F_i \in u_k$ means that folder F_i is accessible by user u_k)

$$h(u_k) = |\{F_i : F_i \in MS(F_t), F_i \in u_k\}|$$

   i.e. the number of folders F_i judged as similar to the target folder F_t belonging to the same user u_k. For each user $u_k \in P_U(F_t)$ the recommendation score $s(F_t, u_k)$ is computed as follows:

$$s(F_t, u_k) = h(u_k) \cdot \sum_{F_i \in MS(F_t), F_i \in u_k} Sim(F_t, F_i) \ ;$$

4. recommend to folder F_t, the top-n ranked users, ranked according to the recommendation score.

The intuition behind Step 3 is that the more a user appears among the owners of the top-s similar folders, the more he is considered as relevant to the target folder.

The second algorithm is a variation of the first one in which Step 3 is replaced with:

3. for each user $u_k \in P_U(F_t)$, consider the profile of u_k, p_{u_k}, and compute the recommendation score as the similarity between the user profile p_{u_k} and the profile f of the target folder F_t, *i.e.*

$$s(F_t, u_k) = CSim(f, p_{u_k}) \ ;$$

The intuition here is to use the user profile of recommendable users $u_k \in P_U(F_t)$ directly in place of the folder profiles of similar folders.

Finally, the third algorithm does not consider the set of similar folders, but just compares the profile of the target folder against all user profiles, using $CSim$. Note that in this way, no ratings are taken into account. Therefore, we remove Step 1, and in Step 2, the pool of candidate users, $P_U(F_t)$, is given by all users known to the system, and Step 3 is as in algorithm 2.

Evaluation method. For each target folder $F_t \in \bar{F}$, we compute the set of recommended users $\{u_k\}$ and ranked them according to their recommendation score $s(F_t, u_k)$. In this rank, we highlight the rank position of user u_k. If the recommendation score is 0, user u_k is ranked 0. We compute *precision* and *recall*, by recommending the top-n users for each target folder, where $n \in \{1, 2, 5, 10\}$. Precision is the fraction between correctly recommended users and the total amount of recommended users, *i.e.*

$$Precision_n = \frac{|CorRec_n|}{49 \cdot n} \ ,$$

where $CorRec_n$ is the set of correctly recommended users among all 49 tests and n is the number of recommended users for each test. Recall, is the fraction between correctly recommended users and the number of test users, *i.e.*

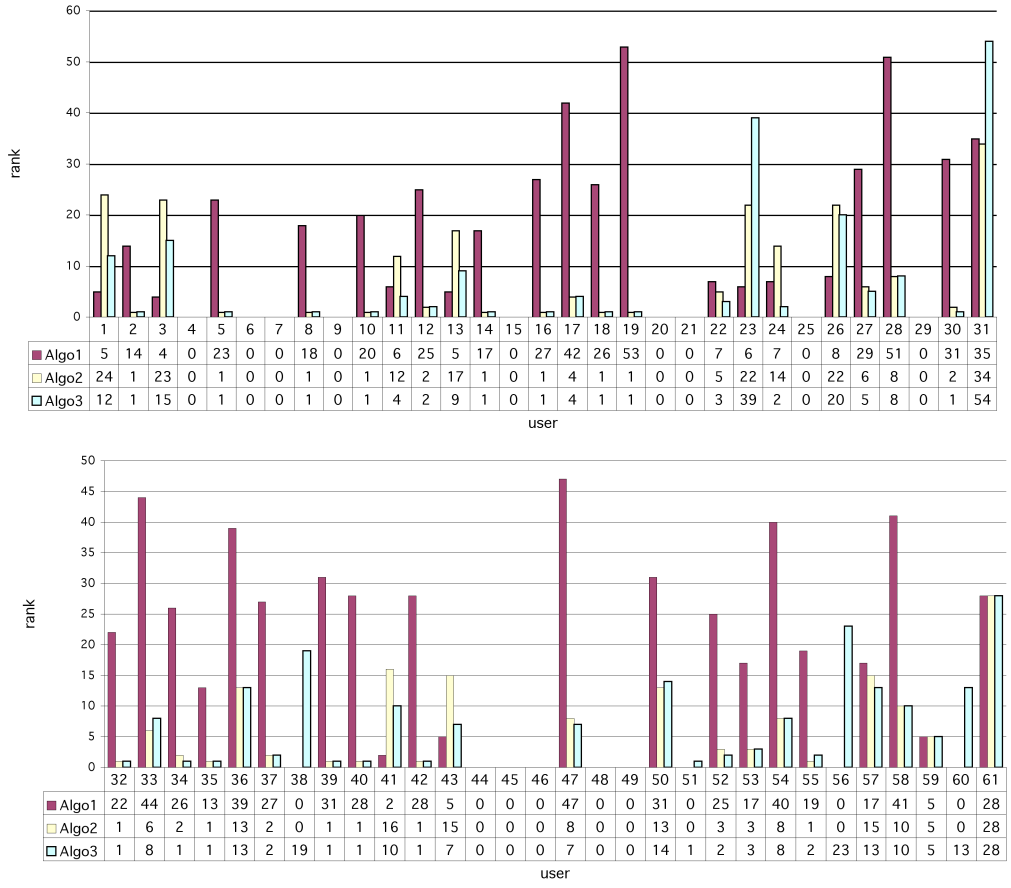
$$Recall_n = \frac{|CorRec_n|}{61} \ .$$

