A PERSONALIZED INFORMATION SEARCH ASSISTANT

M. Elena Renda
Istituto di Informatica e Telematica del CNR, Via G. Moruzzi 1, Pisa, Italy

Keywords: Personalized systems, Automatic source selection, Information Filtering, Metasearch, Rank fusion.

Abstract: A common characteristic of most of the traditional search and retrieval systems is that they are oriented towards a generic user, often failing to connect people with what they are really looking for. In this paper we present PISA, a Personalized Information Search Assistant, which, rather than relying on the unrealistic assumption that the user will precisely specify what she is really looking for when searching, leverages implicit information about the user’s interests. PISA is a desktop application which provides the user with a highly personalized information space where she can create, manage and organize folders (similarly to email programs), and manage documents retrieved by the system into her folders to best fit her needs. Furthermore, PISA offers different mechanisms to search the Web, and the possibility of personalizing result delivery and visualization. PISA learns user and folder profiles from user’s choices, and uses these profiles to improve retrieval effectiveness in searching by selecting the relevant resources to query and filtering the results accordingly. A working prototype has been developed, tested and evaluated. Preliminary user evaluation and experimental results are very promising, showing that the personalized search environment PISA provides considerably increases effectiveness and user satisfaction in the searching process.

1 INTRODUCTION

Though nowadays more information is easily reachable and in a smaller amount of time than years ago, it is becoming increasingly difficult for individuals to control and effectively seek for relevant information among the information resources available on the Internet. The more users are getting on-line, the more their information needs become complex, the more difficult it becomes to find relevant information in a reasonable amount of time, unless the user exactly knows what to get, from where to get it, and how to get it.

Given the exponential growth in the quantity and complexity of information sources available on the Internet, over the last years much effort has been put into the development of approaches to deal effectively with this complexity: Information Retrieval systems have evolved from a simple concern with the storage and distribution of information, to encompass a broader concern with the transfer of “meaningful information”. In particular, users could benefit from “personalized” services and systems for finding relevant information for their interests in a broad sense, gaining in time, quality of the documents and information retrieved, and satisfied information needs. Tailoring the information and services to match the unique and specific needs of an individual user (Personalization) can be achieved by adapting the presentation and/or the services presented to the user, taking into account the task, background, history, information needs, location, etc., of the user; i.e., the user’s context.

1.1 Motivation

A common characteristic of most of the traditional search and retrieval services is that they are oriented towards a generic user. If the same query is submitted by different users to a typical search engine, it will probably return the same result, regardless of who submitted the query. Another important aspect of current search systems is that they often answer queries crudely rather than, for instance, learning the long-term requirements specific to a given user or, more generally, to a specific information seeking task.

In searching the Web, besides the users with their corresponding information needs, we have other main actors playing a fundamental role: the Web Information Resources. The resources available on the Web may be extremely heterogeneous in two main respects: the topic of the information they provide, and the metadata schema they use to describe the provided information.
The alternative to querying each resource individually has been offered by retrieval systems that provide a unified interface for searching over multiple resources simultaneously, called Metasearch Systems (see, e.g., [Aslam et al., 2005; Meng et al., 2001]). Metasearch systems have to deal with the two aspects of resource heterogeneity, giving users the impression of querying one coherent, homogeneous resource.

Thus, a system for searching and browsing the Web tailored and customized “ad-hoc” to the user must “know”:

1. where to search, by selecting a subset of relevant resources among all those that can be accessed (Automatic Resource Selection) (Huang et al., 2007; Nottelmann and Fuhr, 2003);
2. how to query different resources, by matching the query language used by each of the selected resources (Schema Matching) (Madhavan et al., 2005; Renda and Straccia, 2006);
3. how to combine the retrieved information from diverse resources (Result or Rank Fusion) (Callan et al., 1995; Dwork et al., 2001; Lee, 1997); and
4. how to present the results to the user, according to her preferences (Result Presentation) (Chen and Chue, 2005; Liu and Croft, 2002).

We modeled and developed such a personalized information seeking system (in the following called assistant), which helps users in retrieving actual relevant information from the Web with minimum cost, in terms of both effort and time. In order to assist the user in the information seeking task, the assistant has to know the user and her profile, a representation of her background, interests and needs (User Profiling (Amato and Straccia, 1999)). This profile can be then used by the assistant for finding matchings against content profiles for retrieving relevant information and filtering out the irrelevant ones (Information Filtering or Content-based Filtering) (Belkin and Croft, 1992).

