Journal of Universal Computer Science, vol. 3, no. 8 (1997), 1007-1021
submitted: 1/6/97, accepted: 10/8/97, appeared: 28/8/97 © Springer Pub. Co.

A Framework for Filtering News and Managing Distributed Data

Gianni Amati
(Fondazione Ugo Bordoni, Italy
gba@fub.it)

Daniela D'Aloisi
(Fondazione Ugo Bordoni, Italy
dany@fub.it)

Vittorio Giannini
(Fondazione Ugo Bordoni, Italy
gvittori@fub.it)

Flavio Ubaldini
(Università di Roma 'La Sapienza', Italy
flavio@dis.uniroma1.it)

Abstract: With the development and diffusion of the Internet worldwide connection, a large amount of information is available to the users. Methods of information filtering and fetching are then required. This paper presents two approaches. The first concerns the information filtering system ProFile based on an adaptation of the generalized probabilistic model of information retrieval. ProFile filters the netnews and uses a scale of 11 predefined values of relevance. ProFile allows the user to update on–line the profile and to check the discrepancy between the assessment and the prediction of relevance of the system. The second concerns ABIS, an intelligent agent for supporting users in filtering data from distributed and heterogeneous archives and repositories. ABIS minimizes user's effort in selecting the huge amount of available documents. The filtering engine memorizes both user preferences and past situations. ABIS compares documents with the past situations and finds the similarity scores on the basis of a memory-based reasoning approach.

Key Words: information filtering, information gathering, internet, agent-based systems, probabilistic model, memory-based reasoning
Category: H.3, I.2

1 Introduction

New information services (e.g., news services, electronic mail) deal with a variety of processes concerning the acquisition and the delivery of information. Users may receive large amounts of information in electronic form, so that methods of *Information Filtering* (IF) are required to select the incoming documents. In addition, a large class of users can access huge amount of data in wide-world localized archives and repositories, which generally consist of heterogeneous collections of information, e.g., text, sound, picture, etc. There exist a lot of repositories that one could inspect, though only a restricted number of them contain relevant information. A major problem is also the multiplicity of data formats, for which several indexing tools have been developed. Although there are many available index and search facilities, such as Gopher, WAIS, Glimpse, Altavista, Lycos etc., different access and storage methods are used. Many information systems and software agents [Etzioni and Weld, 1994, Borghoff and Schlichter, 1996, Callan et al., 1995, Balabanovic and Shoham, 1995, Dent et al., 1992] have been developed to carry out intelligent tasks which help the user to exploit and use these facilities.

As for the INQUERY system [Callan et al., 1995], a ranking of document collections is presented in order to determine the best collections containing pieces of relevant information. In [Borghoff and Schlichter, 1996] the Constraint–Based Knowledge Broker is used to select information for a specific community interest from a wide number of information repositories.

The Internet Softbot (for Software Robot) in [Etzioni and Weld, 1994] uses a Unix shell and the WWW to access different resources on the network. The Softbot is not an information retrieval agent but a personal assistant able to help the user in accomplishing specific network tasks. The agent is an integrated and goal–oriented interface with planning capabilities, able to find solutions for different unexpected situations. A typical learning agent is WebWatcher [Dent et al., 1992], an information seeking assistant that actively leads the user to interesting WWW pages. The learning model is cooperative and history–based.

Besides these network systems in recent years a large number of information filtering systems have been proposed. Information Filtering and Information Retrieval have been described as two faces of the same coin [Belkin and Croft, 1992], because many of the underline issues are the same. One application area that has been heavily targeted is news filtering [Kilander, 1995]. Examples are NewsWeeder [Lang, 1995] and SIFT [Yan and Garcia-Molina, 1995] .

We present a methodology for dealing with information filtering and gathering from heterogeneous sources. We first introduce an IF system, ProFile (PRObabilistic FILtEring), which filters the stream of documents of Usenet (the netnews) and selectively distributes documents to multiple users with several classes of interests. The fast learning and adapting capabilities of ProFile allows for effectively performing information filtering and dissemination. ProFile learns the user's information need by using his relevance feedback. The adaptive model of ProFile is on–line updated. This model is an application of the generalized probabilistic model of *Information Retrieval* (IR) [Amati and van Rijsbergen, 1995, Amati et al., 1996b, Amati et al., 1996a, Amati et al., 1997b, Amati et al., 1997a]. In the second part of the paper we present ABIS (Agent Based Information System) for accessing different site types in a user-transparent way and for selecting several document types, e.g. text, e-mail, etc. The filtering engine is based on a user's profile managed with memory-based reasoning techniques and it is endowed with the autonomous capabilities of the software agents [Etzioni, 1995].

ABIS is one of the agent living in an environment designed to furnish qualified support to a user in several types of daily activity [Cesta and D'Aloisi, 1996]. Each agent in the environment has the same basic architecture and communicates with the same language, KQML (Knowledge and Query Manipulation Language). Agents are specialized in different human-computer interaction activities [D'Aloisi and Giannini, 1995, Brancaleoni et al., 1997]. The system that has mostly influenced ABIS is described in [Maes and Kozierok, 1993]. This is not an Internet agent, but a meeting scheduling agent system that assists one or more users in managing their agendas. It has a cooperative model since it learns from both the users and their agents. The relevant aspect is that the system tries to predict the user behavior by applying a memory-based algorithm.

