An Architecture and Category Knowledge for Intelligent Information Retrieval Agents*

Hsieh-Chang Tu and Jieh Hsiang
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
email: {tu,hsiang}@csie.ntu.edu.tw

Abstract

Information overload has become a serious problem for users of the World Wide Web. In this paper we propose to use intelligent information retrieval (IIR) agents as a solution to this problem. We identify the desirable features of an IIR agent, including intelligent search, navigation guide, auto-notification, personal information management, personal preferred interface, and tools for easy page-reading. A modularized agent architecture is then proposed. We describe the responsibility of each component and how they are combined to performed to the various tasks of the IIR agent. We point out that group knowledge, acquired from preferences of other users in the same group, may be useful. Knowledge of our agents is primarily represented by categories. After defining and clarifying the difference between clusters, directories and categories, we present category representations as an abstraction of certain desired information. Possible applications of category knowledge are also examined.

1 Introduction

In a short few years, the World Wide Web has become one of the most important media with which people share information resource. Web information is primarily exhibited through Web pages, designed and written by content providers. The enormous amount of available information induces the problem of *information overload*, that there is too much information for people to digest. To alleviate this problem, one needs better information retrieval (IR) software to serve as a filter between the user and the information retrieved over the Web. Such a software should provide *intelligent search*, which provides the user with more interesting Web pages and fewer uninteresting ones. When the user is surfing the net, the software should also be able to suggest interesting URL's to visit. Finally, there may be "hot pages" whose contents may change frequently. The IR system should be able to notify the user of such changes automatically.

Attempts have been made to reduce the problem of information overload. There are browsers that are not only equipped with user-friendly interface but also support Java and other powerful languages. Web directories, such as YAHOO![1], organize "important" Web pages in a way similar to yellow pages. Search engines such as ALTAVISTA[2], EXCITE[3], INFOSEEK[4], and LYCOS[5] use indexing techniques for users to retrieve potentially relevant pages through queries. Intelligent agents, programs which are supposed to exhibit human behavior, are also proposed to help users retrieve, locate, and manage Wed information [Lee97]. Examples include POINTCAST NETWORK[6] and PATHFINDER[7], which offer personalized news and information, FIREFLY[8], which makes movie and music recommendations, and WebWatcher [Arm95] which interactively helps users locate desired information. The design of softbots [Etz94], on the other hand, is aiming at providing integrated solutions to utilize Web resources.

In this paper we propose an architecture of intelligent information retrieval (IIR) agents. We first discuss what we think are desirable features of a good IIR agent. We describe the notion of an *agent community* in section 3, and identify an IIR agent as one agent in the community. We propose an integrated architecture to carry out features of an IIR agent. We further decompose an IIR

*Partly supported by Grant NSC 87-2213-E-002-012 of the National Science Council of the Republic of China.

[1] http://www.yahoo.com/
[2] http://altavista.digital.com/
[3] http://www.excite.com/
[4] http://www.infoseek.com/
[5] http://www.lycos.com/
[6] http://www.pointcast.com/
[7] http://pathfinder.com/
[8] http://www.firefly.com/

agent into components called *subagents*, each of which can be regarded as an independent module to perform a specific function. We also discuss cooperations among subagents. Section 4 discusses the issue of *category* information. Category information plays a crucial role in an agent's knowledge. We distinguish the meaning of a category from the well-known definition of a cluster. Possible applications of category knowledge are addressed next. A concluding remark about IIR agents is given at the end.

2 Essential Features of Web IIR Agents

Before designing an appropriate agent architecture, it is essential to first decide what an IIR agent should do. To answer this question, one should examine what kind of difficulty people encounter when they try to get information over the Web. Some of these obstacles come from the difficulty of using a software, but most problems are caused by information overload – there is too much information for the user to retrieve. Since the purpose of an IIR agent is to assist people retrieve and manage information on the Web. it should have following features:

1. **Intelligent search**: An effective and efficient search of information from a database is a major issue on the research of information retrieval. When people start to search for information from the Web, they often become frustrated when the search result contains too little useful information (or too much garbage). An intelligent agent should give the user an interactive environment so that the user's information need can be pinpointed exactly.

2. **Navigational guide**: When surfing through the Web it is easy to "go astray" in cyberspace. A good IIR agent should provide guides or roadmaps so that users can get assistance when stuck in their navigation on the Web. For instance, the agent may analyze pages recently read by the user in order to suggest related subject areas and pages. Another kind of navigational guide is to highlight potentially interesting hyperlinks.

3. **Information auto-notification**: It is tedious for people to check whether a page has been updated. After the user specifies the kind of information he needs, an IIR agent should be able to detect updated information or even download them automatically. Messages may be sent to user to notify that new data have become available. Furthermore, it is worthwhile for an agent to analyze the user's reading preference so that it may prompt interesting pages to the user automatically.