Of course, the more users we recommend (*i.e.* n increases), the more correctly recommended users we have (recall improves), but the less precise we are (precision decreases). We also report the standard measure

$$F1_n = \frac{2 \cdot Precision_n \cdot Recall_n}{Precision_n + Recall_n} \ ,$$

which gives us an estimate of the combination between precision and recall, as well as their average rank.

Result analysis. The results of our experiments are summarized in Figure 7 and Table 3. In Figure 7, for each test pair (F_t, u_k) we report the rank position of user u_k, according to our recommendation algorithms 1, 2 and 3, respectively. It turns out that algorithm 3 performs best, algorithm 2 is second, while the less effective one is algorithm 1. As we can seen, for algorithms 2 and 3, in many cases the correctly



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Algo1	5	14	4	0	23	0	0	18	0	20	6	25	5	17	0	27	42	26	53	0	0	7	6	7	0	8	29	51	0	31	35
Algo2	24	1	23	0	1	0	0	1	0	1	12	2	17	1	0	1	4	1	1	0	0	5	22	14	0	22	6	8	0	2	34
Algo3	12	1	15	0	1	0	0	1	0	1	4	2	9	1	0	1	4	1	1	0	0	3	39	2	0	20	5	8	0	1	54

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
Algo1	22	44	26	13	39	27	0	31	28	2	28	5	0	0	0	47	0	0	31	0	25	17	40	19	0	17	41	5	0	28
Algo2	1	6	2	1	13	2	0	1	1	16	1	15	0	0	0	8	0	0	13	0	3	3	8	1	0	15	10	5	0	28
Algo3	1	8	1	1	13	2	19	1	1	10	1	7	0	0	0	7	0	0	14	1	2	3	8	2	23	13	10	5	13	28

Figure 7: Rank position of correctly recommended users

Table 3: Effectiveness measures

	algorithm	top-1	top-2	top-5	top-10
Precision	1	0.00	0.01	0.02	0.02
	2	0.29	0.18	0.09	0.06
	3	0.33	0.21	0.11	0.07
Recall	1	0.00	0.02	0.10	0.18
	2	0.23	0.30	0.38	0.48
	3	0.26	0.34	0.44	0.57
F1	1	undef.	0.01	0.04	0.04
	2	0.25	0.23	0.15	0.11
	3	0.29	0.26	0.18	0.13

algorithm	average rank
1	23.18
2	8.28
3	8.09

recommended user is highly ranked. It turns out that algorithm 3, which does not rely on the ratings provided by the users, is the most effective one and, interestingly, that recommending just the top ranked user is the most satisfactory compromise between precision and recall, except for algorithm 1, where taking the top ranked user only failed to be the right one in all test cases. Therefore, precision and recall are 0 and, thus, $F1$ is undefined.