The two filtering systems are ruled by different inferential engines. The first is focused on a single task, i.e. it uses relevance feedback values for selecting on-line textual information. This relevance feedback is adaptive and only the last evaluated documents modify the user profile. The second filtering model is memory-based, hence the past history is taken again into account in order to modify the user profile.

The two systems can be easily integrated: at present the approaches have been separately used in order to evaluate at what extent their different theoretical presupposition contributes to different aspects of the IF problem.

2 Selective Dissemination of Information with ProFile

Document filtering, also known as *Selective Dissemination of Information*, has a long history, most of it based on the unranked Boolean retrieval model of information retrieval (IR) [van Rijsbergen, 1979]. Recently, the term *Information Filtering* (IF) has started being used in place of the old style document filtering, to emphasize the possibility of selectively distributing multimedia information. In the context of this paper we will use this term too, since the technique here presented can also be used to perform multimedia document filtering and dissemination.

Much of the past research in IF has been based on the assumption that effective IR techniques were also effective IF techniques. Many of the IF approaches proposed at the TREC conferences, for example, were based on past successful IR approaches. This view has been criticised recently by Callan [Callan, 1996] and by the proposer of the TREC–5 Filtering track [Harman, 1996]. The idea is that alternative techniques to IR are required in order to design effective IF systems. In particular, IF requires more sophisticated techniques of learning than IR, since it is important to be able to model the user information need with the most efficient use of the information the user provides. An IF system that would require a long and painful training cannot be considered effective despite its filtering performance. The most effective IF system is the one that requires short training to perform reasonably well and that can be easily tuned by the user in an interactive way. Our approach takes into account this feature as a relevant parameter of the whole relevance feedback process and learning.

In [Section 2.3] we evaluate the performance of the learning model when little training data is provided. We show the important effect of using negative data in the relevance feedback (negative feedback), that is using the information provided by documents the user indicated as non–relevant. In IR the use of negative data in relevance feedback has been received with contrasting views. Salton considered it positively [Salton and McGill, 1983], while other researchers considered it dangerous [Aalbersberg, 1992]. Dunlop [Dunlop, 1997] observed that the vector space model [Salton, 1971], the term addition model [Harman, 1992] and the probabilistic model [Harper and van Rijsbergen, 1978, van Rijsbergen, 1979] behave differently for negative feedback. We intend to prove that the model makes an effective use of negative data in relevance feedback and that the presence of negative data speeds up the learning of the parameters of an IF system.

2.1 ProFile

The *ProFile* (PRObabilistic FILtEring) system has been developed at Fondazione Ugo Bordoni in Rome (Italy) in 1996 and has been in used since then by many researchers of that institution for filtering the Usenet News [Amati et al., 1996a, Amati et al., 1997b, Amati et al., 1997a].
Despite being born with the purpose of filtering netnews, e.g. USENET, ProFile can be adapted to filter any incoming stream of information, like email, newswires, or newspaper articles.

In ProFile each user may define a number of conceptual classes to classify the filtered documents: each class has its own profile. IF systems have two ways for assigning a document to a conceptual class. The first one consists of ranking documents according to their similarity with the profiles of conceptual classes. A document is then assigned to the conceptual class with the highest level of similarity. This technique is appropriate when conceptual classes cover the set of all possible documents. Differently, another technique consists in defining a relation to be satisfied by each couple class–document. If the document satisfies the relation, then it is selected for that class, otherwise it is discarded. If a document satisfies relations with more than one class, then it is either classified into all classes or one is chosen (an arbitrary one or the one with the strongest relation, if that can be quantified). The model used by ProFile follows this second approach by exploiting semantic information theory [Hintikka and Suppes, 1970] and decision theory.

ProFile operates according to the following steps:

1. *Definition of the conceptual classes.* The user defines a set of conceptual classes in which he wants to filter and classify the incoming stream of documents. ProFile requires from the user a set of keywords for an approximate initial description of each conceptual class.
2. *Training phase.* The initial description of the user interests is used as a query by the FIFT service (Fub Information Filtering Tool), a customized version of SIFT, a filtering system developed at Stanford. FIFT filters out of the document collection a set of documents that will be used as the "training set". The user go through the documents of the training set and assigns them relevance values with respect to each conceptual class. The relevance values are chosen from a scale of eleven values of interests (from 0 to 10). The user does not need to go through all the documents retrieved. The number of documents used in the training phase constitutes the *training data.*
3. *Filtering phase.* The user decides to activate the filtering phase when he believes that the definition of the conceptual classes are accurate enough. The filtering phase is made up of two sub-phases:
   (a) *Filtering.* ProFile filters the documents and delivers to the appropriate user's conceptual class. The user can see the filtered documents classified into his personal conceptual classes.
   (b) *Tuning.* The user can modify the profiles providing additional information. This can be achieved by giving relevance values to the filtered documents in the same way it is done in the training phase. This phase can be repeated as many times as the user wants.