4. **Personal information management**: Categories, directories, or folders are familiar ways for people to manage tree-structured hierarchical data. It is useful for an IIR agent to manage personal categories in an intelligent way. For instance, the agent may provide suggestions to build a personal category tree for each user. This category information can later be used to help user search or navigate on the Web. Another example of information management is to automatically organizing bookmarks [Maa96] so that the user may handle bookmarks more easily.

5. **Personal preferred interface**: Each user may want to have his preferred interface. For instance, a user may want to set his default background colors for pages that do not indicate background colors. Another simple example is that, since pages may be written by languages with different language codes (e.g., the BIG-5 code for Chinese), a good agent should try to display appropriate characters after detecting what language (code) the page uses.

6. **Tools as reading-aide**: A good IIR agent may also provide tools to help the user with reading retrieved papers. Such tools may include on-line dictionaries and translation programs. These programs are usually stand-along agents themselves. The IIR agent should allow easy incorporation of such tools.

3 Proposed Agent Architecture

3.1 Agent Community

We consider an *agent* as a goal-oriented program with some learning ability. An agent can dynamically adapt to individual users and can perform certain tasks autonomously. An *agent community* is a group of agents working together to serve a group of users. In an agent community, agents interact with each other and can cooperate to solve problems if necessary. It is also convenient to further divide agents into *task agents* and *interface agents*. Each task agent offers a specific service and they communicate with each other to execute more advanced functions. For instance, a typical IIR agent may not support functions such as looking up dictionaries or translating page contents into another language. These functions are done by other task agents (say $\mathcal{T}$). An IIR agent should be able to communicate with $\mathcal{T}$ to offer such services to the user. Interface agents are responsible for keeping and updating the user profiles and communicating a user's need to the task agents.

Among the task agents in the agent community, there is a resource management agent, called *managent*, which plays a special role. The managent keeps the list of the services provided by agents in the community. If a new user joins the group, the managent announces it to all task agents. Functions offered by agents can then be automatically prompted to the new user. This allows the user easy access to services available in the agent community. An interface agent acts like the DeskTop Manager in most systems with a graphics user interface. A browser which allows all possible Web information to be properly displayed can also be regarded as an interface agent.

An example of an architecture of agent community is illustrated in Fig 1.

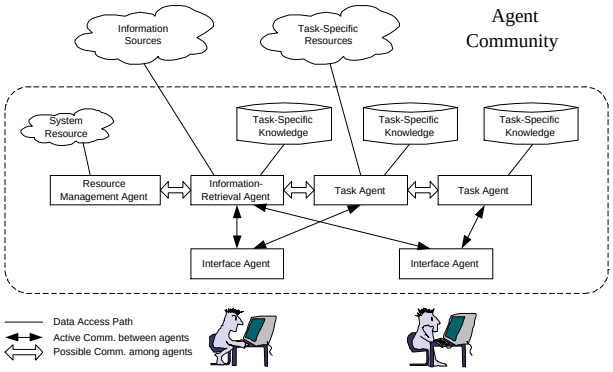

Figure 1: The architecture of agent community

3.1.1 Group and Personal Agents

An IIR agent needs to keep two types of preferences; each user's own preference about search and navigation, and the preference of each group of users. The latter is needed because web pages that are interesting to most users in a group are likely to be interesting to others in the group. Thus some kind of *group preference*, computed from profiles of users in the group, is required. Furthermore, in order to reduce network traffic load, web pages requests by users in the same group should be handled by the same program. These considerations motivate the design of an IIR agent into two layers. Each user has his own *personal agent (PA)* which keeps a profile of user preference, and there is a *group agent (GA)* that handles group knowledge and preference. Intuitively, a PA offers all anticipated features and learns personal preference from the user it serves. A GA accumulates the knowledge about personal preferences it obtains from the PA's, transforms it into the collective group preference, finds interesting pages that reflect the group preference, and monitors web pages that the users wish to be watched. The various functions of the GA and PA are captured in modules which we call *subagents*. We shall describe these subagents and their collaboration in detail in the next section. Meanwhile, the proposed IIR agent architecture is presented in Fig 2.

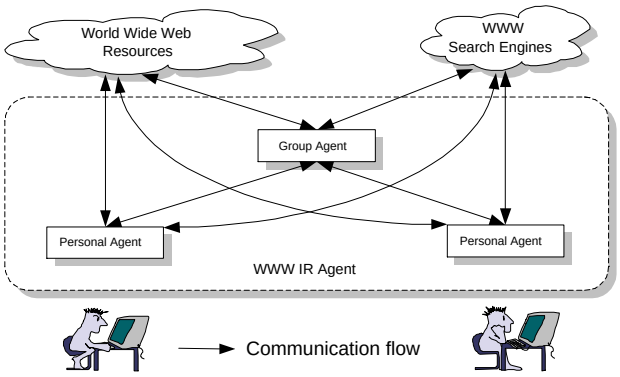

Figure 2: A coarse view to the architecture of information retrieval agents

3.2 Agents and Subagents

Similar to decomposing a program into modules, an agent can also be organized as subagents. Each subagent, working independently, performs some preascribed feature of the agent. Subagents have their own local databases, and share the same knowledge base with other subagents in the same agent. A finer architecture of the IIR agent is described in Fig 3, in which a group agent and a personal agent are enclosed in dashed, rounded rectangles. Boxes within agents represent *subagents*, which are separate, independent program modules. Subagents work together to form a group or personal agent. We brief the functions of each subagent as follow:

- **Communication subagent**: Each (group or personal) agent has a communication subagent which takes the responsibility of sending, receiving, and possibly interpreting messages from the external world. A communication subagent may be regarded as a program listening to certain communication ports in conventional network programming, except that the former can actively watch request queues. It is also desirable to equip the communication subagents with some learning capability or with a uniform and flexible protocol such as KQML, so that

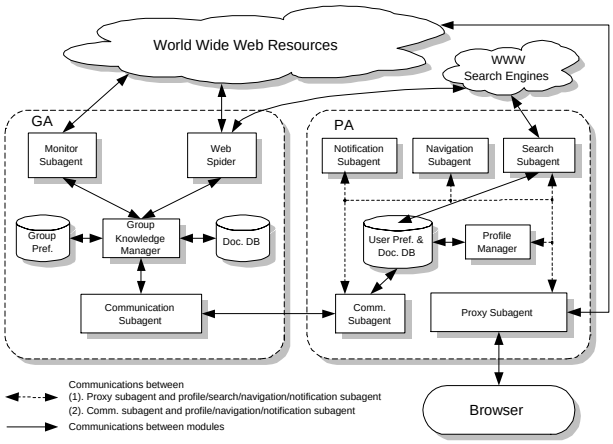

Figure 3: A finer view to the architecture of information retrieval agents

collaborations between agents can become more effective.

- **Proxy subagent**: According to RFC1945[9], a proxy is an intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy subagent is a special program that intercepts messages between the user and the Web. It also serves as a communication subagent between the user and the personal agent. If the user wishes to set his own personal preference, he must interact with the profile manager via the proxy subagent. Since the proxy subagent knows which pages have been accessed by the user, it provides necessary information for the IIR agent to learn about user preference. A proxy subagent caches frequently accessed pages so that unnecessary network traffic can be reduced. It may also be desirable for the IIR agent to pre-fetch pages which may be interesting to the user. Although prefetching pages may increase network traffic, good interaction between the proxy subagent and the man-agent may allow pre-fetching be done when the network traffic is light.

- **Search subagent**: There are already many search engines on the Web. Instead of designing its own, an IIR agent may act as a "meta search engine", which collects results obtained from sending user queries to existing search engines. Since search engines may require different query formats, the search subagent is responsible for interacting with the user so that the user can format his queries properly. The subagent translates user queries to the formats acceptable by (a pre-defined set of) search engines, issues formatted queries to these engines, and collects returned results to the user. The subagent may ask the user for some category information (discussed in detail in Section 4) so that better searching results can be presented to him.

- **Navigation subagent**: Given a set of pages recently read by the user, the navigation subagent attempts to classify these pages into pre-defined categories. If the user gets lost in cyberspace, he may ask the navigation subagent to suggest interesting hyperlinks. The subagent will prompt categories, which are related to the current page being browsed, to the user. Each category contains hyperlinks as well as titles or descriptions about the corresponding Web pages. Some hyperlinks may be manually coded (such as important Web sites), and some are obtained from recently browsed pages. Categorized hyperlinks thus provide the user ways to jump to other pages which are related to pages currently being browsed.

- **Notification subagent**: The user may ask the notification subagent to monitor frequently changed pages. If these pages are changed, the subagent will notify the user automatically. On the other hand, the user may ask the IIR agent to search through the Web to find pages fulfilling requirements specified by the user. The notification subagent should provide the user with a comprehensive way to specify his information needs. The notification subagent does not monitor or search Web pages itself. Instead it sends monitor or search requests to the group agent. The monitor subagent and Web spider (described later) in the group agent are responsible for handling these requests.

- **Profile manager**: The profile manager modifies the user preference, either by interacting with the user directly, or by communicating with the group agent to obtain group preference. It contains a knowledge explainer so that the user can read knowledge stored in the profile. Some knowledge, such as the pages to be monitored or to be searched by the Web spider, can be specified simply by a form or a table. Statistical knowledge, such as a user's category preference, is more difficult to interpret. Since we will represent such knowledge by a set of keywords, the agent needs to let the user know the role of such keywords in the representation. It is also important to allow an experienced user to edit category preference manually. Details about categories will be addressed in section 4.

[9] Request for Comments, No. 1945, a protocol standard for HTTP/1.1

- **Web spider**: The Web spider is a subagent of the group agent, and searches through the Web using a *fish-search algorithm*[10]. To bootstrap the search algorithm, we provide the spider with a list of *index pages*, which contains hyperlinks to related pages. The spider uses these hyperlinks to reach other pages, and in turn uses hyperlinks in the resulting pages to attain more pages. A pre-defined search width and search depth are required so that the spider will not lose its original searching goal by following too deep a chain of pages. The spider may also interact with the user so that search goals can be modified dynamically [Che97]. If the spider finds pages that may be interesting to all users in the group, these pages will be sent to corresponding personal agents so that the users can be notified.

