Information Architecture and Agents[1]

George Cybenko
Robert Gray
Yunxin Wu
Alexy Khrabrov
Thayer School of Engineering
Dartmouth College
Hanover, NH 03755

This paper surveys the architecture and properties of intelligent information agents. A number of issues arising in the future design of general, programmable agents are presented. Examples of agent applications are given with some implementations. A formal model for information agents is developed briefly.

[1]Partially supported by AFOSR contract F49620-93-1-0266, AFOSR/DARPA 89-0536, NSF MIP-89-11025 and The Hewlett-Packard Corporation

Contents

List of Figures

List of Tables

1 Introduction

The current growth in networked information resources is radically changing the nature of computing. A decade ago, most machines had limited access to remote data and services. The rule today is networking and connectivity, even for the millions of personal computers in businesses, homes, and schools. Most of this networking is done in a decentralized way, with network topologies and technologies defined locally and transparent to most users. Similarly, the information resources available on the Internet are not managed centrally so users must rely on an increasing number of navigational tools such as Archie, Mosaic, Veronica and DCIS Navigator to find and retrieve networked information resources. The number and variety of resources will multiply dramatically when full motion video becomes networked. At that time, the user base will shift from computer users to the average citizen who has less information and computer sophistication.

This networking imperative has brought about an *informational watershed.* The information poor side of the watershed is characterized by the need for more storage and bandwidth. In order to improve immediate access to data, industry focused on increasing memory densities, disk drive capacities and communications bandwidth. Electronic access to data has grown from a trickle (pre 1980) to a steady flow (1980 through today) to a deluge when broadband networking and the myriad planned services finally get deployed (late 1990's ?). On the information rich side of the watershed, the hard problems shift from bandwidth issues to the identification, retrieval and management of information.

Such problems in managing networked information resources have lead to the creation of new objects of research - *information agents.* Information agents are programs that model a user's information space, managing the resources and taking actions when appropriate. Information agents are not well defined entities. Surely, they are programs and they manipulate information but that can be said of all computer software. Information agents are distinguished by the fact that they operate at high semantic levels relative to typical programs and their basic domain of operation are networked, in addition to local, resources. Like expert systems, information agents are hard to characterize but easy to recognize. However, it is important to recognize a major distinction between expert systems and intelligent agents. Expert systems model an *expert*, a single person or composite, and make the experience and practice of that expert available to many users. Intelligent agents model the *user*, that user's information needs and subsequent actions so each agent must be customized or programmed appropriately. Accordingly, programmability is an essential ingredient of intelligent agents.

One of the goals of this article is to characterize the problems that information agents are being designed to solve and in the process to elaborate generic functionality that most if not all agents have. This leads us to describing a general architecture into which most existing work can be fit. That architecture consists of a number of ingredients which are separately addressed. The architecture itself suggests a formal model for agents which we briefly explore.

Generally speaking, we value information because information reduces uncertainty about some state. Once uncertainty is sufficiently reduced, it is possible to take some action that leads to a new state. This cycle of uncertainty reduction and action can be repeated. We believe that these ideas can be formalized and studied using techniques from computation theory, information theory, statistics, and systems theory.

Through characterization of such problems and their solutions, we hope to be able to identify the specific computational issues that arise in building agents. Those issues and the overall architecture provide a framework around which future research and evaluation can be made. In the past, models such as Turing machines (for general computation) and the PAC model [Val84] (for machine learning) have been valuable as reference models to evaluate and compare progress. We do not assume that we are presenting a model as powerful or as generic as a Turing machine but we do hope that these discussions will lead to efforts in that direction.

This paper is organized as follows. Section 2 presents examples of agent applications. Section 3 describes a general architecture that is common to most agents and can be used as a framework for further research. Section 4 explores a formal model for agents. Section 5 summarizes the article.

2 Examples of Information Agents

As previously noted, information agents are hard to define but easy to recognize. At this time, what is meant by an agent is best conveyed by discussing a number of existing and possible manifestations of them. Accordingly, this section presents examples of current and future agent applications. Each example is a short sketch of the application rather than a complete discussion of the possibilities and issues. The next section will explore the common feature of these applications and develop an agent architecture that can support them. The challenges that face researchers and application developers are discussed there.

2.1 Scholarly Agents

Modern researchers face the daunting task of keeping track of the developments in their field [C$^+$92]. This task is becoming increasingly challenging as the flow of publications and research announcements continues to grow. Fortunately more and more information is becoming available in electronic form. This raises the hope that literature searches over a broader array of resources, not just commercial systems or local library catalogs, can be automated. In this setting an agent operates as follows. The researcher provides a description of his current research interests. An agent accepts this description and searches a collection of information resources for information that is relevant to the researcher's interests. The information resources are likely distributed and heterogeneous. For example most universities now have online libraries. Many researchers make selected articles and technical reports available through anonymous FTP and have HTML pages that can be accessed with Mosaic or other World Wide Web browsers. Many other resource types exist such as WAIS servers, GOPHER servers and NEWS servers. The agent examines the documents contained within these resources and identifies the documents that are sufficiently similar to the research description. These documents can be returned to the researcher or the researcher can simply be made aware of their existence. Such an agent could run continuously and periodically provide the researcher with recently published information.

2.2 Agents in Finance

Information society of the future cannot control its industrial layer with the *ad hoc* search techniques employed now. Amount of the information required to take a decision is growing as the number of constraints and present options does. An example of emerging constraints is the environmental preservation, both warranted and demagogical; new options accompany every political reconstruction, such as that in Eastern Europe today. Although better tuned now when before, *market economy is not perfectly predictable.* It provides basis of operation for different insurance and reinsurance companies, dealing with risks, which is another form of business information handling. Thus information costs are indirectly increased. *Discontinuities of processes* on stock markets also stir public opinion and increase interest in *controlling business environments.*

Monitoring, investment, and *cooperation* are those problems in business which challenge human effectiveness. Mismanagement of business information directly leads to *financial losses*; on the other hand it urges for improvements in *management information systems* (MIS). In MIS information is to be *where required when necessary*, and be *adequate* and *complete* [Kem62]. Business partners have to agree on the format of their data hierarchy, so that agent will move transparently along the whole *production-revenue* cycle. Since agent deals with *the information in a uniform way*, it can present its results in a *highly organized, structural format.*