2.2 The model of ProFile

In ProFile a document is assumed to be represented by a set of terms (phrases, indexes, words or lexical units) in the set T. We denote by Ω the set of documents which have been examined by the user up to the current instant of time.
We recall that the retrieval function RSJ of the probabilistic model [Robertson and Sparck–Jones, 1976, van Rijsbergen, 1979] is defined as:

$$g(d) = \sum_{i=1}^{n} x_i log \frac{\frac{r^i}{n_R - r^i}}{\frac{n^i - r^i}{N - n_R - n^i + r^i}} + K \qquad (1)$$

where x_i is 1 or 0 according to whether the term t_i occurs or not in the document d, N is the number of documents of Ω, n^i of which include the term t_i, n_R is the number of relevant documents of Ω, r^i of which include the term t_i. K is assumed to be the same for all documents d.
The assumptions of the (binary independence) probabilistic retrieval model are the following:

a) the terms are stochastically independent,

b) the term indexing in a document is binary (restricted to 0 and 1, irrespective of the frequency with which a term occurs in a document),

c) the term distribution in the set of relevant retrieved items is the same as the distribution in the complete set of relevant items,

d) all non–retrieved items can be treated as non–relevant,

e) the relevance feedback values are binary,

f) documents are in one–to–one correspondence with their syntactical representation.

The learning model of ProFile [Amati and van Rijsbergen, 1995] extends conditions b) and e) to non-binary values. A full exploitation of frequency and relevance values are thus allowed. ProFile does not also use the hypothesis d).The new assumption is that all non relevant documents must be effectively evaluated non relevant by the user in the training data.

A learning theory for IF is a triple $< \Omega, \mathcal{A}, R >$. $\mathcal{A}$ is the power set of Ω, namely the set of all subsets of Ω. R is a measure defined by the user starting from the mutually exclusive elementary events, that is the documents d of Ω. This function is lifted from the elementary events to all the events e of the space $\mathcal{A}$ by using the additivity axiom, i.e. $R(e) = \sum_{d \in e} R(d)$.

R corresponds to the subjective measure of relevance that the user assigns on the event space of Ω. Its form is a scale of relevance weights $R(d)$, with $0 \leq R(d) \leq 1$, arbitrarily generated by the user. In ProFile, for example, the user can assign to any retrieved document a value, $R(d)$, that belongs to a scale of 11 degree of relevance. This values are naturally mapped to the interval $[0, 1]$. The dual measure of *non-relevance*, $\neg R(d) = 1 - R(d)$, is also defined.

Functions defined from Ω to the set of real numbers are called *random variables*. In ProFile model a random variable is associated to each term $t \in T$. With a little abuse of language instead of using the notation ξ_t for denoting the random variable associated to t, we use t itself. Given a document $d \in \Omega$, the value $t(d)$ of the random variable t may be either the term frequency in the document d, or the binary weighting (which takes the values 1 and 0 according to the occurrence or not of the term t in the document d respectively), or the *idf* weighting (defined as $idf(t) = -log(n/N)$, where n is the number of documents in which t occurs and N is the number of documents in the collection [Salton and McGill, 1983]).

The random variables are organized in a matrix $\langle t(d) \rangle_{d \in \Omega, t \in T}$. The rows are vector representations $\langle t(d) \rangle_{t \in T}$ of the documents d, while the columns are the random variables $t \in T$.

We can define the *conditional expectation* of a discrete random variable t with respect to the measure R as:

$$E_R(t) = \frac{\sum_{d \in \Omega} t(d) R(d)}{R(\Omega)} \tag{2}$$

Note that if $0 \leq t(d) \leq 1$ then $0 \leq E_R(t) \leq 1$.

When the system must decide whether a term is relevant or not on the basis of the expected measures of relevance and non-relevance of documents, an error can occur and then a loss is produced. To make this decision the system computes the *expected monetary value* of decision theory:

$$EMV(t) = \lambda_1 E_R(t) - \lambda_2 E_{\neg R}(t) \tag{3}$$

where λ_1 and λ_2 are positive, λ_1 is the "gain" when t is relevant to the user and λ_2 is the "loss" when t is not relevant to the user. "t is relevant" whenever $EMV(t) > 0$, which is equivalent to $w(t) > 0$, where

$$w(t) = log \frac{\lambda_1 * E_R(t)}{\lambda_2 * E_{\neg R}(t)} \tag{4}$$

The λ's in the equation (4) are computed as follows. t has the "a priori" relevance value $E_R(t)$. Suppose that t is relevant to the user information need. If "t is relevant", then the user gains the amount of information of non-relevance of t, denoted by $Inf_{\neg R}(t)$. Similarly, λ_2 is the amount of information of relevance of t, that is $Inf_R(t)$. In information theory, if e is an event with probability $p(e)$, the amount of information is taken to be inversely proportional to its probability, namely $-log\ p(e)$. Hintikka [Hintikka and Suppes, 1970] instead suggests to use as a measure of information of an event e with probability $p(e)$ the relative number of alternatives that the event excludes, and it can be formalized by the formula $Inf_p(e) = 1 - p(e)$ (in particular certain events are not informative at all). In our model we need to assign the amount of information to random variables instead to events. In a similar way, and observing that the expectations do not go beyond the value 1, we may define the amount of information as $Inf_R(t) =_{def} 1 - E_R(t)$. If we denote $\neg t = 1 - t$, we get $Inf_{\neg R}(t) = 1 - E_{\neg R}(t) = E_{\neg R}(\neg t)$ and $Inf_R(t) = 1 - E_R(t) = E_R(\neg t)$.
Substituting the values of the λ's into (4), we obtain

$$w(t) = log \frac{E_{\neg R}(\neg t) * E_R(t)}{E_R(\neg t) * E_{\neg R}(t)} \tag{5}$$

The vector $\langle w(t) \rangle_{t \in T}$ in ProFile is the weighted description of the user's profile. It is easy to show that from the formula (5) we easily derive the retrieval function of the binary probabilistic retrieval (1) in the following hypotheses:

1. R is the counting measure for the relevance of documents i.e. $R(d)$ takes the value 0 or 1;
2. $\langle t(d) \rangle_{d \in \Omega, t \in T}$ is the *counting document-term matrix*, that is:

$$t(d) = \begin{cases} 1, & \text{if the i-th term occurrs in d;} \\ 0, & \text{otherwise.} \end{cases}$$

A discussion and a comparison with several models of learning can be found in [Amati et al., 1996b].