- **Monitor subagent**: This subagent takes requests from personal agents and monitor specified pages to see if their contents have been modified. Monitoring can be done by downloading the page and checking it with an older version of the same page. The main reason for putting the monitor subagent as part of the group agent instead of the personal agent is to reduce network traffic (since many users may want to monitor the same pages) and to reduce the complexity of personal agents.

- **Group-knowledge manager**: Group knowledge is information pertinent to the interests of a group of users. There are basically two kinds of knowledge known to the group agent. The first kind is more of a record keeping nature. It includes pages specified by the users to monitor, the number of pictures or voice files in a page, and pages satisfying certain condition (such as containing at least two image files). The second kind of knowledge is obtained from statistics of pages accessed or read by the users. This knowledge is represented by a set of attributes and weights, which can be interpreted using notions from fuzzy sets of probability. A simple statistical knowledge is the histogram which counts the times of a page accessed by the users. Pages frequently requested may be regarded as "hot pages". Another statistical knowledge, namely the category information, plays a central role of knowledge to our IIR agents.

3.3 Collaboration among Subagents

Subagents collaborate to perform functions of an IIR agent. In the rest of the section we describe how the collaboration is done for the processes of intelligent search, auto-notification, navigation guide, and personal information management.

3.3.1 Process of Intelligent Search

A typical working scenario of intelligent search is described in Fig 4. In order to make the search "intelligent", the group agent needs some initial knowledge about categories[11] (which amounts to the initial knowledge about group preference). It announces the category knowledge to personal agents so that each personal agent will have the same initial category knowledge. The left part of the off-line preference processing shown in Fig 4 illustrates this idea. After a user sends a query, the search subagent receives this message from the proxy subagent. It then translates the user query to queries that are acceptable to existing search engines on the Web, and gathers results returned from sending translated queries to the search engines. The search subagent uses category knowledge to filter out pages it deems uninteresting, and presents the final result (via the proxy subagent) to the user. The user marks pages as interesting, uninteresting, or no comment. These labelled pages, which indicate the types of pages interesting to the user within certain categories, will be used later in the learning process. After a pre-set period of time, the personal agent communicates with the group agent about what it has learned from the user. This makes it possible for the group agent to modify its category knowledge from the user preference. The group agent then communicates back its newly gained knowledge to the personal agents.

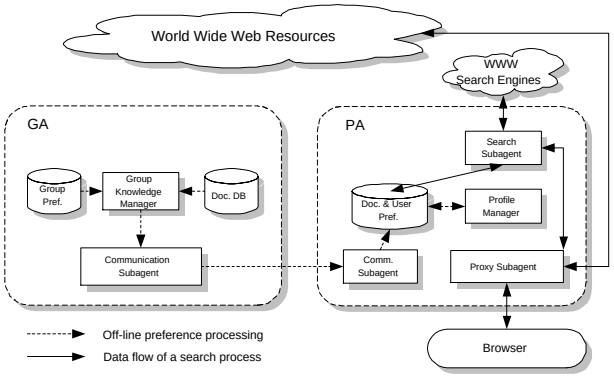

Figure 4: The process of intelligent search, including the propagation of knowledge from the group agent to the personal agent

[10] A description on the algorithm can be found in `http://www.eecs.wsu.edu/~bamberg/hypercourse/fishsearch.html`.

[11] Intuitively, categories are classes in a classification. We shall talk about categories in detail in Section 4.

We remark that the user has control over personal preference learned. This is illustrated in Fig 4 as the off-line preference processing between profile manager and the user profile. The profile manager should explain, as most as it can, what preference it has learned to the user. If the user is experienced enough, he can modify the preference to match his information need.

3.3.2 Processes of Auto-Notification, Navigation Guide, and Personal Information Management

We illustrate the process of auto-notification in Fig 5. At the beginning, the user specifies what he needs to the notification subagent. One possible specification simply indicates *page patterns*, and pages satisfying these patterns can be regarded as interesting. The notification subagent sends pattern messages to the group knowledge manager so that personal requests can be stored in the group preference database. The Web spider analyzes group requests (from the group preference database) so that Web pages interesting to group users acquire more attention. As soon as interesting pages are found, they will be put into the group document database. Similarly, the monitor subagent watches specified Web pages to see if they have been changed. It writes messages to the group document database if it wants to inform group users that some pages are changed. The group knowledge manager checks personal requests with notifiable documents, and transmit necessary information to the corresponding notification subagent. The user receives prompting message once the notification subagent decides to notify him.