The first thing agent will do all the time in background is *economic monitoring.* Agent monitors external and internal, *macro* and *micro* economic parameters. Internal monitoring may include links to the company's own inventory database, statistical data, OLTP, etc. Obvious discrepancies with normal expectations, such as an essential difference between the stock decrease and transaction throughput, must accumulate and *warn* the agent's owners, i.e. the human operators, asking them to *converge* the real world and the "virtual" universe of documents. Of these only the latter is available to the agent.There is an inavoidable ***human input*** at some level of the information handling, so the audit function of an intelligent agent will be of great use. In practice there is usually no way for an agent to observe an isolated parameter it monitors and conclude it is unreliable and submit it to the human audit. (Yet it's possible for some elementary data, or when a basic law about the data's origin is violated; when time read by an agent starts going back, the time sensor is probably broken.) It is *correlation of several parameters* which constitutes an economic situation; an "emergency" such as a big debt arises when there are no money at the same time. Correlations of "warning indicators" are studied by Victor McGee at Tuck School of Business Administration ar Dartmouth. It's reasonable if trying to foresee and act prudently, to notice warranted correlation and refer to such a maintained permanently *database of experiences.* A question of *investment* may require an **estimate, list of options,** or **recommendation** as an *answer.* A rewarding thing for the agent to do is to optimize the resources to make *robust bets.* There are three ways to predict outcome, including results of bets, described in [Khr94]. The latter work, performed at Dartmouth, presents a general framework for agent's decision-making.

2.3 Agents in Medicine

2.3.1 Problem area and specification

Health care reform is the social priority. It's an illustration of what is progress considered to be about: information reuse, centralization, standard indexing and online access. All the care providers, patients, lawers etc. are already in the field. The question is, how to avoid wasting effort in creating, maintaining, duplicating, and locating a *Patient Record?* (PR). "A record, if it is to be useful to science, must be continuously extended, it must be stored, and above all it must be consulted" [Bus45]. Consensus is to mold PR into CPR, *Computer-based Patient Record.* Its new format and access techniques will radically change the way how medical personnel *associates* a patient with his acquired medical data. An agent can help to get the most from a CPR in a way which *encourages* CPR technology. It will promote better *navigation, uniform update, event-triggered response, and prognostic assistance.* Also it's supposed to *absorb various secondary functions*, including legal consent, billing scheme ... *ad infinitum.*

2.3.2 Navigation

Full-text retrieval allows for sophisticated correlation schemes, which may help to put many data fragments *together.* It is the case with medical data, often collected by different providers of different institutions. Adverse effects of medications can be exhaustively captured by an agent, while encouraging interdisciplinary research and better cooperation. An allergic patient will be expressly marked as such in any attempt of the navigator to use the allergenes. Cardiovascular patient's animated picture may represent his heart pupming with difficulty, urging the doctor in an intuitive way to alleviate the difficulties. Wide experience of medical data correlation would be helpful in improving *classification of injuries, illnesses and causes of death*, which in turn will simplify the CPR *navigation.*

2.3.3 Uniform Update

When new data about a patient come in sight, it is necessary to make them available whenever the patient is observed. Even before the centralized CPR database, agent may act *proactively* to inform all the imaginable relevant institutions. Here agent will inforce the *uniform update*, preventing access of obsolete data; note that, agent will traverse the sites involved *exhaustively.*

2.3.4 Event-triggered Response and Prognostic Assistance

This reflects the agent's potential in monitoring many input parameters, and correlating them for the sake of identification of a *pattern.* Machine learning may improve this capability of the agent's pattern recognition. “Naturally there is nothing which would be impossible to understood in a long time, fixing in memory and in writing what is already recognized” [Cic64]. Pattern here is a broad-sense combination of events, such as critical sections of EEG, causing a real-time *warning*. A mere urge to get a refill is a possible response triggered by the critical section on the time scale (the prescription had expired). Agent may include *a patient's model* for *virtual treatment*, which can be consulted when choosing the treatment and testing hypotheses through *medical imitation modeling*.

2.4 Industrial Design

Advances in manufacturing productivity are being stymied by the increasing scale and complexity of the products being made. It is well known that significant increases in productivity can be made by component reuse. In the computer software arena software reuse is a major concern that has led to much activity in the study of object-oriented design and other techniques for recycling previous efforts. We describe an example agent application that may be less familiar to the computing community. A similar scenario exists for software reuse.

Large-scale design projects often involve thousands or even millions of distinct components. For example the Boeing 777 contains over five million parts. In many situations engineers design each part from scratch even though equivalent parts already exist. For example industry analysts estimate that up to eighty-five percent of jetliner parts can be taken from previous designs [Boe94]. However most parts are redesigned despite the potential for component reuse. Part of the problem is that many designers are paid per design. There is little incentive for design reuse. However the larger part of the problem is that there are no facilities for efficiently and accurately searching a large collection of existing designs. Existing designs should be collected into one or more online databases. The designer would present a rough description of the needed part to an agent which would then search the databases for equivalent or similar parts. Equivalent parts could be used directly whereas similar parts might need just a small modification. The inherent challenge in realizing such an agent is developing an effective electronic representation of a part. The representation must allow the agent to determine which parts match potentially complex queries. It is easy to imagine a designer making the request ”I need a part that looks about like the provided sketch, weighs less than two pounds, can withstand vertical and horizontal pressure of two hundred pounds per square inch and can be attached with type B fasteners.” Such a request involves both graphical and textual information and several different kinds of part features. [E^+94] discuss other roles that agents can play in collaborative design tasks.

2.5 Organizational Agents

When a document collection becomes large, without *a priori* structure, it is difficult to efficiently manage and comprehend the documents and their interrelationships. Thus automatically defining structure is a desirable function of information agents. Two examples of structuring are automatically adding hyperlinks to a collection of documents and automatically grouping documents into related clusters.

2.5.1 Creating Hyperlinks

Recent work at Dartmouth has focused on the automatic generation of hyperlinks in unstructured text documents. Here a hyperlink is defined as a link from a word or words in one document to relevant information in another document. Unstructured means that the text of the documents does not explicitly state the desired links. In other words the documents do not contain phrases such as ”Refer to the the article on communism for additional information.” Key phrases were identified in each document. Two cases were considered. In the first case each sentence was a key phrase; in the second case each proper noun phrase was a key phrase. A dictionary-lookup heuristic was used to identify proper noun phrases. A latent semantic indexing package was used to determine which documents were relevant to each key phrase. LSI was chosen since it is one of the best *generic* text-retrieval methods [Gup94]. The LSI package assigns a numerical similarity score to each phrase/document pair. The highest scoring pairs become the hyperlinks. This approach generates many useful links in a collection of *TIME* magazine articles but generates many nonsensical links as well. The main shortcomings are that the approach ignores the context of the phrases and does not impose any constraints on the structure of the hyperlink collection. Future work will address these shortcomings. An agent could use this or other techniques to generate hyperlinks in a collection of documents that had been retrieved for user consideration. Other text-analysis techniques for automatically generating hyperlinks are discussed in [Hay88], [Rea91] and [Rin91].