3.5 Collection recommendation algorithm

Objective. The objective of the collection recommendation algorithm is, given a target folder F_t of user u, to recommend to F_t those collections that potentially contains records relevant for the topic addressed by the folder itself.

Test set. To create the test set, we considered the 300 folders belonging to three different categories of Science, *i.e.* Agriculture, Anomalies and Alternative Science, and Astronomy. these folders are our target folders F_t, where we check how good our recommendation algorithm is in recommending relevant collections to the folder F_t;

Algorithm. The recommendation algorithm is based on a simple two-step schema:

1. the computation of an approximation of the content of each collection; and
2. the selection of the top-n collections deemed as most relevant to the target folder, relying on the approximations of the collections' content.

The first step is done only once for all collections and consists of the computation of a representation of what a collection is about, *i.e.* the so-called *collection topic* or *language model* of the collection. This data is then used in the second step to compute the *recommendation score* for each collection, *i.e.* a measure of similarity between the folder topic and the collection topic and, thus, establishes the relatedness of the collection to the target folder.

The language model of a collection consists of a list of terms with their term weight information. We rely on the so-called *query-based sampling method* [9], which has been proposed for automatically acquiring statistical information about the content of a collection. A major feature is that it requires only that a collection provides a query facility and access to the records, that are in the result of a query. Informally, the method is an iteration of the following steps 2 and 3:

1. issue a random query to the collection (as start-up);
2. add the select the top-k records to the sample;
3. select randomly a record from the current sample. Select randomly n terms from this record. Build a new query using these terms and issue the query to the collection.

The iteration continues until a stop criterion is satisfied [9]. As a result, a *sample* set of records for each collection has been gathered. This set is our resource description, or approximation of a collection.

Once we have gathered a sample (approximation) of each collection, we are ready to perform collection recommendation. So, let F_t be a target folder and let f_t be its profile (vector of terms t_{tk} with relative weight w_{tk}). The collection recommendation score $G(F_t, C_l)$ for a collection C_l with respect to the folder F_t is defined as follow:

$$G(F_t, C_l) = \frac{\sum_k p(t_{tk}|C_l)}{|f_t|} \ , \tag{2}$$

where $|f_t|$ is the number of non-zero weighted terms in the profile f_t. The *belief* $p(t_{tk}|C_l)$ in C_l, for term t_{tk} appearing in F_t's profile f_t is computed using a variant of the CORI algorithm [10]:

$$p(t_{tk}|C_l) = T_{l,tk} \cdot I_{tk} \cdot w_{tk} \tag{3}$$

$$T_{l,tk} = \frac{df_{l,tk}}{df_{l,tk} + 50 + 150 \cdot \frac{cw_l}{\overline{cw}}} \tag{4}$$

$$I_{tk} = \frac{\log\left(\frac{|C|+0.5}{cf_{tk}}\right)}{\log\left(|C| + 1.0\right)} \tag{5}$$

where:

[9] More details about this technique in our context can be found also in [11]

$df_{l,tk}$	is the number of records in the approximation of C_l containing term t_{tk};
cw_l	is the number of terms in the approximation of C_l;
$\overline{cw}$	is the mean value of cw over the approximation of C_l;
cf_{tk}	is the number of approximated collections containing t_{tk};
$\|C\|$	is the number of the collections.

Finally, given a target folder F_t, all collections C_l of a collection environment are ranked according the their collection recommendation score $G(F_t, C_l)$ and the top-n are recommended to F_t.

Evaluation method. The evaluation phase is rather involved and consists of two parts:

1. the evaluation of the effectiveness of a collection approximation, *i.e.* comparing the learned resource description of a collection with the real resource description for that collection ; and

2. the evaluation of the effectiveness of the collection recommendation.