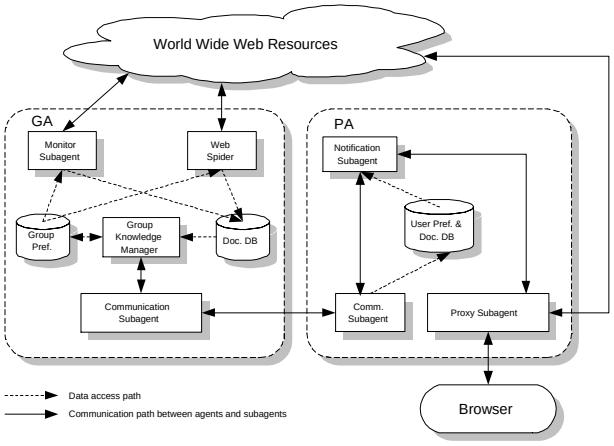

Figure 5: The process of auto-notification

To perform navigation guide, the navigation subagent consults user preference and the documents stored in the personal agent. It analyzes pages recently read by the user, and prompts related categories to the user. Hyperlinks stored in categories allow the user to visit related pages. The process of navigation guide, not including propagation of knowledge (e.g., category knowledge) from the group agent to personal agents, is shown in Fig 6 (a).

Fig 6 (b) pictures the process of personal information management. The profile manager consults the user profile and displays stored knowledge to the user. User preference includes personalized interface, interesting categories and their representations, interesting-page patterns, and other system settings (e.g., cache size used by the proxy subagent). Personal information management allows an experienced user to control his own preference profile.

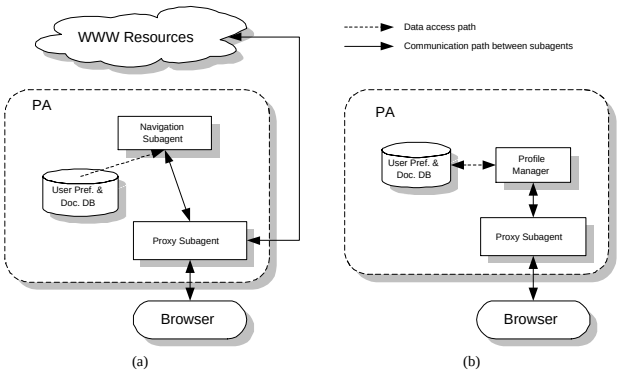

Figure 6: (a) The process of navigation-gude, not including the propagation of knowledge from the group agent. (b) Subagents involved in personal information management

4 The Formulation of Category Knowledge

Web pages normally contain multimedia information. However, it is difficult to handle information stored in non-text form. One solution is to describe multimedia information by a sequence of words [Gug96] so that it can be treated as normal texts, and use conventional IR techniques to handle the text information. In this section, we assume that information stored in Web pages can be processed via text processing. We call a page a *document* to emphasize that the retrieval is done by text processing.

Human beings are familiar with using *classification* techniques to manage a large amount of objects. Similar objects are collected into the same group so that they can be retrieved conveniently. One of the most popular methods is *clustering*, which puts documents with similar features or keywords into the same cluster. However, since features or keywords may not reveal semantic information of a document, a cluster may contain too much noise to be truly useful. A more effective way is to group documents according to semantic concept. We call a group of documents that captures certain semantic concept a *directory*. The use of directories, although intuitively reasonable, is not feasible computationally since it is extremely difficult to compute the semantic information of a document precisely. We therefore introduce a notion of *category*, which computes an *approximation* of a directory of documents. In the following subsections, we first review the definition of *document vector model*, which is a popular method in document processing and will also be used in our method. We then discuss the relationship among a cluster, a directory, and a category in more detail in subsection 4.2. A special category called `User_Category` is then introduced to capture a user's recent browsing preference. Finally, we examine possible applications of category knowledge in an IIR agent.

4.1 The Document Vector Model

Allowing inputs in natural language is one way to make an IR system friendly. However, natural language understanding is a notoriously difficult task. Therefore it is common to employ some "approximation" method to analyze the queries and documents. One popular method for processing documents is the *vector model* [Sal89], which regards each document as a vector. Let $\mathcal{V}$ be a *finite* vocabulary of words and let $v = |\mathcal{V}|$. The *word space* $\mathcal{W}$ is the v dimensional vector space over real numbers. Each document in a given database is represented by a vector $\mathbf{d}$ (called a *document vector*) in $\mathcal{W}$. For convenience, from now on we use $\mathbf{d}$ to denote both a document and the associated document vector. We regard the set of all "valid" Web pages (i.e., pages known or accessible by the IIR agent) as a database $\mathcal{D}$ (which is also regarded as a subset of $\mathcal{W}$). Informally, the value of the i^{th} component of a vector $\mathbf{d}$ is computed from some statistical property which is a function of the i^{th} word in $\mathcal{V}$, the document $\mathbf{d}$, and the database $\mathcal{D}$. Given a vector $\mathbf{p} = (p_1, \cdots, p_v) \in \mathcal{W}$, we use $|\mathbf{p}| = \sqrt{\sum_{i=1}^{v} p_i^2}$ to denote the *length* of $\mathbf{p}$. For simplicity, we normalize each document vector $\mathbf{d} \in \mathcal{D}$ so that $|\mathbf{d}| = 1$. We remark that, in the vector model, it is possible to have the same document vector representing different documents.