2.5.2 Automatic Clustering

In the *vector space* model of text documents [SM83], points in high dimensional vector space represent documents and the distances between the points are correlated with the similarities between the documents. Using Cluster Analysis methods or Vector Quantization techniques [And73, Kun93], documents with a vector representation can be grouped and a hierarchical cluster structure can be built automatically. This structure is very useful for efficiently searching and browsing the collection. The structure can be visualized in 2D space as an assistant tool for retrieval and browsing. More detailed discussion can be find in Section 3.5.

The automatic clustering of the documents and visualization of the clustering can be based on the latent semantic indexing method [D$^+$90] and have been implemented. An example is grouping personal e-mail letters according to text and not just the headers. The letters of the responses of paper reviewers form a distinct cluster. The grouping of the letters can also be done by using the "subject" information in heads of e-mails, but it is less robust because the senders sometimes omit the "subject". (Actually, the intelligent information agent should be able to use the "subject" and the text comprehensively.) Clustering results of different document collections have a common problem — the clusters can be unbalance. The size (number of documents) of the clusters can be a few hundred or just one. This will compromise the effectiveness of automatic clustering. There are methods for balanced clustering but they will introduce biases — a large but dense cluster may be separated and two small clusters relatively far away may merge. There is a trade-off between balance and unbiasedness. With the concept of *redundancy* in document collections and the corresponding metric that is introduced in separated sections in this article, we find that naturally balanced cluster structures (without forcing balance) indicate good information collections in the sense that they cover the information subspace well with the least amount of redundancy.

2.6 Other Examples

The July 1994 issue of *Communications of the ACM* was devoted to agent research. Several articles discussed existing agent applications. A brief summary of some of these applications is presented here. Pattie Maes from the MIT Media Laboratory presents four "interface" agents – an agent that handles electronic mail, an agent that schedules meetings, an agent that filters electronic news and an agent that recommends suitable entertainment [Mae94]. One of the distinguishing features of these agents is that they gradually *learn* which decisions to make by observing the user, accepting user feedback and receiving advice from other agents. The user feels comfortable with the agents because they adapt themselves to his behavior and take over the decision making task gradually. Such a learning ability will be essential in many agent applications. Etzioni and Weld from the University of Washington discuss a "softbot-based interface to the Internet" [EW94]. The softbot has a logical model of each Internet resource that describes how to invoke the resource and the effects of resource invocation. The softbot accepts a request and uses a goal-oriented planning algorithm to decide which sequence of resource accesses will best meet the goal. Mitchell and several others from Carnegie-Mellon University describe an agent that helps the user to schedule meetings [M$^+$94]. The agent is similar to the scheduling agent of Patti Maes in that it learns the user's scheduling preferences – room, location, duration, importance relative to other meetings and so on – through routine use. Ted Selker from the IBM Almaden Research

Center presents the COACH system which can provide “interactive adaptive computer help [in] text-based interfaces” [Sel94]. The COACH system monitors user actions to create a user model. This model is used to decide when and what kind of unsolicited advice should be offered to the user. Alderman separates the task-specific knowledge from the learning and reasoning algorithms. This allows the system to be easily ported from one problem domain to another. Other articles highlight the use of agents in ”groupware” or group-oriented software and in design tasks [E^+94, Gre94].

2.7 Disk Management Agents

Hierarchical file systems have become the norm today. Their utility in managing large collections of files places the burden of organization squarely on the user who must decide when to extend the hierarchy and where to place files. File naming conventions are idiosyncratic and inflexibly personal. Gigabyte disks are becoming affordable for many users who can easily end up being responsible for managing local, personal disks with thousands of files.

There is a small but growing interest in developing agent-like utilities that can organize and manage a personal file system using content instead of directories and naming conventions. Researchers at Apple Computer have introduced the notion of “piles” which are content based clusters of files dealing with similar subjects [R^+93]. An agent assists users by creating piles for new files when needed, assigning new files to existing piles if there is enough correlation and locating files associatively according to content when the user wants to call up an existing file.

Such an associative file system agent would dynamically create, modify and combine these piles of files, operating in background and possibly adapting to the individual users patterns of file usage.

3 Architecture Overview

The agents presented in the previous section appear to be dissimilar. Many researchers compound this apparent dissimilarity by inventing new terms to describe their particular agents. Readers now see references to “intelligent agents, intelligent interfaces, adaptive interfaces, knowbots, knobots, softbots, userbots, taskbots, personal agents and network agents” [Rie94]. However these agents perform similar tasks. Each agent manages an information space or some portion of an information space. The information space can range from the messages in an electronic mailbox to an electronic conferencing room, from a distributed database of part designs to all of the information resources spread across the Internet, from all the files in a local directory to all files available through anonymous FTP. The agent searches the information space for relevant information, interprets and analyzes the information, presents the results to the user and possibly takes some automatic action on behalf of the user. The automatic action modifies the contents of the information space in some way. This description provides our definition of an agent.

An agent manages an information space or a portion of an information space. It identifies relevant information and possibly takes some automatic action on the basis of that information. The automatic action may modify the contents of the information space.

This view of agents leads to an agent architecture that is organized around information resources. Figure 1 shows the proposed architecture. The architecture addresses two goals. It is common to most agents and divides agents into well-defined components that can be addressed separately. Hopefully the architecture will provide a convenient framework for future research.

Agents are written in a high-level agent specification language. Agent execution begins when an application selects and starts the agent. The application can pass parameters to the agent such as where to search and whether the search should be broad or specific. The application might even pass a description of the desired agent behavior. The agent could then be thought of as an interpreter for