Concerning the evaluation of the effectiveness of a collection approximation, in accordance with [9], we have used two metrics to evaluate the quality of the approximation: the *ctf ratio* (CTF) to measure the portion of term occurrences in the collection vocabulary (V) that are covered by the terms in the learned vocabulary (V') and the *Spearman Rank Correlation Coefficient* (SRCC) to measure the correspondence between the learned and the actual document frequency information. This metrics are calculated using Equation (6) and (7) below, where ctf_i is the number of times term t_i occurs in the collection C, $\delta(i)$ is the rank difference of a common term $t_i \in V' \cap V$. The two term rankings are based on the learned and the actual document frequency df_i. n is the total number of common terms, *i.e.* $n = |V' \cap V|$.

$$CTF = \frac{\sum_{t_i \in V'} ctf_i}{\sum_{t_i \in V} ctf_i} \tag{6}$$

$$SRCC = 1 - \frac{6}{n^3 - n} \sum_{t_i \in V' \cap V} \delta(i)^2 \ . \tag{7}$$

Concerning the second evaluation point (*i.e.* the effectiveness of collection recommendation) we require some additional steps. Indeed, we neither know a priori, which are the correct collections to be recommended to a target folder, nor we are willing to evaluated it manually. A way to automated effectiveness evaluation of a collection recommendation algorithm is as follows. For each the three collections environment, we have built a collection containing all the records belonging to the environment. This collection is called *baseline collection*. The baseline collection has been indexed. This index is called *baseline index*. For indexing (and retrieval, see later on) we have used the Jakarta Lucene search engine [10]. So, we have three baseline collections and

[10] `http://jakarta.apache.org/lucene`

indexes, one for each environment. We also indexed all collections of all environments, which we call *collection index*.

For each target folder F_t, we consider its profile f_t. Each folder profile represent a query that is matched against a baseline collection of a collection environment. As a result, the retrieval engine give us back a ranked list $rank(F_t)$ (called *baseline rank*) of records belonging to the environment, deemed by the system as relevant to the query, *i.e.* relevant to the folder F_t. Now, we proceed as follows. For each collection environment and for each target folder F_t, we use the recommendation algorithm to compute a *recommendation score* $G(F_t, C_i)$ for each collection C_i of the environment. We select the top-n collections according to their score. Then we match the query f_t against each collection C_i of this list and get a ranked list of records $ret_i(F_t)$. Now we merge these ranked lists into a unique ranked list $ret(F_t)$. The fusion of multiple ranked lists is done by computing, for each record, a normalized global retrieval score, $RSV_g(d_j)$, obtained from the their normalized retrieval score, $RSV'_i(d_j)$, within the ranked lists, *i.e.*

$$RSV_g(d_j) = \frac{RSV'_i(d_j) + 0.4 \cdot RSV'_i(d_j) \cdot G(F_t, C_i)}{1.4} \quad (8)$$

$$RSV'_i(d_j) = \frac{RSV_i(d_j) - min_k(RSV_i(d_k))}{max_k(RSV_i(d_k)) - min_k(RSV_i(d_k))} , \quad (9)$$

where $RSV_i(d)$ is the score of record d in the ranked list $ret_i(F_t)$ (note that a record does appear at most in one ranked list, as there is no record overlapping among the collections). Therefore, for target folder F_t, on one hand we have the baseline rank $rank(F_t)$ which has been computed over the baseline collection (without selecting collections) and on the other hand we have the rank $ret(F_t)$ obtained from the fusion of rank lists *with* collection selection. The effectiveness of the recommendation algorithm is computed as the similarity among the two ranked lists $rank(F_t)$ and $ret(F_t)$ and is based using set-based measures like Precision and Recall defined as follows:

$$Precision = \frac{|rank(F_t) \cap ret(F)|}{|ret(F_t)|} \quad (10)$$

$$Recall = \frac{|rank(F_t) \cap ret(F_t)|}{|rank(F_t)|} . \quad (11)$$

We have also considered the harmonic mean ($F1$-score) of these two values computed as $F1 = 2 \cdot Precision \cdot Recall/(Precision + Recall)$.

As the measure of Precision and Recall does not consider the ordering of the two ranked lists to compare, we have also used the $SRCC$ to measure the correspondence between the two ranked lists, *i.e.* $SRCC = 1 - \frac{6}{n^3-n} \sum \delta(i)^2$, where this time $\delta(i)$ is the rank difference of a record i appearing in both ranks and n is the total number of records common to both ranks.