An intuitive way to represent the similarity between two document vectors is by the *distance* between them. A smaller distance means that the two vectors are closer and therefore the associated documents are more similar. Let $\mathbf{p}$, $\mathbf{q}$ be two points (i.e., document vectors) in $\mathcal{D}$, we denote the distance between $\mathbf{p}$ and $\mathbf{q}$ by $d(\mathbf{p}, \mathbf{q})$, which is a non-negative real number. Let $\mathbf{p} = (p_1, \cdots, p_v)$ and $\mathbf{q} = (q_1, \cdots, q_v)$, one commonly used definition of $d(\mathbf{p}, \mathbf{q})$ is

$$d(\mathbf{p}, \mathbf{q}) = 1 - \sum_{i=1}^{v} p_i q_i,$$

which measures the similarity by the inner product of $\mathbf{p}$ and $\mathbf{q}$.

In order to automatically classify similar document vectors into the same group, one needs a definition to measure the distance between two sets of document vectors. Let P, Q be two sets of points in the word space, there are various definitions of $d(P, Q)$, the distance between P and Q. Two popular definitions are the *single-link distance* and the *complete-link distance* [Jai88, Sal89]. The former defines $d(P, Q)$ by $\min_{\mathbf{p} \in P, \mathbf{q} \in Q} d(\mathbf{p}, \mathbf{q})$, while the latter defines $d(P, Q) = \max_{\mathbf{p} \in P, \mathbf{q} \in Q} d(\mathbf{p}, \mathbf{q})$. Intuitively, the single-link distance is the distance between the most similar pair of points from the two sets (one from each set), while the complete-link distance refers to the distance of the least similar pair of points in P and Q.

4.2 Clusters, Directories, and Categories

Recall that $\mathcal{D}$ denotes a set of document vectors representing pages on the Web. A *cluster* X (of $\mathcal{D}$) is a subset of $\mathcal{D}$ such that there is a high degree of association (measured by a chosen distance function) between members in X. In practice we also require that members from different clusters have low degrees of association. Clusters are generated by *unsupervised learning* techniques, which means that the learning is performed without *labelled* training examples. By a labelled training example we mean that the classification result (i.e., whether a document belongs to a cluster) is explicitly specified in advance. Some commonly used clustering methods include the c-means algorithm [Sch92], the Learning Vector Quantization (LVQ) [Mak85], and the fuzzy clustering techniques [Zim91]. It has been shown that the cluster-based approaches can be helpful for the user to browse large document collections [Cut92]. However, clusters generated automatically are difficult to interpret, since "similar documents" defined from a distance measure may not be meaningful to most people.

Like a cluster, a *directory* is a subset of points in $\mathcal{D}$. The main difference between a cluster and a directory is

that the former is a "syntactical" group of documents, while the latter is a "semantical" group of documents. A directory is pre-defined by the retrieval system (often manually by the designer), and its semantical content means that we can *name* it in a way familiar to most people. For instance, we may group documents (semantically) related to computers by a directory call `/computer`, and organize documents (semantically) related to computer architecture by a sub-directory call `/computer/architecture`. In the real world, people have a lot of experience in handling information with this kind of naming structure. For instance, in a computer system, users are familiar with attaching a mnemonic path name to each file directory.

The main problem with using directories in a Web database is that its construction is almost impossible to automate[12]. Thus in this paper we propose a notion of a *category*, which is an approximation of a directory. Given a directory Δ, we denote an associated *category* by C_Δ. A category is also a subset of points in $\mathcal{D}$[13].

Assume that we are given a set of (possibly hierarchical) directory names. Since in practice directories exist only in an abstract sense, we approximate them using a *training set* $T \subseteq \mathcal{D}$. Each sample in the training set T is explicitly labelled as belonging to one or more directories. Our task, then, is to create, for each directory Δ, a category C_Δ from the information provided by T. Since categories are generated by a computer, we shall introduce a vector representation of categories in the following section.

4.3 Category Representation

Hierarchical categories can be represented by a tree structure. We call each node in such a tree a *category node*. A category node is labelled by a *category name*, which is the path starting from the root of the tree to that tree node. We use $\nu(C)$ to denote the category node of the category C. A category C' is said to be a *subcategory* of C if $\nu(C)$ is an *ancestor* of the $\nu(C')$. It is called a *proper subcategory* if $\nu(C)$ is the *parent* of $\nu(C')$. A category without subcategories is a *leaf category*. A category which is neither root nor leaf is an *intermediate category*.

A category C is represented by a *prototype vector* $\mathbf{c} \in \mathcal{W}$ and a *radius* $\epsilon(C)$, which is a positive number. The interpretation of category representation is given as follows. If C is a leaf category, C is defined as the set of document vectors $\mathbf{d}$'s such that $d(\mathbf{c}, \mathbf{d}) \leq \epsilon(C)$. If C is an intermediate category, C is defined as the union of its subcategories and the document vectors $\mathbf{d}$'s such that $d(\mathbf{c}, \mathbf{d}) \leq \epsilon(C)$. If C is the root category, its radius is set to 1 so that it consists of all document vectors in the database. Intuitively, a category should have a radius bigger than those of its subcategories.