Another orthogonal aspect of personalization we considered when modeling the assistant was the information organization, i.e., supporting the users in the task of organizing the information space they are accessing to, according to their own subjective perspective.

In this paper we present this Personalized Information Search Assistant, called PISA, an environment where the user will not only be able to search/retrieve/be informed about documents relevant to her interests, but she will also be provided with highly personalized tools for organizing documents and information into a personal workspace. The major novelty of PISA is that it combines all the characteristics of an on-line metasearch system with working space organization features in a desktop application, providing the user with a single user point of view personalized search environment. User evaluation and preliminary experimental results are very promising, showing that the personalized search environment PISA provides considerably increases effectiveness and user satisfaction in the searching process.

The paper is organized as follows: the next Section provides an overview of the possible PISA competitors; Section 3 introduces PISA, describing its functionality; Section 4 describes PISA architecture in detail; in Section 5, the evaluation methodology is described and the experimental results are reported; finally, Section 6 concludes, providing an outline for further developing this work.

2 PERSONALIZED SYSTEMS

The personalization task is becoming fundamental in searching and finding relevant information, as the amount of information and providers increases at vertiginous rates. The environments where personalization is being used are databases, newsgroups, discussion lists, electronic journals, search engines, e-commerce Web sites, and so on. The requirement for personalization is well known, for instance, in the context of Digital Libraries (DLs). Some DLs provide simple personalized search functionality, such as providing the so-called alerting services (see, e.g. (Faensen et al., 2001)), i.e., services that notify a user (typically by sending an e-mail) with a list of references to new documents deemed relevant to some of the user topic of interest (manually specified). Other DLs, for instance, give users the possibility to organize their personal information space (see, e.g., (Fernandez et al., 2000)), and collaborate within community of users with similar interests (see, e.g., (Renda and Straccia, 2005)).

Many commercial information filtering systems use the approach of user-defined profiles, used to personalize search results. Other systems, reflecting the desire to place most of the burden of constructing the user profile on the system, rather than on the user, rely on the development of “models” that are collections of good guesses about the user (see, e.g., the PIA system (Albayrak et al., 2005), PENG (Bailie et al., 2006)). These systems are on-line personalized services, which often provide only part of the features PISA has, or require collaborative filtering among the users of similar groups. Differently, the Personalized Information Search Assistant we will present
3 THE ASSISTANT

PISA acts as go-between the user and the information resources, when searching the Web. The main principle underlying the personalized environment we propose is based on the folder paradigm. That is, the user can organize the information space into her own folder hierarchy, using as many folders as she wants, named as she wants, similarly to what happens, e.g., with directories in operating systems, and folders in e-mail programs. In our system, a folder is a holder of documents relevant to the user and, typically, contains semantically related documents. This means that the content of a folder implicitly determines the topic of the folder. For this reason, we associate to each folder its profile, a compact representation of what the folder is about. Thus, folder profiles, which depend on the documents the corresponding folder currently contains, determine the documents that will be retrieved for that folder. The user’s set of folder profiles represents the collection of topics the user is interested in; consequently, the user profile consists of the collection of profiles related to the folders she owns.

3.1 System Functionality

PISA functionality can be logically organized into two main categories (Figure 1): working space organization, and metasearch.

Working Space Organization. The working space organization functionality allows the user to login to the system, manage folders and documents, update profiles, and set up her personal data and system preferences; on the other hand, the assistant, based on the user behaviors, tries to “understand” her interests and automatically generates a “profile” representing the user (the user profile), and a set of profiles representing her interests (the folder profiles). These profiles, along with the user preferences, are then used as filters over the results obtained for the specific user request, in order to deliver only the “right” information, and present the personalized result list in the way that is more suitable for the user. Folder profiles and the user profile are updated from time to time (Scheduled Profile Updating). When a user has considerably changed the content of a folder, she may also request an immediate update (On-demand Profile Updating) of the profile.