In ProFile the user may define an arbitrary number of *conceptual classes* $C_1, C_2, \ldots C_m$. Then the probabilistic model $< \Omega, \mathcal{A}, R_C >$, as described above, can be applied to each class C.

The vector of all weights $w^C = \langle w^C(t) \rangle_{t \in T}$, as defined by Equation (5), will be matched with the new document vector x by a similarity function SIM. In ProFile we use the normalized inner product:

$$SIM(x, w^C) = \frac{x \cdot w^C}{\|x\| \cdot \|w^C\|} \tag{6}$$

Whenever a new document is evaluated by the user, we have to update the entire vector of weights w^C. This requires the updating of the expectation values, $E_R(t)$, $E_R(\neg t)$, $E_{\neg R}(t)$ and $E_{\neg R}(\neg t)$, for all the terms $t \in T$. This is an easy task since expectations are linear functions. We need to store $(1 + |T|) \cdot m$ global parameters, that is the values $R_C(\Omega)$ and $E_{R_C}(t)$. Then relations among the new and the old values are ruled by simple transition equations.

2.3 Comparison with the binary probabilistic model and evaluation of Profile

In the previous section we have already pointed out that our model can be reduced to the the binary probabilistic model with particular relevance measure and frequency analysis. An important difference between the two models is the assumption on the set of non–relevant documents. In the binary probabilistic model the retrieval function is generally applied under the hypothesis that the set of *non–relevant documents* can be estimated by the set of documents *not known to be relevant*. In this context the negative feedback provides very little additional information since the set of non–relevant documents is the vast majority of the document collection. In contrast with the classical probabilistic model, we consider only the set of documents evaluated by the user as a sample of the entire document collection. With this hypothesis, the action of marking a document as non–relevant has a larger effect on future retrievals. We observe that learning is faster by exploiting both positive and negative feedback. Furthermore, when learning is restricted to only highest frequent terms in the training data, we obtain a further improvement in the performance.

The conclusion is that the behavior of the model of ProFile under negative feedback is different from that of the models examined by Dunlop [Dunlop, 1997]. ProFile makes an effective use of negative feedback which even speeds up the learning.

To measure the effectiveness of ProFile we used the two classical valuation values used for the traditional IR systems: recall and precision. *Recall* (R) is the ratio between the number of retrieved documents that are relevant to a query and the number of all relevant documents in the collection. *Precision* (P) is the ratio between the number of retrieved documents that are relevant to a query and the number of all retrieved documents. The performance of ProFile learning model is here evaluated when little training data is provided (see [Fig. 1]), since we observed that a typical user wants to activate the filtering phase after only 20 or 30 examined documents. We also observed 70% precision and 5.4% recall, 59% precision and 8.9% recall, 51.4% precision and 14.2% recall, 31.2% precision and 27.3% recall after retrieving 10, 20, 40 and 80 documents respectively. Other results on ProFile are reported in [Amati et al., 1997b, Amati et al., 1997a]

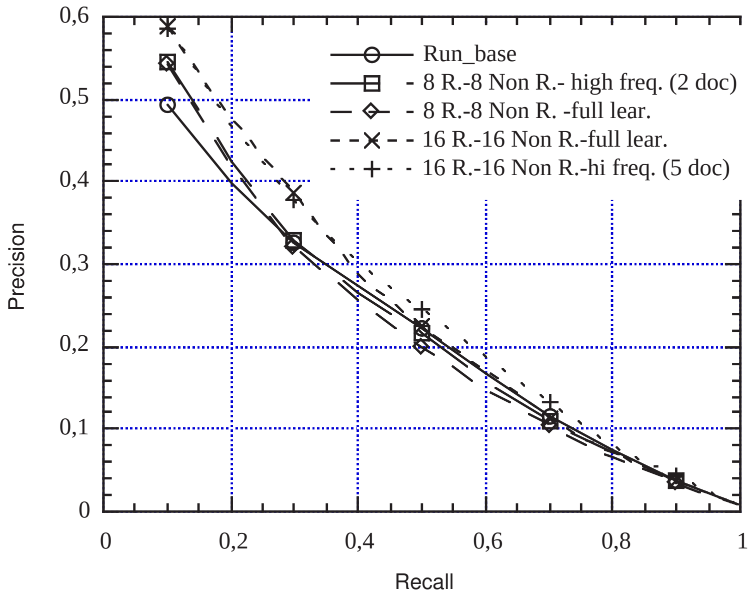

Figure 1: ProFile: performance by a training set of a balanced number of relevant and non relevant documents (8 relevant and 8 non relevant, 16 relevant and 16 non relevant). For low values of recall ProFile is insensitive with respect to cutting off the lowest frequent terms.