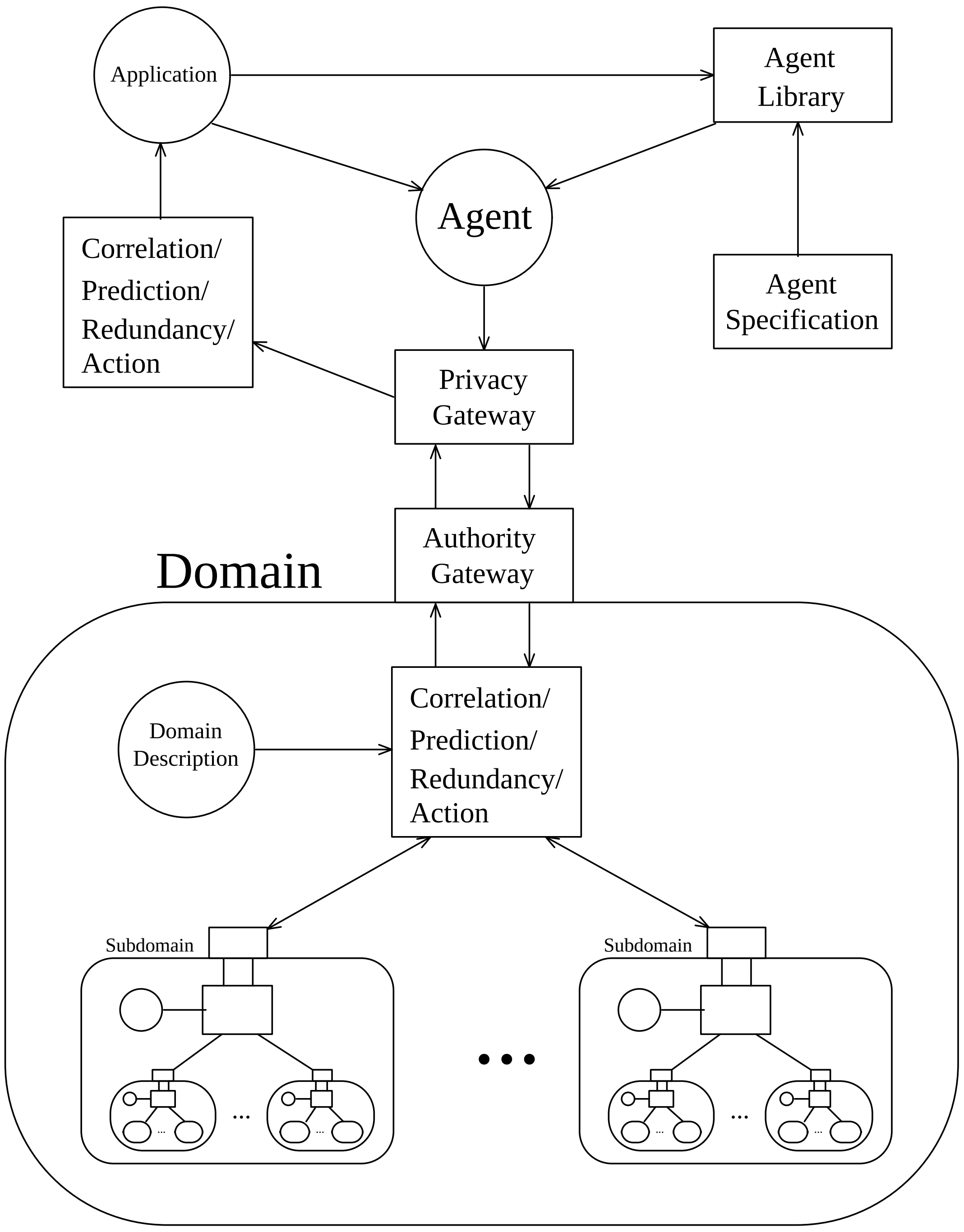


Figure 1: Agent architecture

user-defined agents. This opens up the whole field of end-user programming which is aptly demonstrated by the KidSim environment. The KidSim environment allows children to construct simulations through languageless programming [SCS94].

The agent works with some set of information resources which can be referred to as the domain of the agent. The set of information resources could be large in which case the domain is likely organized into a hierarchy. The agent can hopefully determine the probability that a portion of the hierarchy contains relevant information without examining the complete contents of that portion. This would improve search efficiency since the agent could skip entire sections of the hierarchy.

An information resource maintains some collection of documents and provides an interface for accessing those documents. A document is any set of data and can be represented in various ways. Different searching and decision making tasks call for different representations. Resources include "traditional" entities such as files, FTP servers and pages on the World Wide Web and include "nontraditional" entities such as a human user, other agents and the application that started the agent. The documents associated with nontraditional resources might not physically exist. Instead the documents are the set of possible user, agent or application responses to agent requests.

The agent searches the selected domain for documents that are relevant to its task (*correlation*). The agent can then make a decision based on the information (*prediction*. Finally the agent can take some automatic action on behalf of the user. Most agents will perform multiple correlation, prediction and action steps over the course of their execution. In particular many agents will perform a correlation, prediction and action step at each level in the hierarchy as they combine the information that came from multiple information resources. As the agent performs its tasks it must recognize that a document is useful only if it provides the user or application with *new* information. Relevant information is not useful if the user or application already has the same information in other forms. The agent must be prepared to employ some method of redundancy control so that it does not waste time, finances and other resources in identifying and retrieving information that has already been provided.

The final components of the architecture provide for authority and privacy. Authority is the problem of denying unauthorized access to an information resource. Authority can be divided into static and dynamic cases. The static case deals with access permissions that are determined by who owns the agent. The dynamic case deals with access permissions that are determined by how much the agent pays for the desired information. Privacy is the problem of preventing malicious users from examining an agent and its output. The information implicitly and explicitly contained in the agent and its output can be used to build a partial model of the agent's owner. This model could then be used towards unethical ends. Privacy is more than a matter of denying unauthorized access to the agent since many agents will execute on remote machines and most agents will obtain documents from remote resources. There is no way to prevent the owner of these remote machines and resources from examining the agent and the documents that the agent requests from the resources. A means is needed for hiding the identity of the agent's owner while providing a way for information providers to perform authentication checks and send bills and other information to the agent's owner.

The following sections elaborate on this architecture and highlight the relevant research issues. There is one section for each of the major architectural components.

1. Document representation
2. Domain
3. Agent specification
4. Prediction
5. Correlation
6. Redundancy
7. Authority

8. Privacy

9. Interface - i.e. the application that uses the agents.

A tremendous amount of work has been done in addressing most of these architectural components. However much of this work has not been in the context of agents. The challenge is not only to develop new solutions but to identify the existing work that can be used to realize each architectural component. We highlight some relevant existing work but do not attempt a complete literature overview.

3.1 Representation of documents

A document is any set of data. Documents can be traditional text documents, audio samples, full motion video, static images, readings from a pressure sensor and so on. Any document can be represented in several ways. For example an image can be represented as a collection of monocolor segments or as a collection of histograms. The choice of representation affects how easily a correlation agent can find relevant documents and how easily a predictive agent can make decisions based on those documents. Representations can be divided into two classes. Lossless representations describe the original document completely in the sense that the document can be recreated perfectly if only the representation is known. Lossy representations do not describe the document completely but instead can be viewed as a summary of certain features of the document. Generally lossy representations should be more compact than the document that they describe.

Different applications and document types call for different representations. However any representation can be regarded as a vector. Each element of the vector describes some feature of the document. The choice of features depends on the type of document and the information that the representation is supposed to convey. The features do not have to be numeric. There are three different operations that can be performed on these feature vectors.

1. *Create* - Construct a feature vector that represents a given document. This operation is often referred to as "extract" since features are being *extracted* from the document.