Metasearch. The search mechanisms (Figure 2) provided by PISA are essentially of two types:

1. Filtered Search: the user is interested in finding new documents not yet retrieved for the current folder and she is:
   - looking for new documents (Search New) - relevant to the folder - published on the re-
sources after the last search was performed (information maintained, for each folder, by storing the \texttt{SEARCHTIMESTAMP}); or
- looking for new documents related to the folder by providing one or more keywords (Personalized Search).

2. Simple Search: the user does not associate any folder to the keywords she looks for, \emph{i.e.}, she issues a “simple query” like through Web search engines.

The Search New mechanism can be performed On-Demand for a specific folder at user request, or for all the folders the user owns at a scheduled time (Scheduled Search New), according to the settings the user configured in her system preferences.

Filtered Searches may be accomplished in at least two ways: (i) through query expansions techniques (Carpineto et al., 2001), \emph{i.e.}, by expanding the query with significant terms of the folder profile and then submitting the expanded query; or (ii) issuing the query, and then filtering the result list w.r.t. the folder profile (Belkin and Croft, 1992; Callan, 1998).

The latter approach is used in Personalized Search, where the profile is used as a post-filter, \emph{i.e.}, after the results have been retrieved, while in Search New the folder profile is used as a pre-filter, by selecting some of the significant terms of the profile and using them as the query (recall that the user does not provide any keyword). Another important difference between these search mechanisms is the folder–query association: while in the Filtered Searches the user explicitly declares to use the folder profile as a filter, and the folder will be the final repository of the results, in the Simple Search only the user profile can be used, if possible, for filtering the retrieved documents and the repository of the results will be the user \texttt{HOME} folder (folder created by default together with the \texttt{TRASH} folder). It is worth noting that there is always a current folder: if no folder is selected, the current folder is the \texttt{HOME} folder.

The metasearch functionality allows the user to decide what kind of search she wants to perform over the Web; on the other hand, when a search is started either on-demand or at a scheduled time the assistant automatically selects the information resources to query, applies schema matching (if necessary), queries the selected resources, combines the results in a single result list and filters the results, either by means of the folder profile -if the query is associated with a given folder, or by means of the user profile otherwise.

As an example, here we show what happens when the user wants to perform a Personalized Search. The user selects a folder and wants to search for documents relevant to that folder containing the keywords she provides. \texttt{P/SA}:

1. automatically selects the resources relevant to the profile of the selected folder or uses -if any- the resources selected by the user;
2. applies the schema matching for each selected resource;
3. searches the selected resources and combines the result lists into a single ranked list;
4. uses the folder profile for filtering out some of the results;
5. delivers the results to the user into the selected folder according to the user preferences;
6. updates the \texttt{SEARCHTIMESTAMP} for the selected folder.

4 IMPLEMENTATION

\texttt{P/SA} has been entirely developed using the Java Programming language, to guarantee the portability of the application across different platforms. Furthermore, in developing the prototype we took care of its modularity: each component can be easily modified/enriched or substituted with minimal effort. In particular, the prototype is based on the following development environment and libraries:

\begin{itemize}
\item Java Platform, Standard Edition, and the Java Development Kit (JDK), version 6.0\footnote{\url{http://java.sun.com/}};
\item MySQL version 5.0.51 and the MySQL Connector/J JDBC version 3.1.8\footnote{\url{http://www.mysql.com/}};
\item Apache Lucene library version 2.2\footnote{\url{http://lucene.apache.org/}}.
\end{itemize}

The architecture (Figure 3) consists of the \texttt{Graphical User Interface} (GUI); the \texttt{User Database}, for storing user, folders, documents, preferences and profiles data; the \texttt{Profiler}; the \texttt{Source Selector}; the \texttt{Schema Matcher}; the \texttt{Fusion Module}; and the \texttt{Filter}. In the following we describe each \texttt{P/SA} component and corresponding functionality in detail.

4.1 \textbf{Graphical User Interface}

From the user’s perspective, \texttt{P/SA} GUI consists of a main menu and a set of windows and actions allowing the user to personalize the system step by step, via the folder and document management, the filters and the set of preferences she can modify. The application has
A main pull down menu, with each entry of the menu corresponding to a user action. Every action can be also invoked through keyboard shortcuts.

Each component in PISA has a tooltip text, which comes out by moving the mouse over it, for providing instant help to the user.