The collection we used is the *TREC–5 B* [Harman, 1996] a subset of the collection used in the experiments done in 1996 in the context of the TREC 5 initiative. The collection is made of 3 years (1990–92) of selected full text articles of the Wall Street Journal. The total number of documents (articles) in the collection is about 75.000. Each document is, on average, about 550 words in length. The size of the collection is about 260*Mbyte*. We also used a set of

50 already prepared queries (or topics, as they are called in TREC) with the corresponding set of relevant documents that were used for the training and for the evaluation.

3 ABIS

ABIS (Agent Based Information System) has been designed to assist users in retrieving information in repositories, archives and databases accessible through the network and to disengage as much as possible the user from knowing too many things, such as what is available on the network, how the data are structured and organized, whether new entries and sites are available, where the repositories are located, what retrieval services are at disposal or what access languages and modalities are used.

ABIS follows the software agent metaphor [Etzioni, 1995, CACM, 1994]. An agent is an intelligent entity designed to act on behalf of a user. ABIS is not only an intelligent support program, but an entity capable of acting autonomously, facing unexpected events and cultivating a trustful relationship with the user.

ABIS's main goal is to overcome those problems we encounter with currently used search engines, that is the problem of resizing the number of filtered documents. Besides this activity, ABIS is able to predict the appropriate action the user would fulfill in a particular situation, e.g. *store in a folder, delete, print,* etc. The user–agent interaction builds, maintains an updates both the user profile and the representation of past situations. Different from the ProFile system, where the feedback consists in the user's relevance assessments, ABIS directly learns from user's response and actions.

The system supports two search modalities: the *query mode* and the *surfing mode*. In the first mode, ABIS seeks for specific documents that satisfy a user's request from a collection of archives, whose construction is explained below. In the surfing mode, the agent autonomously navigates the network looking for relevant documents. In this search the agent is driven by a $Preference\ Profile$ that describes the current user interest. In both modalities, the search takes into account the *Situation Set*, where the whole interaction history is memorized and further used for predicting the user's choices. The user interacts with the agent through a Web-based interface that consists of several frames: initialization and setting of the user profile, query formulation, document browsing, request for explanations and suggestions.

The architecture of the agent reflects some basic choices adopted in a larger project whose goal is the development of an integrated environment in which a personal agent manages and cooperates with several other expert agents [Cesta and D'Aloisi, 1996, Brancaleoni et al., 1997].

The agent model consists of three components each specialized in a different activity: communication, reasoning and execution. The communication component is devoted to interact with the user and the external world. The reasoning part is in charge of maintaining the knowledge bases of the agent, providing reasoning and problem solving competence. The executive component is the operative arm of the agent: it can employ external resources or pick up actions from a repertoire. This component consists of Harvest [Hardy et al., 1995] and three modules devoted to the three main high-level tasks of the agent: the surfing mode, the query mode and the local services provider.

Harvest is a powerful public-domain program that offers "an integrated set of tools to gather, extract, organize, search, cache and replicate relevant information across the Internet". It can manage any type of resource and consists of two main components, the *gatherer* and the *broker*. The gatherer first collects the data across the network by utilizing any possible access method (FTP, Gopher, Wais, News, HTTP, etc) and then builds an internal representation of the data by using the SOIF format (Summary Object Interchange Format). In the SOIF construction phase the relevant attributes of a document are extracted. For example, the attributes of a technical report are title, author, date, keywords, abstract, etc. The documents are finally stored in local databases. The broker task is retrieving documents from these databases.

Harvest's capability of representing documents with attributes allows for sofisticated retrieval. Instead of a "blind" search through the entire document, it is possible to select parts of the document saving time and maximizing the probability of a correct match.

The similarity between two documents is obtained first by evaluating the similarity between equal (or semantically consistent) attributes and then by extracting a global score. This match is more effective and precise and reduces in general the number of the retrieved documents.

The ABIS–user interaction starts with the initialization phase in which the user sets the preferences. There are three main sets of preferences: the $Constraint$ Set, the $Personal\ List$ and the $Preference\ Profile$.

The $Constraint\ Set\ \mathcal{CS}$ maintains a description of the user's general preferences concerning the modality of dispatching, the search type, the archive types, the maximum number of documents to be retrieved, the way of dealing with a document, etc.

The $Personal\ List$ contains the sites (URLs) with more relevant documents and this list is used for starting the search.

The $Preference\ Profile$ describes the real user interest. At the beginning the user fills in several forms by specifying keywords for different attributes. The attributes consist in the set $\mathcal{A}_t$ of all the fields whereby Harvest summarizes the documents. Boolean operators are used to link parts of the $Preference\ Profile$. Also compound words are allowed. The $Preference\ Profile$ and the $Personal$ $List$ are used to initialize Harvest.

According to the information given by the user, ABIS builds a first model of the user and initializes the operative components. All these structures are part of the user profile $\wp$.

3.1 Information Representation and User Profile

In ABIS all the relevant entities, i.e., documents, query and $Preference\ Profile$ are represented according to a unique data model. As a consequence, the operations of comparison, filtering and prediction are easier and more efficiently performed. In fact, ABIS refines the SOIF format according to the recognized type: each type presents a different summarization due to its internal decomposition. The SOIF format of a document is characterized by a proper set of attributes, where each attribute contains a significant description of the original document. Different types of objects may share the same attribute names but their semantics is often different, while different attribute names can refer to similar content.