Result analysis. At first, we report the evaluation of the effectiveness of the sampling method.

Table 4: Collections and their samples characteristics

Environment	Avg Record			Avg Terms		
	Source	Sample	Sample%	Source	Sample	Sample%
ScienceI	3,027.47	279.17	9.22%	10,2375	17,707.2	17.29%
Quasi-random	809.67	238.3	29.43%	46,197.7	18,083.17	39.14%
Random	696.32	295.74	42.47%	48,571.31	27,121.01	55.83%

Table 5: Sample statistics

Environment	CTF			$SRCC$		
	Max	Min	Avg	Max	Min	Avg
ScienceI	98%	74%	87%	97%	53%	80%
Quasi-random	100%	71%	98%	100%	63%	85%
Random	92%	78%	87%	90%	75%	85%

In Table 4 we report the characteristics of the collections and their approximations in terms of number of records gathered and number of terms in the samples, respectively.

In Table 5 we report the results of CTF and $SRCC$ effectiveness metrics. For example, observing the ScienceI case, we can note that acquiring just the 9% of the records of the source we are able to have a very close representation of the content of the collections as we obtain a CTF of about 90% and an $SRCC$ of 80%. Moreover, note that the effectiveness of the approximations are quite independent from the content homogeneity of the collections in the collection environments. But, for instance, by observing the Random environment case, we can note that the amount of records acquired by the sample process is a much greater in percentage than that for the ScienceI case in order to get similar CTF and $SRCC$ values. Essentially, and quite intuitively, the more heterogeneous is a collection with respect to its content, the more records have to be gathered in its sample to reasonably approximate the collection's content.

At second, the evaluation of the effectiveness of the recommendation algorithm. For each target folder, we have evaluated the effect of varying the number of collections to be recommended and the number of records (from 1 to 500) to be considered in each of the two ranked lists $rank(F_t)$ and $ret(F_t)$.

In Figures 8 and 9, we report the results for the ScienceI case. As previously observed, in this case a collection is homogeneous, *i.e.* its records belong to the same topic or set of topics. The recommendation of just the top-1 collection for each target folder produces an high F1 value, about 80%, and an high SRCC value. This means that our algorithm is able to find the most appropriate collection w.r.t. the target folder in case the collection are multitopic. Recommending more than one collections produces a deterioration of the results as the selected collections contains documents less pertinent to the target folder topic. In Figures 10 and 11 we report the results for the Quasi-random case. As previously observed this kind of collections are more heterogeneous than the previous ones, *i.e.* the records of a collection may belong to

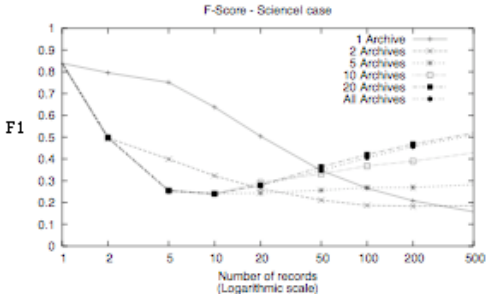


Figure 8: ScienceI case: F1 values.

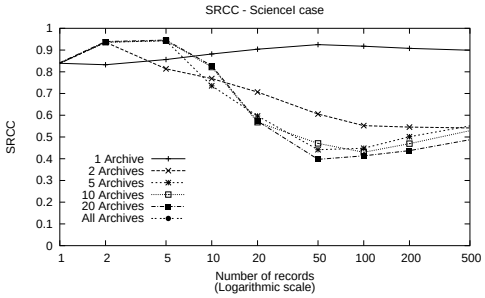


Figure 9: ScienceI case: SRCC values.

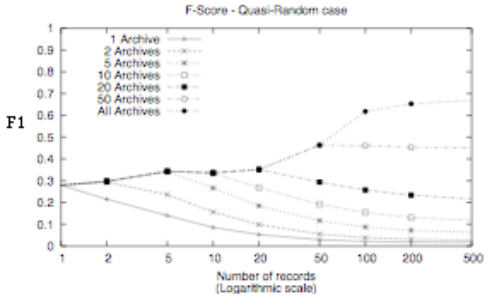


Figure 10: Quasi-random case: F1 values.