**Login Window.** The first window the user is presented with PISA is the login window. After logging in, the user will be presented with the main user interface window and she can use the system, until she decides to quit. When the user accesses the assistant for the first time (after registering), the assistant automatically creates the HOME and TRASH folders.

**Main Window.** The main window is composed of three parts: the folder listing panel (on the left), the document listing panels (on the right), and the search panel (at the bottom) (Figure 4).

The folder listing panel is a tree representing the user hierarchical folder structure. By selecting one folder, the user can: (i) have a view of the documents the folder contains; (ii) rename the selected folder; (iii) create a new folder as a child of the selected one; (iv) delete the selected folder; (v) empty the selected folder; (vi) move a folder from an existing parent folder to a new parent folder (by simply moving the folder in the folder tree - drag&drop).

The document listing panel is a table representing the documents contained in the folder. The table has several columns, each one describing an attribute of the document: the NAME, i.e., the title of the document retrieved, the URL, the RESOURCE from which

---

4Forbidden for HOME and TRASH folders.
5Forbidden for TRASH folder.
- when the system periodically updates the folder profiles;
- when the system periodically search for new documents for each folder owned by the user;
- how the system notifies new documents (the options are: a pop-up window, a sound, or nothing; the default setting is: no event);
- how the system ranks the new documents found (the options are: by score, by date, by resource, or no preference; the default setting is: by score). 

Both the Preferences and the Personal Settings windows can be accessed by the user both via the pull down keyboard shortcut.

**Warning Windows.** These windows are used by the system to warn the user on an invalid or unsuccessful operation.

**Confirm Windows.** These windows are used by the system to ask the user to confirm some actions, like, e.g., quitting the application.

**Acknowledgment Windows.** These windows are used by the system to acknowledge the user of the outcome of an action she started.

### 4.2 User Database

For each user, PtSA locally creates and maintains a database with several tables, and provides methods for creating, reading and updating them. The **User Database** has been realized with the MySQL open source database. The connection and interactions with the database has been realized with the MySQL Connector/J, a native Java driver that converts JDBC (Java Database Connectivity) calls into the network protocol used by the MySQL database.

When the user first register herself to PtSA, the system automatically creates (i) the **Preferences Table**, for storing the system preferences (set by the user or provided by default by the system); (ii) the **Settings Table**, for storing the personal information provided by the user; (iii) the **Folders Table**, for storing all the information related to the folders owned by the user; (iv) the **Documents Table**, for storing all the information related to the documents already retrieved for the user; (v) the **Profiles Table**, for storing the user profile and the folder profile of each folder owned by the user.

### 4.3 Profiler

The Profiler's task is to create user and folder profiles, and update them either on-demand or at a scheduled time. In the following we will describe how to build and maintain these profiles by adopting the approach proposed in (Renda and Straccia, 2005).

Let's denote by \( t_k, d_j, \) and \( F_i \) a text term, a document, and a folder, respectively. Following the well-known vector space model, each document \( d_j \) in a folder \( F_i \) is represented as a vector of weights:

\[
d_j = (w_{j1}, \ldots, w_{jm})
\]

where \( 0 \leq w_{jk} \leq 1 \) corresponds to the “importance value” of term \( t_k \) in document \( d_j \), and \( m \) is the total number of terms occurring in at least one document saved in the folder (Salton and McGill, 1983). The folder profile \( f_i \) for folder \( F_i \) is computed as the centroid (average) of the documents belonging to \( F_i \), i.e.: 

\[
f_i = \frac{1}{|F_i|} \sum_{d_j \in F_i} d_j .
\]

This means that the profile of \( F_i \) may be seen as a data item itself (Belkin and Croft, 1992) and, thus, is represented as a vector of weighted terms as well: 

\[
f_i = (w_{i1}, \ldots, w_{im})
\]

The profile \( p_u \) of the user \( u \) is built as the centroid of the user’s folder profiles, i.e., if \( F_u \) is the set of folders belonging to the user \( u \), then:
As for folder profiles, the profile \( p_u \) of user \( u \) is represented as a vector of weighted terms as well: 
\[
p_u = < w_{u1}, \ldots, w_{um} >.
\]

Besides the folder and user profiles, the Profiler is also responsible for the personal data the user provided (if any) and the system preferences she set.