A document d is represented as a list $d = (< a_{d_1}, v_{d_1} >, \ldots, < a_{d_n}, v_{d_n} >)$ of $< attribute, value >$ pairs. The attribute a_{d_i} belongs to the set $\mathcal{A}_t$ of possible attributes. Each v_{d_i} depends on a_{d_i} and is a set of terms of the form $v_{d_i} = (t^1_{v_{d_i}}, \ldots, t^m_{v_{d_i}})$. Each v_{d_i} can be seen as a document itself: an actual document is therefore decomposed into smaller parts v_{d_i} each of which is a specialized sub-representation of the original document.

ABIS supports *structured queries* in which it is possible to specify values for different attributes connected by the Boolean operators *and, or, not.* The form of the query is the following:

$$\begin{aligned} q &= q_u | q \wedge q | q \vee q | \neg q \\ q_u &= (< a_{q_1}, v_{q_1} >, \ldots, < a_{q_n}, v_{q_n} >) \end{aligned} \tag{7}$$

The attributes a_{q_i} belongs to $\mathcal{A}_t$ and their values v_{q_i} consists of a set of terms specified by the user.

A profile $\wp$ consists of different structures: the $Preference\ Profile\ \mathcal{PP}_u$, the $Situation\ Set\ \mathcal{SS}$ and the $Constraint\ Set\ \mathcal{CS}$.

The $Preference\ Profile$, $\mathcal{PP}_u$ has the following form:

$$\begin{aligned} \mathcal{PP} &= \mathcal{PP}_u | \mathcal{PP} \wedge \mathcal{PP} | \mathcal{PP} \vee \mathcal{PP} | \neg \mathcal{PP} \\ \mathcal{PP}_u &= (< a_{p_1}, v_{p_1} >, \ldots, < a_{p_k}, v_{p_k} >) \end{aligned} \tag{8}$$

where a_{p_j} is an attribute of $\mathcal{A}_t$ and v_{p_j} its value.

The $Situation\ Set$ is a database of situation S_i. A situation is a 4-tuple $S =< q, d_i, act_i, \mathcal{E}_i >$ in which q is a query (or the $Preference\ Profile$), d_i is a retrieved document, act_i is the action performed on d_i and $\mathcal{E}_i$ concerns data for low level functionalities. From the $Situation\ Set$ we can rebuild the 'history' of document retrieval. After a search session, the user may decide for document rejection or acceptance, and may choose an action for each retrieved item. The profile $\wp$ is updated with the last situation.

3.2 Filtering Documents with ABIS

ABIS supports two search modes, document retrieval and autonomous navigation of the network. The matching functions for retrieval and searching are similar.

3.2.1 The Query Mode

The query mode is activated by a user request. The request is a partial description of the item(s) he wants to retrieve. The window of the query mode contains slots (named from the set $\mathcal{A}_t$) for the attribute values, e.g., a list of keywords, the name of the author, the title, the type, etc.

ABIS supplies data for Harvest and processes its answers. After having translated the query into the brokers' language, the agent addresses the search towards a collection of brokers depending on the chosen modality.

The result of this process is a set of items $\mathcal{D}_{all}$ that needs to be further filtered. $\mathcal{D}_{all}$ is compared with the the query to find the most similar items. Then the $Situation\ Set$ is used to predict a user action.

The similarity between the query q and a document $d \in \mathcal{D}_{all}$ is given by the formula:

$$sim(q,d) = \sum_{a_i \in \mathcal{A}_{t_q}} sim(q.a_{q_i}, d.a_{q_i}) \tag{9}$$

where the score $sim(q.a_{q_i}, d.a_{q_i})$ between the attribute vector a_{q_i} in the query q and the same attribute vector in the document $d \in \mathcal{D}_{all}$ is given by the formula:

$$sim(q.a_{q_i}, d.a_{q_i}) = q.a_{q_i} \cdot d.a_{q_i} \tag{10}$$

If $sim(q,d)$ is less than a fixed threshold, the document d is deleted from $\mathcal{D}_{all}$. The result is a set $\mathcal{D}_q$ of items actually satisfying the query.

The agent then predicts the possible user action for each item in $\mathcal{D}_q$. The *memory-based reasoning* method introduced in [Stanfill and Waltz, 1986] is applied to the *Situation Set* $\mathcal{SS}$ with respect to the elements of $\mathcal{D}_q$ (see details in Section 3.3).

3.2.2 The Surfing Mode

The agent navigates the network in the surfing mode and selects documents for the user according to his current *Preference Profile* $\mathcal{PP}_u$. The agent periodically visits (interesting) sites, checks for new items and discovers new sites.

A way of navigating into the hyper–textual space offered by the network is to apply the so–called *robot* paradigm. A robot is a program that starts from a document and recursively descends the connected documents following the links as they come. This search is based on the assumption that the reachable documents from a set of URLs interesting to the user, have a higher probability of being relevant. In our framework, the robot just follows the links and then ABIS computes similarities between the connected documents and the *Preference Profile*.

The agent instructs the robot on how to navigate the network and to collect the items satisfied by the current *Preference Profile*.

The search method is similar to the *fish-search* algorithm [Bra and Post, 1994] which performs a depth-first navigation with the heuristic of marking the visited links. The links are periodically visited.

Similarly to the query mode described above, the selected set is matched again against the *Situation Set*: the documents together with a connected action are proposed to the user.