Notice that a parent category is defined from its subcategories. This is different from conventional hierarchical categorization techniques where a subcategory is defined only when its parent category is defined. Our "bottom-up" definition is inspired from the observation that the vector representation of a subcategory is normally more precise than that of a parent category.

The generation of the categories (which approximate the intended directories) is done using *supervised learning* techniques from the labelled training set T. Several methods are known, such as the least square functional approximations [Sch96] or the training algorithms for linear text classifiers [Lew96]. Some earlier experiments show that a prototype vector learned from specific user interests achieves encouraging results in selecting interesting Web pages [Sho95]. In practice documents satisfying a user interest may be defined as a personal directory. We remark that the allowance of one document vector to be classified into several directories may complicate the learning process.

Once appropriate category representations are computed from the training set, they can be used to perform document classification. That is, we may classify a document vector $\mathbf{d}$ (which may not belong to the training set) into an existing category C, if one of the following holds:

1. C is a leaf category and $d(\mathbf{d}, \mathbf{c}) \leq \epsilon(C)$.
2. C is an intermediate category, and either $\mathbf{d}$ is classified to some subcategory of C, or $d(\mathbf{d}, \mathbf{c}) \leq \epsilon(C)$.
3. C is the root category.

4.4 A Special Category: `User_Category`

Interesting pages browsed by the user usually reveals information about the user preferences. Such information can be used to form a special category called `User_Category`[14] (which is represented by a vector $\mathbf{u}$ and radius $\epsilon(\mathbf{u})$). We assume that the radius (or threshold) $\epsilon(\mathbf{u})$ is set by the user. The agent adopts the following rules to modify the prototype vector $\mathbf{u}$ automatically:

1. Let $\mathbf{d}$ be the vector representing the browsing page. We adjust $\mathbf{u}$ by

$$\mathbf{u} \leftarrow \frac{\mathbf{u} + \eta\mathbf{d}}{|\mathbf{u} + \eta\mathbf{d}|},$$

[12] A case in point is YAHOO!, which has one of the best directory structure on the Web today.

[13] As indicated before, a category may also contain Web hyperlinks.

[14] It is also possible to form several user categories for the same user. For simplicity we consider here only the case where one such category is formed.

where $\eta < 1$ is a positive number called the *learning rate*.

2. After some period of time, the agent may decide to "forget" information collected from pages browsed a long time ago. Let J be the set of pages recently browsed by the user. For any document vectors $\mathbf{p}$ and $\mathbf{q}$, we use $\mathbf{p} \ominus \mathbf{q}$ to denote a vector whose i^{th} element is $p_i - q_i$ if $p_i - q_i > 0$, and is 0 otherwise. We construct a vector $\mathbf{w} = (w_1, ..., w_v) \in \mathcal{W}$ by setting $w_i = 1$ if the i^{th} word appears in J, and $w_i = 0$ otherwise. The vector $\mathbf{u}$ is adjusted by

$$\mathbf{u} \leftarrow \frac{\mathbf{u} \ominus \tau\mathbf{w}}{|\mathbf{u} \ominus \tau\mathbf{w}|},$$

where $\tau \leq 1$ is a positive number called the *discarding rate*.

Knowledge of `User_Category` can be used in the auto-notification process to filter out uninteresting pages. On the other hand, the *group recent preference* may be built from each user's `User_Category` knowledge so that the Web spider can tune its search direction to find more interesting pages.

4.5 Utilizing Category Knowledge in an IIR Agent

Since categories are obtained via supervised learning with a training set, it contains more semantic information and closer resembles the intended directories than clusters. The knowledge contained in the categories is beneficial to intelligent information retrieval. It enables us to do document classification, which implies that we may use categories as filters to exclude documents that do not belong to categories that interest a specific user. In the following, we describe how category knowledge can be used in different aspects of an IIR agent.

- Intelligent search: Category knowledge can be used to filter out uninteresting documents. In addition to the normal query, the agent may ask the user to provide category information (e.g., specify interesting categories) so that documents not belonging to the specified categories can be filtered out. To be more specific, let P be the set of documents returned by search engines. If C_1, ..., C_m are interesting categories, a document $d \in P$ will be presented to the user if d is classified as belonging to C_i, where $1 \leq i \leq m$.

- Navigation guide: Let us assume that each category contains sample documents (stored in the group agent) which come from either training samples or pages identified by users to belong in the category. When a user needs navigation guide, the agent first analyzes recently browsed pages to determine which categories (say $\mathcal{C}$) are related to the user's recent interest. A category tree, with categories in $\mathcal{C}$ highlighted or ranked high, is prompted to the user. After the user clicks an interesting category, pages stored in the selected category can be prompted to the user as suggested pages.