### 4.4 Source Selector

**Automatic Resource Selection** is based on the assumption of having a significant set of documents available from each information resource (see, e.g., (Callan, 2000)). Usually, these documents are obtained by issuing random queries to the resource (information resource sampling, see, e.g., (Callan and Connell, 2001)). This allows to compute an approximation of the content of each information resource, i.e., a representation of what an information resource is about (information resource topic or language model of the information resource).

As a result, a sample set of documents for each information resource is gathered. This set is the resource description or approximation of the information resource. This data is then used in the next step to compute the resource score for each information resource, i.e., a measure of the relevance of a given resource to a given query. In the following we describe how P/SA computes the resource goodness for automatic resource selection by using an adapted version of the CORI (Callan, 2000; Callan et al., 1995) resource selection method.

Consider query \( q = \{ v_1, \ldots, v_q \} \). For each resource \( \mathcal{R} \in \mathcal{R} \), we associate the resource score, or simply the goodness, \( G(q, \mathcal{R}) \), which indicates the relevance of resource \( \mathcal{R} \) to the query \( q \). Informally, a resource is more relevant if its approximation, computed by query-based sampling, contains many terms related to the original query. However, if a query term occurs in many resources, this term is not a good one to discriminate between relevant and not relevant resources. The weighting scheme is:
\[
G(q, \mathcal{R}) = \sum_{v_k \in q} p(v_k | \mathcal{R}) \frac{|q|}{|v_k|},
\]
where \( |q| \) is the number of terms in \( q \). The belief \( p(v_k | \mathcal{R}) \) in \( \mathcal{R} \), for value \( v_k \in q \), is computed using the CORI algorithm (Callan, 2000; Callan et al., 1995):
\[
p(v_k | \mathcal{R}) = T_{i,k} \cdot I_k \cdot w_k
\]
\[
T_{i,k} = \frac{d_{f,i,k}}{d_{f,i,k} + 50 + 150 \cdot \frac{c_{f,i}}{c_{f,k}}}
\]
\[
I_k = \log \left( \frac{|\mathcal{R}| + 0.5}{c_{f,k}} \right) \log (|\mathcal{R}| + 1.0)
\]
where:
- \( w_k \) is the weight of the term \( v_k \) in the query;
- \( d_{f,i,k} \) is the number of documents in the approximation of \( \mathcal{R} \) with value \( v_k \);
- \( c_{f,i} \) is the number of values in the approximation of \( \mathcal{R} \);
- \( c_{f,k} \) is the mean value of all the \( c_{f,i} \); and
- \( |\mathcal{R}| \) is the number of the resources.

In the above formulae, \( T_{i,k} \) indicates the number of documents that contain the term \( v_k \) in the resource \( \mathcal{R} \). As \( c_{f,i} \) denotes the number of resources in which the term \( v_k \) occurs, called resource frequency. \( I_k \) is defined in terms of \( c_{f,i} \) inverse resource frequency: the higher \( c_{f,k} \) the smaller \( I_k \), reflecting the intuition that the more a term occurs among the resources the less it is a discriminating term. The belief \( p(v_k | \mathcal{R}) \) combines these two measures.

Finally, given the query \( q \), all information resources \( \mathcal{R} \in \mathcal{R} \) are ranked according to their resource relevance value \( G(q, \mathcal{R}) \) and the top-\( n \) are selected as the most relevant ones.

### 4.5 Schema Matching

Given a user query \( q = \{ A_1 = v_1, \ldots, A_q = v_q \} \), written with a specific schema \( T \), called target or global schema, and a resource \( R \) with its own schema \( S \), called source schema, the Schema Matching problem (Dhamankar et al., 2004; Renda and Straccia, 2006) can be defined as the problem of transforming each attribute \( A_T \in T \) of the query in the correct attribute \( A_S \in S \), in order to submit the query to \( R \).