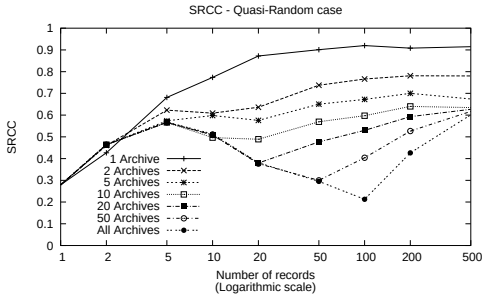


Figure 11: Quasi-random case: $SRCC$ values.

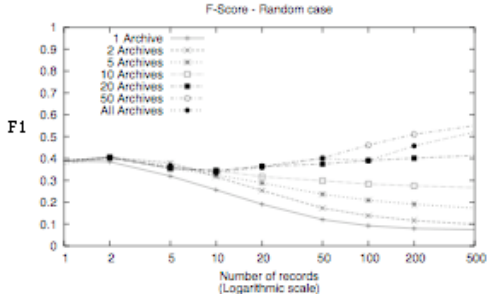


Figure 12: Random case: F-Score values.

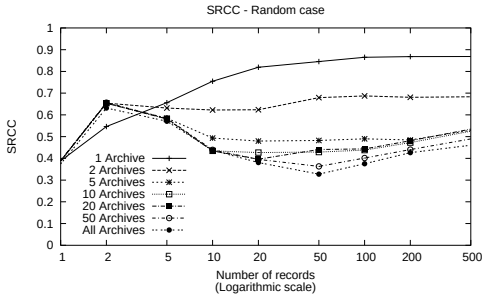


Figure 13: Random case: *SRCC* values.

different but a limited set of topics, and records about a topic are distributed among a limited number of collections. In this case recommending just the top-1 collection to each target folder produces a lower F1 than for the previous case. Moreover, the performance decrease if the number of records to be recommended increases. Recommending more than one collections produces an improvement of the results as the selected collections contain records relevant to the target folder. Concerning the *SRCC* curve we note that increasing the number of records in the rank list, each curve has a decreasing phase and finally it increases. The end of the decreasing phase coincides with the point where the F1 value starts to decrease, which means that we are able to select a minor number of records but the ranked lists are more similar.

In Figures 12 and 13 we report the results for the Random case. In this environment, the collections are highly heterogeneous, *i.e.* the records of a collection can belong to many different topics, and the records of a category are distributed, potentially, among all collections. This is the worst case. We can note that the performance decreases if the number of records in the rank list increases, while its increases if the number of collections to recommend increases. However, our algorithm is still able to find the most appropriate collection w.r.t. the target folder.

4 Conclusions

Since the Web, and consequently the information contained in it, is growing rapidly, every day a huge amount of "new" information is electronically published and new Digital Libraries are available to satisfy the user information needs. We described here a Digital Library environment that is not only an information resource where users may submit queries to get what they are searching for, but also a collaborative working and meeting space. Indeed, users looking within an information resource for relevant data might have overlapping interests, which may turn out to be of reciprocal interest for the users: users might well profit from each other's knowledge by sharing opinions and experiences. As such, we have presented CYCLADES, a personalized collaborative Digital Library environment, in which the user functionality may be organized into four categories: users may (*i*) search for information; (*ii*) organize the information space (according to the "folder paradigm"); (*iii*) collaborate with other

users sharing similar interests; and (*iv*) get recommendations. CYCLADES not only provides recommendation of records, as it usually happens in personalization system dealing with documents, but by taking advantage of the highly collaborative environment, it may recommend also communities, collections and users as well. Particular attention has been paid to the recommendation part and to the experiments showing the effectiveness of the adopted algorithms.

Acknowledgments

This work is funded by the European Community in the context of the CYCLADES project IST-2000-25456, under the Information Societies Technology programme.