3.3 Predicting User Behavior

The prediction of the user's action on a document d is based on a comparison with the documents in the $\mathcal{SS}$. The chosen action is that performed in the most similar item.

The similarity between two documents is computed by applying the memory-based method. It has been used [Stanfill and Waltz, 1986] for predicting the pronunciation of a letter in a speech system. We have modified and adapted the algorithm to the filtering context.

The original idea is to represent a structured object as a record containing a fixed set of *fields*. A *target* is an object with at least one empty field whose value has to be predicted. The memory-based reasoning assumes that the empty value can be deduced from the best match between the target and the available set of records. An algorithm is applied to measure the dissimilarity between the target and each record in the database. The empty field takes the value from the corresponding field of the most similar record.

In ABIS objects are documents, fields are attributes, the empty target field is the action, the database is the *Situation Set*. Let us consider a document target $d \in \mathcal{D}_q$ for which the agent has to decide the action act in the set $\mathcal{ACT}$ of possible actions. This document is compared with the documents $d_{\mathcal{SS}_j}$ in the situation set.

The procedure calculates the dissimilarity Δ of d with respect to the documents $d_{\mathcal{SS}_j}$ and then chooses the action connected to the document with the lowest value of Δ:

$$\Delta(d, d_{\mathcal{SS}_j}) = \sum_{a_i \in \mathcal{A}} \delta_{a_i}(d.a_i, d_{\mathcal{SS}_j}.a_i) \tag{11}$$

where

$$\delta_{a_i}(d, d_{\mathcal{SS}_j}) = \begin{cases} 0 & \text{if } sim(d.a_i, d_{\mathcal{SS}_j}.a_i) \geq t_u \\ w_{a_i}^{act}(d) & \text{otherwise} \end{cases} \tag{12}$$

In the original algorithm $sim(d.a_i, d_{\mathcal{SS}_j}.a_i) = 0$ if the attribute values were equal. Since the attributes have textual values, the similarity is reckoned with the inner product as in (10). In our context it is then more appropriate to compute dissimilarity up to a threshold t_u.

The distance δ is zero when the two features are similar, otherwise it is necessary to assign a weight to the attributes with respect to the document d that measures the influence of that attribute in predicting act.

We adopt the metric proposed in [Stanfill and Waltz, 1986] based on the frequencies of having the same action in similar documents:

$$w_{a_i}^{act}(d) = \sqrt{\sum_{v \in \mathcal{ACT}} \left(\frac{|\{d_{\mathcal{SS}_j} : sim(d.a_i, d_{\mathcal{SS}_j}.a_i) \geq t_u \textit{ and } d_{\mathcal{SS}_j}.act = v\}|}{|\{d_{\mathcal{SS}_j} : sim(d.a_i, d_{\mathcal{SS}_j}.a_i) \geq t_u\}|}\right)^2} \tag{13}$$

The action related to the most similar document can be accepted or not by the user. If the user has relinquished the control to the agent, ABIS directly applies act on d.

4 Conclusion

We have presented the probabilistic and the memory–based reasoning models for filtering information on Internet. Two systems ProFile and ABIS are currently working on these models. ProFile's learning phase is fast and profiles are updated on-line. Moreover document selection can be done on-line since the selection function is based on the binary classification principle and not on the document ranking principle as used by the conventional information retrieval systems.

ABIS' main task is to refine the selection of the gathered information from the network. The memory–based reasoning model is indeed appropriate whenever the document selection comes along to an automatic action to be performed on the document itself. The action type is obtained by inductively reasoning on all past situations which are suitable represented in the user profile. Data are homogeneously represented with the SOIF format.

While ProFile selects huge amount of data for a wide number of users, ABIS performs intelligent tasks for the selected information. The two systems can be easily integrated.

References

[Aalbersberg, 1992] Aalbersberg, I. (1992). Incremental relevance feedback. In *Proceedings of ACM–SIGIR 92 Conference*, pages 11–22, Denmark.

[Amati et al., 1997a] Amati, G., Crestani, F., and Ubaldini, F. (1997a). A learning system for selective dissemination of information. In *IJCAI-97, Fifteenth International Conference on Artificial Intelligence*, Nagoya, Japan.

[Amati et al., 1997b] Amati, G., Crestani, F., Ubaldini, F., and Nardis, S. D. (1997b). Probabilistic Learning for Information Filtering. In *RIAO97,Computer–Assisted Information Searching on Internet*, Montreal, Canada.

[Amati et al., 1996a] Amati, G., D'Aloisi, D., Giannini, V., and Ubaldini, F. (1996a). An lntegrated system for filtering news and managing distributed data. In *First International Conference on Practical Aspects of Knowledge Management (PAKM)*, Basel, Switzerland.

[Amati and van Rijsbergen, 1995] Amati, G. and van Rijsbergen, C. (1995). Probability, information and information retrieval. In *Proceedings of the First International Workshop on Information Retrieval, Uncertainty and Logic*, Glasgow, Scotland.

[Amati et al., 1996b] Amati, G., van Rijsbergen, C., and Ubaldini, F. (1996b). The maximum expected utility principle and information retrieval. In *Proceedings of the Conference on Information, Statistics and Induction in Science*, pages 129–140, Melbourne, Australia. World Scientific.