P/SA relies on a simple and effective method to automatically learn schema mappings proposed in (Renda and Straccia, 2006). It is based on a reformulation of the CORI resource selection framework presented in the previous Section. Renda and Straccia (Renda and Straccia, 2006, page 1079) state than, “similarly to the resource selection problem, where we have to automatically identify the most relevant libraries w.r.t. a given query, in the schema matching problem we have to identify, for each target attribute, the most relevant source attribute w.r.t. a given structured query”. Given a resource \( S \) and its metadata...
schema with attributes $S_1, ..., S_n$, the resource selection task can be reformulated in the schema matching problem as follows: given an attribute-value pair $A_i = v_i$, with $A_i$ being an attribute of the target schema $T$, select among all the attributes $S_j$ those which are most relevant to the attribute $A_i$ given its value $v_i$ and map $A_i$ to the most relevant attribute.

Let $R_k \in \mathcal{R}$ be a selected resource. The problem is to find out how to match the attribute-value pairs $A_i = v_i \in q$ (over the target schema) into one or more attribute-value pairs $A_k = v_k$, where $A_k$ is an attribute of the (source) schema of the selected resource $R_k$. Now consider the resource $R_k$ and the documents $r_1, ..., r_t$ of the approximation of $R_k$ $\text{Approx}(R_k)$ (computed by query-based sampling). Each document $r_i \in \text{Approx}(R_k)$ is a set of attribute-value pairs $r_s = \{A_{k1} = v_{k1}, ..., A_{k_j} = v_{k_j}\}$.

From $\text{Approx}(R_k)$, we make a projection on each attribute, i.e., for each attribute $A_k$ of the source schema we build a new set of documents:

$$C_k = \bigcup_{r_s \in \text{Approx}(R_k)} \{r | r := \{A_{k_j} = v_{k_j}\}, A_{k_j} = v_{k_j} \in r_s\}.$$  

The idea proposed in (Renda and Straccia, 2006) is that each projection $C_{k1}, ..., C_{kb_k}$ can be seen as a new library, and CORI can be applied to select which of these new resources is the most relevant for each attribute-value pairs $A_i = v_i$ of the query $q$ (see (Renda and Straccia, 2006) for more details).

### 4.6 Rank Fusion

In PISA, we adopted the rank-based method called CombMNZ, considered as the best ranking fusion method (Lee, 1997; Renda and Straccia, 2003). CombMNZ combination function heavily weights common documents among the rankings, based on the fact that different search engines return similar sets of relevant documents but retrieve different sets of non-relevant documents.

Given a set of $n$ rankings $R = \{r_1, ..., r_n\}$, denote with $\hat{\tau}$ the fused ranking (or fused rank list), which is the result of a rank fusion method applied to the rank lists in $R$. To determine $\hat{\tau}$, it is necessary to determine the fused score $s^\hat{\tau}(i)$ for each item $i \in U$, being $U = \bigcup_{r \in R, i \in \tau}(i)$, and order $\hat{\tau}$ according to decreasing values of $s^\hat{\tau}$. In linear combination ranking fusion methods, the fused score $s^\hat{\tau}(i)$ of an item $i \in U$ is defined as follow:

$$s^\hat{\tau}(i) = h(i, R)^\gamma \cdot \sum_{\tau \in R} \alpha_\tau \cdot w^\tau(i),$$  

where (i) all the rank lists $\tau \in R$ have been normalised according to the same normalization method; (ii) $y \in \{0, 1\}$ indicates whether hits are counted or not; and (iii) $\sum_{\tau \in R} \alpha_\tau = 1$ where $\alpha_\tau \geq 0$ indicates the priority of the ranking $\tau$. In (Renda and Straccia, 2003) the authors report experimental results on comparing several rank-based and score-based fusion methods. According to the results reported in that paper, in PISA: (i) each rank list $\tau \in R$ has been normalised and the normalised weight $w^\tau(i)$ of an item $i \in \tau$ has been computed according to the rank normalization method:

$$w^\tau(i) = 1 - \frac{|\tau(i) - 1|}{|\tau|};$$  

(iii) $y = 0$, i.e., hits have not been counted; (iii) $\alpha_\tau = 1/|R|$, i.e., all rank lists have the same priority.

### 4.7 Filter

When the ranked results are available, the Filter role is to filter out some of the results. In particular, if the search issued was the Personalized Search, the Filter has to compare each document w.r.t. the folder profile. Recall that each document $d_j$ is represented as a vector of weights $d_j = (w_{j1}, ..., w_{jm})$, where $0 \leq w_{jk} \leq 1$ corresponds to the “importance value” of term $t_k$ in document $d_j$ (Table 1), and that each profile is represented as a vector of weighted terms as well, i.e., $f_i = (w_{i1}, ..., w_{im})$ (Table 2).