2. *Transform* - Transform one representation into another without examining the original document. The input to the transform operation is a feature vector that represents a document. The output is a feature vector that represents the same document in a different way. For example a feature vector that gives the size of each monocolor segment in an image can be easily transformed into a feature vector that gives the number of pixels of each color. Other conversions are more difficult or even impossible if the input vector does not contain enough information. For example the reverse transformation from pixel counts to segment sizes is impossible. In such cases it would be necessary to refer back to the original document and extract the desired feature vector.

3. *Compare* - Compare two feature vectors and provide some numerical measure of their similarity. Each compare operation might return a real number between zero and one. Zero means that the two feature vectors are highly dissimilar and one means that the two feature vectors are highly similar. The similarity of the feature vectors should indicate the similarity of the original documents. The two vectors might not have the same type in which case a create or transform operation must be performed before the compare operation. For example a user might want to compare two images that are represented in different ways. One representation would be transformed to the other. Less obvious but more important is that a user might want to compare two entirely different kinds of documents. For example suppose that an application wants to find all images that match the mood of a particular piece of music. Some representation of the music would be transformed into a representation of a prototypical picture that matches the mood of the music.

Figure 2 illustrates the three operations. Different extract, transform and compare operations can be collected into one or more libraries. It is easy to imagine one library that contains operations for working

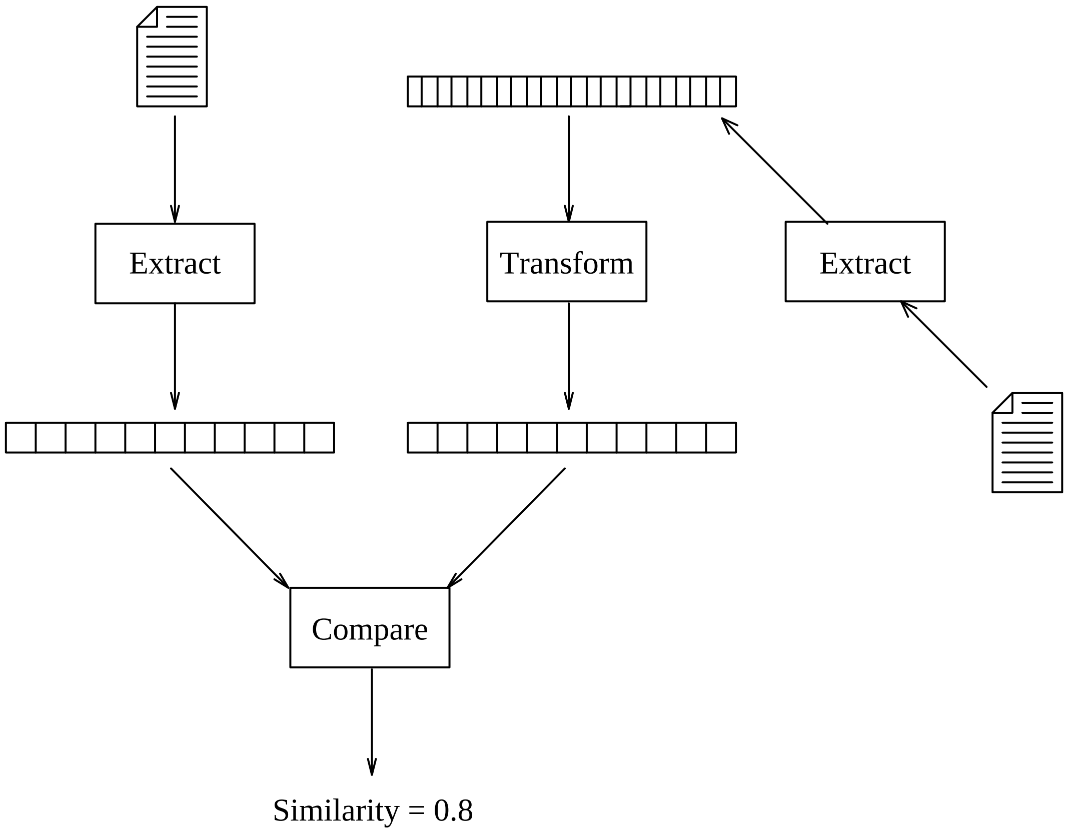


Figure 2: The three operations that can be performed on feature vectors – extract, transform and compare

with straight ASCII text, a second library that contains operations for working with static images and so on. The application developer uses the libraries that are appropriate for her agent.

The feature vector model is similar to the vector space models that have been proposed throughout the corpus of information retrieval literature. Vector space models have been discussed most extensively in the context of text retrieval. [Sal73] and [SM83] discuss the SMART system at Cornell which represents documents as term-frequency vectors and uses the cosine of the angle between vectors as a measure of document similarity. [D+90] and [Gup94] discusses latent semantic indexing which uses the same cosine measure as the SMART system but first performs a singular value decomposition in order to reduce the dimension of the term frequency vectors. Vector space models have not been discussed as extensively in the context of image and audio retrieval. However most information retrieval work implicitly assumes a feature vector model. Each document is represented by some set of features. Each feature is a numerical measure, a keyword or an expression. Comparisons between sets of features are used to determine document similarity.

The previous discussion leads to several research issues.

1. Which representation is best for a given document type and application? Much work has been done in this area. [SM83], [D+90] and [Gup94] explore various term-frequency representations for straight text. The recent IEEE *International Conference on Multimedia Systems and Computing* highlighted several different ways of representing images and text. Two of the more interesting papers presented were [G+94] and [RWG94]. [G+94] discusses an image database system that represents images as both collections of monocolor segments and a collection of histograms. [RWG94]

discusses algebraic video in which an algebraic expression is used to represent the content of a video stream. Unfortunately work in this area tends to require a case-by-case approach. The most interesting work would be to develop compact representations that are highly descriptive and applicable in a wide range of applications.

2. Which algorithms should be used to extract a representation from the original document? This area requires a case-by-case approach since any extraction is highly dependent on the document type and the desired representation. The extraction algorithm must identify all of the desired features in the original document. The development of extraction algorithms is closely related to the selection of a good representation since a representation is *practical* if and only if there is an efficient algorithm for constructing that representation.

3. Which algorithms should be used to transform one representation into another? This area also requires a case-by-case approach since any transformation is highly dependent on the input representation and the desired output representation. The most interesting work would be to identify highly descriptive intermediate representations. A transformation could then be implemented as two subtransformations. The input representation is transformed to the intermediate and then the intermediate is transformed to the desired. Intermediate representations reduce the total number of transform operations at the possible cost of more work per operation.