[Balabanovic and Shoham, 1995] Balabanovic, M. and Shoham, Y. (1995). Learning Information Retrieval Agents: Experiment with Automated Web Browsing. In *Proceedings of the AAAI 95 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments.*

[Belkin and Croft, 1992] Belkin, N. and Croft, W. (1992). Information Filtering and Information Retrieval: two sides of the same coin? *Communication of the ACM*, 35(12):29–38.

[Borghoff and Schlichter, 1996] Borghoff, U. and Schlichter, J. (1996). On Combining the Knowledge of Heterogeneous Information Repositories. *Journal of Universal Computer Science*, 2(7):515–532.

[Bra and Post, 1994] Bra, P. D. and Post, R. (1994). Information Retrieval in the World-Wide Web: Making Client-Based Search Feasible. In *Proceedings of the First International Conference on the World-Wide Web.* `http://www.elsevier.nl/cgi-bin/ID/WWW94`.

[Brancaleoni et al., 1997] Brancaleoni, R., Cesta, A., and D'Aloisi, D. (1997). MASMA: A Multi-Agent System for Scheduling Meetings. In *Proceeding of the 2nd International Conferenxe on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 31–50, London.

[CACM, 1994] CACM (1994). Special Issue on Intelligent Agents. *Communication of the ACM*, 37(7).

[Callan, 1996] Callan, J. (1996). Document filtering with inference networks. In *Proceedings of ACM SIGIR*, pages 262–269, Zurich, Switzerland.

[Callan et al., 1995] Callan, J., Lu, Z., and Croft, B. (1995). Searching distributed collections with inference networks. In *Proceedings of the 18th ACM SIGIR Conference*, pages 21–28, Seattle, Washington.

[Cesta and D'Aloisi, 1996] Cesta, A. and D'Aloisi, D. (1996). Active Interfaces as Personal Assistants: a Case Study. *SIGCHI Bulletin*, 28(3):108—113.

[D'Aloisi and Giannini, 1995] D'Aloisi, D. and Giannini, V. (1995). The Info Agent: An Interface for Supporting Users in Intelligent Retrieval. In *Proceedings of the ERCIM Workshop "Towards User Interfaces for All: Current Trend and Future Efforts"*, pages 143–155.

[Dent et al., 1992] Dent, L., Boticario, J., Dermott, J. M., Mitchell, T., and Zabowski, D. (1992). A Personal Learning Apprentice. In *Proceedings of the National Conference on Artificial Intelligence*, pages 96–103. MIT Press.

[Dunlop, 1997] Dunlop, M. (1997). The effect of accessing non–matching documents on relevance feedback. *ACM Transaction on Information Systems*, (Forthcoming).

[Etzioni, 1995] Etzioni, O., editor (1995). *Working Notes of the AAAI Spring Symposium on Software Agent*. AAAI Press.

[Etzioni and Weld, 1994] Etzioni, O. and Weld, D. (1994). A Softbot-Based Interface to Internet. *Communication of the ACM*, 37(7):72–76.

[Hardy et al., 1995] Hardy, D., Schwartz, M., and Wessels, D. (1995). Harvest: Effective Use of Internet Information (Harvest User's Manual, Version 1.2). Technical Report CU-CS-743-94, University of Colorado at Boulder.

[Harman, 1992] Harman, D. (1992). Relevance feedback revisited. In *Proceedings of ACM–SIGIR 92 Conference*, pages 1–10, Denmark.

[Harman, 1996] Harman, D. (1996). Overview of the fifth text retrieval conference (TREC–5). In *Proceeding of the TREC Conference*, Gaithersburg, MD, USA.

[Harper and van Rijsbergen, 1978] Harper, D. and van Rijsbergen, C. (1978). An evolution of feedback in document retrieval using co–occurrence data. *Journal of Documentation*, 34(3):189–216.

[Hintikka and Suppes, 1970] Hintikka, J. and Suppes, P., editors (1970). *Information and Inference*, chapter J. Hintikka: 'On semantic information', pages 3–27. Synthese Library. D. Reidel publishing company, Dordrecht-Holland.

[Kilander, 1995] Kilander, F. (1995). A brief comparison of news filtering software. *Unpublished paper*.

[Lang, 1995] Lang, K. (1995). NewsWeeder: learning to filter netnews. In *Proceedings of ML 95*, pages 331–339.

[Maes and Kozierok, 1993] Maes, P. and Kozierok, R. (1993). Learning Interface Agents. In *Proceeding of the National 11th National Conference on Artificial Intelligence (AAAI 93)*, pages 459–465. AAAI Press/The MIT Press.

[Robertson and Sparck–Jones, 1976] Robertson, S. and Sparck–Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146.

[Salton, 1971] Salton, G. (1971). *The SMART Retrieval System*. Prentice Hall, New Jersey.

[Salton and McGill, 1983] Salton, G. and McGill, M. (1983). *Introduction to modern Information Retrieval*. McGraw–Hill, New York.

[Stanfill and Waltz, 1986] Stanfill, C. and Waltz, D. (1986). Toward Memory-Based Reasoning. *Communication of the ACM*, 29(12):1213–1228.

[van Rijsbergen, 1979] van Rijsbergen, C. (1979). *Information Retrieval, second edition*. Butterworths, London.

[Yan and Garcia-Molina, 1995] Yan, T. and Garcia-Molina, H. (1995). SIFT - a tool for wide-area information dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, pages 77–186.