<table>
<thead>
<tr>
<th>$f_i$</th>
<th>...</th>
<th>$l_k$</th>
<th>...</th>
<th>$l_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>$w_{11}$</td>
<td>...</td>
<td>$w_{1k}$</td>
<td>...</td>
</tr>
<tr>
<td>$d_2$</td>
<td>$w_{21}$</td>
<td>...</td>
<td>$w_{2k}$</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$d_n$</td>
<td>$w_{n1}$</td>
<td>...</td>
<td>$w_{nk}$</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t_1$</th>
<th>...</th>
<th>$l_k$</th>
<th>...</th>
<th>$l_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$w_{11}$</td>
<td>...</td>
<td>$w_{1k}$</td>
<td>...</td>
</tr>
<tr>
<td>$f_2$</td>
<td>$w_{21}$</td>
<td>...</td>
<td>$w_{2k}$</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$f_n$</td>
<td>$w_{n1}$</td>
<td>...</td>
<td>$w_{nk}$</td>
<td>...</td>
</tr>
</tbody>
</table>

In order to compute the content similarity $sim_{ij}$ between the folder profile $f_i$ and the document $d_j$, we compute the well-know cosine metric, i.e., the scalar product between two row vectors, and select only those documents with $sim_{ij} > 0$.

Furthermore, the Filter will deliver up to the maximum number of documents, as requested by the user, and visualize them according to the user settings, as set in the System Preferences Window.
5 USER EVALUATION

In order to provide a preliminary evaluation of PISA usability, we asked 10 users, after a short presentation of the functionality, to use the system and test the GUI. The users first highlighted that such a personalized system is safer to use locally, in terms of privacy (i.e., they did not like the idea of being profiled on the server side or by on-line services). All the users reported that PISA resemblance with a common email program helped them to quickly understand how several GUI components work. The GUI has been classified as intuitive and robust.

To evaluate PISA effectiveness in providing personalized services, we asked the users to create a certain number of folders, populate them with “pertinent” documents, update the correspondent profiles, and issue a number of queries ranging from 1 to 10 for each profile. They created 30 different profiles, and issued a total number of 150 queries. The returned results have been scrutinized and classified by the users as either relevant or irrelevant for the corresponding profile, and the precision performance metric (which, we recall, is defined as the ratio of the number of relevant documents to the total number of retrieved documents) has been evaluated. The maximum number of returned query results has been set to 10. In order to evaluate the benefits of PISA personalized search mechanisms, we asked the user to run the same set of queries without a profile, when they first accessed the system, with empty HOME folder and issuing a simple query (i.e., with no profile, no automatic source selection, no filtering).

Data Sets. On-line web information resources periodically modify their interfaces, so that the wrappers to their result pages have to be maintained constantly up-to-date. In order to avoid spending time in such a tedious activity and concentrate on the personalization evaluation, we decided to download the content of some resources and implement a “static” interface to these local resources. For this purpose, we implemented an indexing engine for locally storing a certain number of information resources.

The INDEXER has been implemented taking advantage of the Lucene libraries, which provide Java-based indexing and search technology, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities. In the indexing process, we have analyzed the individual documents and their content, split into terms, applied stemming, and eliminated stopwords. In the retrieval phase, Lucene libraries allow us to get back statistical information on the resources, such as the frequencies of the individual terms at the field level and at the document level, and the resource size.

We have locally downloaded and indexed 8 resources for a total of about 45,000 searchable documents:

1. BIBDB, containing more than 5000 BibTeX entries about information retrieval and related areas;
2. DUBibDB, containing almost 3463 documents with bibliographic data from the Uni Duisburg University BibDB;
3. HCI, containing 26381 documents with bibliographic data from the Human-Computer Interaction (HCI) Bibliography;
4. DC, containing 6276 OAI documents in Dublin Core format;
5. ETDMS, containing 200 OAI electronic theses;
6. RFC1807, containing 467 OAI documents in RFC1807 format;
7. WGA, containing 265 documents from the European Web Gallery of Art 6;
8. NGA, containing 864 documents from the American National Gallery of Art 7, Washington, DC.