4. How does a compare operation assign a similarity score to a given pair of feature vectors? This is the most important area and has seen a substantial amount of work. One common technique is to interpret the cosine of the angle between two feature vectors as a measure of their similarity [D^+90], [Sal73], [Gup94]. This technique has the advantage that it can be used in a wide range of applications. It is particularly effective when comparing term-frequency vectors. However this cosine measure does not provide good results in all applications. It certainly does not work if the feature vectors are non-numeric. Therefore different similarity measures must be developed. The challenge in this area is to develop similarity measures that are general enough to be used in a range of applications but specific enough to give good results for each application. One possibility is parametric similarity measures where the parameters can be adjusted for each application. Similarity measures are discussed further in the correlation section.

3.2 Domain

An agent makes use of some collection of information resources. An information resource maintains a collection of documents and provides an interface for accessing those documents. The interface can range from nonexistent to complicated enough that we might want to talk about "intelligent" resources. Two broad kinds of information resources can be distinguished.

1. A resource that allows direct access to its documents. An agent is free to examine the documents and perform any processing that it desires. The resource interface might be essentially nonexistent or might provide only a few operations such as billing services. Most local resources – especially resources that belong to the agent's owner – fall into this class. Examples include a single file, all the files in a certain directory and all the messages in a certain mailbox.

2. A resource that does not allow direct access to its documents or does not contain any documents in the traditional sense. All access to the documents is through the operations provided in the interface. These operations might include query operations that return all documents relevant to a set of input vectors, suggested compare and extract operations and billing operations. Most remote resources fall into this second class of resource. Examples include online libraries, WAIS databases and files available through anonymous FTP. However it is essential to note that this class also includes such resources as other agents and human users (or more precisely applications that interact with human users). The documents associated with these resources may not physically

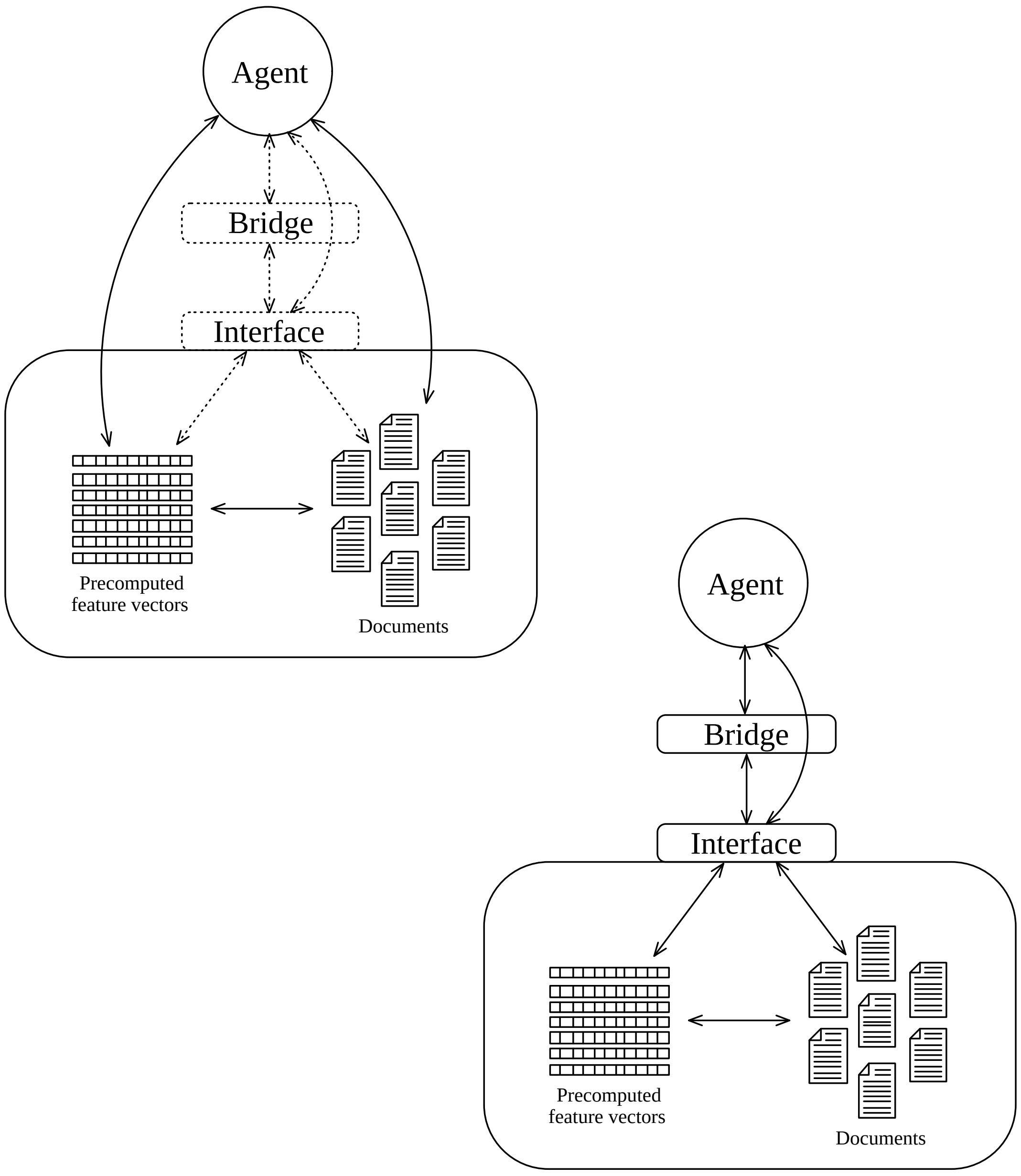


Figure 3: The two kinds of information resources. The first kind allows direct access to its documents and often has an essentially nonexistent resource interface. The second kind does not allow direct access to its documents. Instead all access must be through the interface.