- Auto-notification: The use of category knowledge in auto-notification is similar to that in intelligent search, namely to use categories to filter out uninteresting documents. To be more specific, the user sets document criteria so that documents (found by the Web spider) matching the constraints can be suggested to the user. The constraints specified typically consist of (natural language) queries, shallow multimedia information (i.e., number of pictures, images, or voice files in a page), plain document information (such as document location, document size, the number of hyperlinks, or possibly document author), and interesting categories.

- Personal information management: A user may preserve interesting pages as bookmarks. Storing bookmarks hierarchically is valuable to manage pages that are identified as interesting to the user. People may simply store bookmarks under hierarchical categories offered by the personal agent. The user can rename, add, delete, or modify the category names stored in the personal agent. If the personal agent find that there are too many documents or bookmarks stored, it may group them into several clusters and then ask the user to give a name to each cluster. In this way categories may grow semi-automatically (since the user has to give names to them), and cluster analysis will be helpful to construct personal categories.

A profile manager is responsible for explaining the meaning of category knowledge to the users. Explanation of category knowledge is helpful to a naïve user to understand what has been stored in the personal agent. Recall that a category C is represented by a vector $\mathbf{c}$ and a radius $\epsilon(C)$. Adjusting the i^{th} component of $\mathbf{c}$ corresponds to tuning the "importance" of the i^{th} word (in $\mathcal{V}$) to C, while modifying $\epsilon(C)$ amounts to changing the size or range of C. An experienced user can probe whether the adjustment of weights satisfies his demand, by classifying sample documents into adjusted categories.

5 Concluding Remarks

The growing popularity of the World Wide Web worsens the problem of information overload. IIR agents are proposed as one possible solution to assist people manage Web information. We point out the desirable features of an IIR agent, and propose an agent architecture which supports the implementation of these features. Subagents of an IIR agent are also identified so that they can be designed and implemented separately. Collaboration among subagents are illustrated and discussed.

We then turn to the question of classification of Web documents. We then introduce a notion of categories which capture better the informal but "conceptually ideal" notion of directories. We describe how the categories can be represented by vector models and can be obtained through supervised learning with a training set of documents. Our notion of categories can be automated and seems better than the more common approach of clusters, which are built via unsupervised learning harder to understand by human. How categories can be utilized in an IIR agent is also described.

Clustering analysis has attracted much attention in traditional IR research. There is, however, little study about hierarchical categories and their representations. It is imperative to encourage more theoretical, as well as experimental, studies about categories.

References

[And73] M. R. Anderberg, *Cluster Analysis for Applications*, New York: Academic, 1973.

[Arm95] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, *WebWatcher: A Learning Apprentice for the World Wide Web*, AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, 1995.

[Sho95] M. Balabanović and Y. Shoham, *Learning Information Retrieval Agents: Experiments with Automated Web Browsing*, AAAI-95 Spring Symposium on Information Gathering from Heterogenous, Distributed Environments, 1995.

[Che97] H. Chen, Y. M. Chung, M. Ramsey, C. C. Yang, P. C. Ma, and J. Yen, *Intelligent Spider for Internet Searching*, Proceedings of the 31^{st} Hawaii International Conference on System Sciences, HICSS-30, Vol. 4, pp. 242-252, 1997.

[Cut92] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey, *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*, SIGIR'92, 1992.

[Etz94] Oren Etzioni and Daniel Weld, *A Softbot-Based Interface to the Internet*, Communications of the ACM, Vol. 37, No. 7, pp. 72-76, July 1994.

[Gug96] Eugene J. Guglielmo and Neil C. Rowe, *Natural-Language Retrieval of Images Based on Descriptive Captions*, ACM Transactions on Information Systems, Vol. 14, No. 3, July 1996, pp. 237-267.

[Jai88] Anil K. Jain and Richard C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[Lee97] Joseph K. W. Lee, et al., *Intelligent Agents for Matching Information Providers and Consumers on the World-Wide-Web*, Proceedings of the Thirtieth Annual Hawaii International Conference on System Sciences, IEEE, 1997.

[Lew96] D. D. Lewis, R. E. Schapire, J. P. Callan, R. Papka, *Training Algorithms for Linear Text Classifiers*, ACM SIGIR'96, 1996.

[Maa96] Yoelle S. Maarek and Israel Z. Ben Shaul, *Automatically Organizing Bookmarks per Contents*, Fifth International World Wide Web Conference, May 1996.

[Mak85] J. Makhoui, S. Roucos, and H. Gish, *Vector Quantization in Speech Coding*, Proceedings of IEEE, Vol. 73, No. 11, 1985, pp. 1551-1588.

[Sal89] Gerard Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, 1989.

[Sch92] Robert Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, 1992.

[Sch96] Jürgen Schürmann, *Pattern Classification: A Unified View of Statistical and Neural Approaches*, John Wiley & Sons, 1996.

[Zim91] H. -J. Zimmermann, *Fuzzy Set Theory – and Its Applications*, 2^{nd}, Revised Edition, Kluwer Academic Publishers, 1991.