Part of these resource collections have been provided by INEX - Initiative for the Evaluation of XML Retrieval 8. In particular, DUBibDB and HCI collections are part of the INEX Heterogenous Collection Track 2006.

For the profiling and filtering tasks, we computed term weights of the documents by applying the well known $tf.idf$ term weighting model (first introduced in ( Sparck Jones, 1972)). The term frequency $tf_{ij}$ of term $t_i$ in document $d_j$ is defined as:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \cdot$$ (12)

where $n_{ij}$ is the number of occurrences of the considered term $t_i$ in document $d_j$, and the denominator is the sum of the number of occurrences of all terms in document $d_j$. The inverse document frequency $idf_i$ is a measure of the general importance of the term $t_i$ in the corpus of documents $D$ and is defined as:

$$idf_i = \log \frac{|D|}{|\{d \in D : t_i \in d \}|},$$ (13)

where $|D|$ is the total number of documents in the corpus, and the denominator is the document frequency of term $t_i$, i.e., the number of documents where the term $t_i$ occurs: $df_i = |\{d \in D : t_i \in d \}|$.

6http://www.wga.hu
7http://www.nga.gov
8http://inex.is.informatik.uni-duisburg.de
A high weight in $tf \cdot idf$ is reached by terms with a high term frequency in the given document and a low document frequency in the whole collection of documents. Thus this model is a good discriminant of common terms.

Results. The average and variance of precision for the sets of queries submitted are reported in Table 3. As seen from the Table, PISA is very effective in improving precision, which is almost doubled w.r.t. the case of no personalization. In particular, PISA resulted very effective in: (i) filtering out irrelevant results; and (ii) delivering relevant results in presence of very general queries. The effectiveness of PISA in discarding irrelevant results can be deduced by Table 3, which reports the average (and variance) of the number of returned documents in case of personalized and non-personalized queries (we recall that the maximum number of returned documents was set to 10 in both cases). As seen from the Table, the average number of returned documents dropped from 9.79 without personalization to 8.23 with personalization, with higher variance in the latter case. As for (ii), we mention a specific query a user highlighted (several similar queries displayed the same behavior): the query “model” (an intendedly very general query) had precision improved from 0 to 0.7 when executed in the “database” folder, w.r.t. the case of no personalization.

Table 3: PISA experimental results.

<table>
<thead>
<tr>
<th>Precision</th>
<th>No. of Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>No Profile</td>
</tr>
<tr>
<td>Average</td>
<td>0.72</td>
</tr>
<tr>
<td>Variance</td>
<td>0.08</td>
</tr>
</tbody>
</table>

6 CONCLUSIONS

In this paper we presented PISA - a Personalized Information Search Assistant, a desktop application which provides the user with a highly personalized information space where she can create, manage and organize folders, and manage documents retrieved by the system into her folders to best fit her needs. Furthermore, PISA offers different mechanisms to search the Web, and the possibility of personalizing the result delivery and visualization. PISA learns user and folder profiles from user’s choices, and these profiles are then used to improve retrieval effectiveness in searching, by selecting the relevant resources to query and filtering the results accordingly. Preliminary user evaluation and experimental results are very promising, showing that the personalized search environment PISA provides considerably increased effectiveness and user satisfaction in the searching process.

The PISA prototype has been developed pursuing the goal of realizing modularity, so that each component can be easily modified or substituted with minimal effort. We are currently working to extend PISA by including more sophisticated result presentation techniques. Suppose the documents retrieved are considered not relevant by the user, it could be useful not to entirely download the documents. The assistant could highlight important passages within the documents, presenting the user only with the “best” document passage (Passage Retrieval) (Liu and Croft, 2002), or summarize the documents, presenting the user only with the document summary (Summarization) (Chen and Chue, 2005). After analyzing the passages or the summaries, the user can decide whether it is worth downloading the documents and save it in her information space. Furthermore, we plan to investigate different ways of modeling the user, the documents and the corpus (as proposed, for instance, in (Teevan et al., 2005) and references therein), in order to further improve search effectiveness.

ACKNOWLEDGEMENTS

We wish to thank all the people who volunteered to test, debug and evaluate PISA. A special acknowledgment to P. S., whose suggestions and inestimable support helping us to improve this work.

REFERENCES