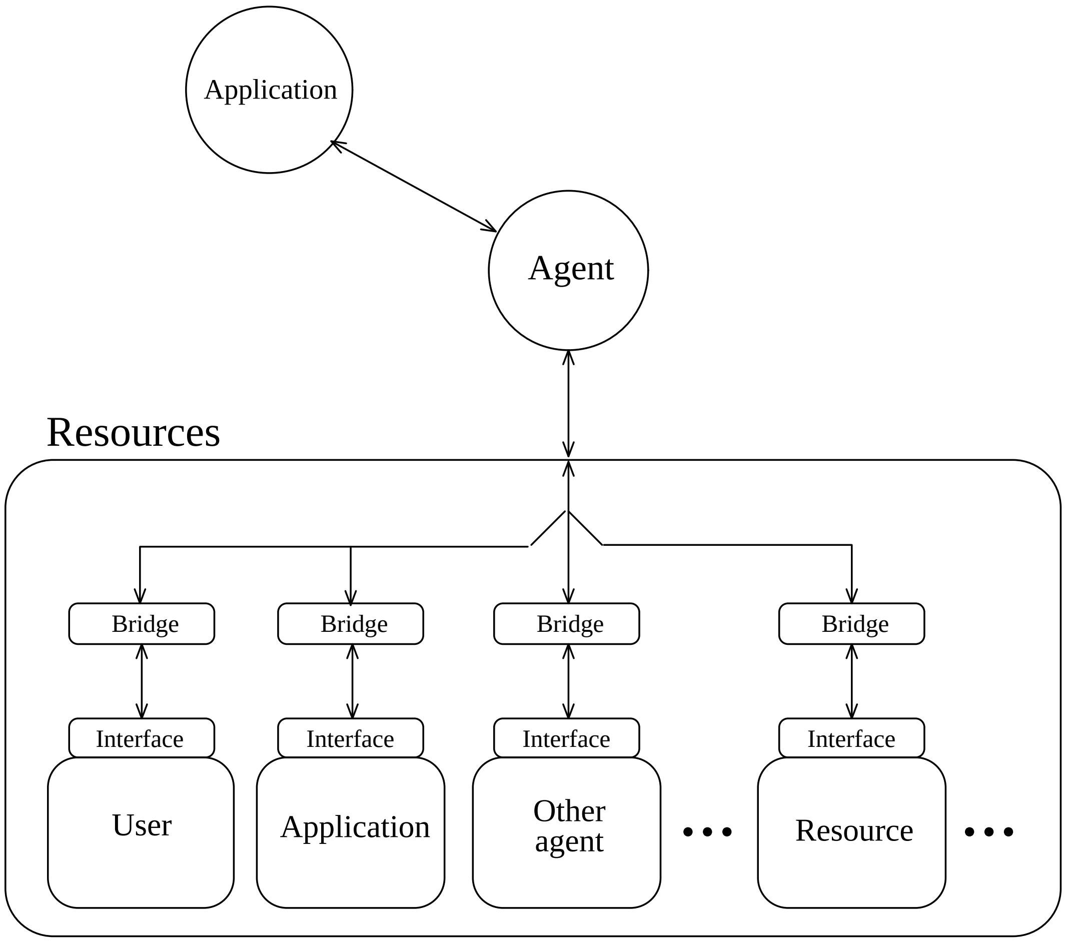


Figure 4: A simple view of an agent architecture. An application starts up an agent which then interacts with a range of resources.

exist. Instead the documents are the set of possible user responses and the set of possible outputs from another agent. Figure 3 shows the two kinds of resources. Figure 4 shows a simple view of an agent architecture. An application starts up an agent which then interacts with a range of resources.

Most resources will have different interfaces. It is necessary to develop a standardized interface in order to simplify the task of searching a heterogeneous collection of information resources. The standardized interface or *bridging model* is built on top of each resource interface. Requests made to the standardized interface are translated into requests that the resource interface understands. Agents would use the standardized interface whenever possible instead of accessing the resource interface directly. The main issue is which operations should be included in the standardized interface. It is not necessary to support every conceivable operation. Agents with unusual or specific needs can bypass the standardized interface and interact directly with the resource interface. Thus the standardized interface is simplified at the expense of incorporating a high level of resource awareness into some agents. Since most agents are primarily concerned with searching each resource for relevant information, a simple and reasonable approach is to organize the standardized interface around a query operation. The query operation would accept some set of feature vectors as input and find all documents within the resource that are relevant to the set. Note that the resource could be an application or another agent.

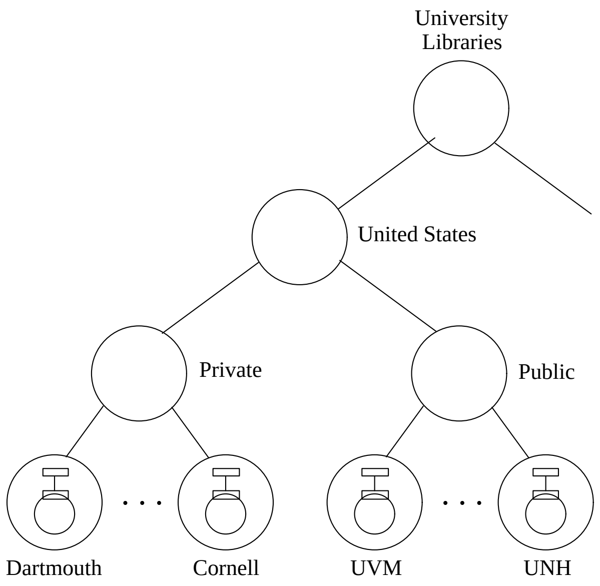


Figure 5: A simple domain hierarchy that organizes online university libraries

There are other possibilities for the bridging model as evidenced in recent work. [EW94] discuss a softbot-based interface to the Internet which uses a logical model of each Internet resource. The logical model specifies how to invoke the resource and what happens when the resource is invoked. The bridging model could be a set of these logical resource models. [GK94a], [FW91] and [G$^+$92] discuss the agent communication language KQML and its subcomponent KIF. KQML is a standardized language that autonomous software processes can use to exchange information and requests. The bridging model might translate some subset of KQML into commands that can be applied directly to the resource interface. It could be argued that there is no need for two levels of interface. Since the bridging model is a standard, the resource interface should be a direct implementation of the bridging model. Certainly the interface to many new resources will be a direct implementation of the bridging model. However we must not ignore the tremendous amount of electronic resources that are already available. In addition there are likely to be multiple "standards". It is more realistic to assume that each resource provider will design their own particular interface and implement the more popular bridging models on top of that interface.

The number of information resources available to a particular agent is vast. Thus one or more domains are associated with the agent. Each domain specifies some set of information resources that are relevant to the agent's task. There are three ways that a domain can be specified. (1) An application can pass the domain to the agent as a parameter; (2) the domain can be hardcoded within the agent or (3) the agent can employ some algorithm to discover the domain on its own. An agent might make use of any combination of these three methods. The third method introduces the possibility of the user or application developer partially specifying a domain and then relying on the agent to "fill in" the remainder of the specification.

Many agents will *search* a domain for relevant information. However it is infeasible for an agent to search every resource in a large domain. In addition it is a burden on the user or application developer if a domain must be specified by listing every resource in that domain. For the sake of efficiency and user convenience the world should be divided into a domain hierarchy. Each domain within the hierarchy either contains a single resource or contains a collection of subdomains. An agent would search every subdomain within the specified domain and accumulate the results. Searching a subdomain would consist of searching its subdomains. The recursive descent stops when the agent reaches a domain that contains a single resource. Figure 5 illustrates a simple domain hierarchy for electronic university libraries. University libraries can be divided into geographic regions. Each geographic region can be divided into public and private institutions. Finally the universities themselves appear at the bottom level of the hierarchy. The information resource associated with each university is that university's online library. Such a hierarchy will clearly make domain specification easier. It will make a search more efficient only if the agent can determine whether a subdomain contains relevant documents without examining every document within that subdomain. This would allow the agent to skip entire sections of the hierarchy. Search efficiency and other issues are discussed in the following paragraphs.