References

[1] Henri Avancini and Umberto Straccia. Personalization, collaboration, and recommendation in the digital library environment CYCLADES. In *Proceedings of the IADIS International Conference Applied Computing (AC-04)*, pages 589–596, Lisbon, Portugal, 2004. IADIS Press.

[2] Henri Avancini and Umberto Straccia. User awareness in cyclades: a collaborative and personalized environment for open digital archives. In *Proceedings of the IADIS International Conference on Web Based Communities (WBC-04)*, pages 67–74, Lisbon, Portugal, 2004. IADIS Press.

[3] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor, and G. Woetzel. Basic support for cooperative work on the world wide web. *International Journal of Human Computer Studies*, (46):827–846, 1997.

[4] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proc. 15th International Conf. on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.

[5] Kurt Bollacker, Steve Lawrence, and C. Lee Giles. A system for automatic personalized tracking of scientific literature on the web. In *Digital Libraries 99 - The Fourth ACM Conference on Digital Libraries*, pages 105–113, New York, 1999. ACM Press.

[6] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, Madison, Wisconsin, USA, 1998.

[7] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[8] Jamie Callan. Distributed information retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, Hingham,MA, USA, 2000.

[9] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[10] Jamie Callan, Zhihong Lu, and Bruce W. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR-95)*, pages 21–28, Seattle, WA, 1995.

[11] Leonardo Candela and Umberto Straccia. The personalized, collaborative digital library environment CYCLADES and its collections management. In Jamie Callan, Fabio Crestani, and Mark Sanderson, editors, *Multimedia Distributed Information Retrieval*, number 2924 in Lecture Notes in Computer Science, pages 156–172. Springer Verlag, 2004.

[12] Mariella Di Giacomo, Dan Mahoney, Johan Bollen, Andreas Monroy-Hernandez, and Cesar M. Rouiz Meraz. Mylibrary, a personalization service for digital library environments. In *Proceedings of the Second DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries*, Dublin City University, Ireland, 2001.

[13] Daniel Faensen, Lukas Faulstich, Heinz Schweppe, Annika Hinze, and Alexander Steidinger. Hermes: a notification service for digital libraries. In *ACM/IEEE Joint Conference on Digital Libraries*, pages 373–380, 2001.

[14] Lourdes Fernandez, J. Alfredo Sanchez, and Alberto Garcia. Mibiblio: personal spaces in a digital library universe. In *ACM DL*, pages 232–233, 2000.

[15] Edward A. Fox and Gary Marchionini. Digital libraries: Introduction. *Communications of the ACM*, 44(5):30–32, 2001.

[16] Norbert Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 3(17):229–249, 1999.

[17] David J. Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[18] N. Good, J.B. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, , and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *In Proceedings 1999 Conf. American Association of Artificial Intelligence*, pages 439–446, 1999.

[19] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR-99)*, pages 230–237, Berkeley, CA USA, 1999.

[20] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J.Riedl. Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[21] W Meng, K.L Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. Determining text databases to search in the internet. In *Proc. of the 24th Int. Conf. on Very Large Data Bases (VLDB-98)*, pages 14–25, New York, USA, 1998.

[22] Alexandros Moukas. *Amalthaea*: Information discovery and filtering using a multiagent evolving ecosystem. In *Proceedings Practical Applications of Agents and Multiagent Technology*, London, GB, 1996.

[23] Yves Rasolofo, Faïza Abbaci, and Jacques Savoy. Approaches to collection selection and results merging for distributed information retrieval. In *Proc. of the 10th Int. Conf. on Information and Knowledge Management (CIKM-01)*, 2001.

[24] M. Elena Renda and Umberto Straccia. A personalized collaborative digital library environment. In *5th International Conference on Asian Digital Libraries (ICADL-02)*, number 2555 in Lecture Notes in Computer Science, pages 262–274, Singapore, Republic of Singapore, 2002. Springer-Verlag.

[25] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[26] Luis Mateus Rocha. Talkmine and the adaptive recommendation project. In *ACM DL*, pages 242–243, 1999.

[27] Gerard Salton and J. Michael McGill. *Introduction to Modern Information Retrieval*. Addison Wesley Publ. Co., Reading, Massachussetts, 1983.