1. How is the hierarchy specified? A name and description should be associated with each domain in the hierarchy. For domains that contain a single resource the description would include at least the network locations of the bridging and resource interfaces. For domains that contain a collection of subdomains the description would include at least the network locations of the subdomain descriptions. Building the domain hierarchy is a matter of constructing the appropriate domain descriptions.

   Each domain name would either explicitly specify the network location of the domain description OR would be a symbolic name that a server could map to a network location. The latter is more reasonable in terms of user and developer convenience. Each domain description should have an owner that will maintain the description. This owner might just be the owner of the agent since it is reasonable to allow users to create their own domains if the appropriate domains do not exist. Other descriptions would be owned by the resource providers or the organizations for whom the users work.

2. How is the agent's domain specified? An agent works with a collection of information resources. This collection can be thought of as some combination of the domains in the domain hierarchies. Ideally the method of specifying the collection should be independent of the particular task being performed. This introduces the possibility of a logical system in which domain names are connected with operators such as AND, EXCEPT FOR, AND THEN and so on. The AND THEN operator would be used to express temporality since the agent's domain might change over time. Each occurrence of AND THEN would have an associated time duration. The agent would map the domain names to the network locations of the domain descriptions as discussed above.

3. How can the agent avoid searching the entire domain? The description of each domain should contain a summary of the documents that are in the domain. The agent would examine the contents of a domain only if the summary indicated that the domain contained relevant information. Such a scheme is reasonable only if the summary provides an accurate view of the contents of the domain and the summary can be examined significantly faster than the contents of the domain. Therefore we need a highly compact summary that is general enough so that an agent does not miss relevant documents and specific enough so that an agent does not waste its time searching a domain that does not contain relevant documents. It might be impossible or undesirable to associate a summary with certain domains in which case the agent would have no choice but to precede to the subdomains. The development of an appropriate summary structure might be the most important issue in the domain area. The summary structure is simple in some instances. For example suppose that each information resource supports only keyword search. The summary associated with each resource would be a list of the keywords for which the resource contains information. The summary associated with each node in the hierarchy would be the union of the summaries associated with the node's children. More realistic situations will require more complicated summary structures.

4. Where should the search be performed? An agent examines both documents and domain descriptions as it searches for relevant information. There are two choices that need to be made. The agent can be transmitted to the machine on which the documents reside or the documents can be

transmitted to the machine on which the agent resides. Similarly the agent can be transmitted to the domain descriptions or the domain descriptions can be transmitted to the agent.

It is often more efficient to transmit the agent to the documents. This is especially true when the agent needs to examine a large document collection and there is either no resource interface or the interface does not provide an operation that exactly matches the agent task. In the first case – if agents are not allowed to migrate – feature vectors for every document must be transmitted to the agent. In the second case the agent must decompose its task into the operations that *are* provided in the interface and make multiple remote calls. Each call brings more intermediate data across the network. This intermediate data represents a significant waste of bandwidth if it is not useful beyond the end of the agent's task. Transmitting one feature vector per document is even worse since most documents will turn out be irrelevant. Transmitting the agent to the document collection avoids this communication overhead. The agent executes local to the document collection and returns only the final result. However transmitting an agent to a remote machine introduces numerous complexities. The agent must either precede from machine to machine in round-robin fashion or run multiple copies of itself on each machine which raises the specter of the Internet worm [Spa89]. The agent must be able to run on different machine architectures since there is no guarantee that the domain is contained within a uniform architecture. Privacy and authority become more complicated since remote machines must be protected from malicious agents at the same time that agents are protected from inappropriate observation.

These issues have been addressed in the context of ***transportable agents***. A transportable agent is a program that can migrate from machine to machine in a heterogeneous network [KK94]. The program chooses when and where to migrate. It can suspend its execution at an arbitrary point, transport to a new machine and resume execution on the new machine. Transportable agents were developed as the next step in the remote procedure call [BN84], remote evaluation [SG90] and SUPRA-RPC [Sto94] hierarchy. Transportable agents have several advantages over these paradigms. Transportable agents consume fewer network resources since intermediate data does not need to be brought across the network *and* agents can communicate easily among themselves. Remote evaluation and SUPRA-RPC allow subprograms to be sent to the remote machine but these subprograms are anonymous entities. There is no convenient way for one subprogram to communicate directly with another which makes it difficult to share partial results. At best the client must collect partial results from the remote subprograms and then distribute the results. This requires twice as much communication. In addition transportable agents are a convenient paradigm for distributed computing since the communication channels are hidden but the location of the computation is not [JvRS95]. This makes transportable agents easier to use than low-level facilities that require the programmer to handle all communication but more powerful than schemes such as process migration in which the *system* decides when to migrate a program based on a fixed set of criteria [SS94]. Most existing transportable agent systems provide the *go* instruction which migrates the agent to a new machine. The statement after the *go* is executed at the destination machine. The details of packaging and transmitting the agent are hidden. Transportable agents are advantageous even when views as just an extension of the traditional client/server model. They allow clients and servers to program which other which greatly extends the functionality that application and server developers can offer to their customers. In addition applications can dynamically distribute their server components on startup [JvRS95]. When used to their full potential, transportable agents replace the client/server model with a peer/peer model. Agents interact as peers and are not divided into the fixed roles of client and server. The peer/peer model offers much more flexibility when developing distributed applications [Lew95]. Finally transportable agents do not require a connection between the local and remote machines which – in combination with their low use of network resources – makes them ideally suited for mobile computing [Whi94]. Mobile computing is characterized by low bandwidth, high latency and periods of disconnection from the network.

There are several transportable agent systems although most have not progressed beyond the prototype stage. However Telescript is a commercial product from General Magic that is being used in the new AT&T PersonaLink network [Rei94]. Telescript is the most robust and secure system. Telescript agents are written in an object-oriented interpreted language. An interpreted language is used so that agents can migrate easily among heterogeneous machines and to provide an extra layer of security between the agent and the bare hardware. The language includes the special instruction *go* which transports the agent to a new machine. Each machine runs a server that accepts and executes incoming agents. One of the focuses of Telescript is security. Agents have both *credentials* and *permits*. Credentials are cryptographic signatures that a machine uses to authenticate the identity of the agent's owner. Permits specify whether an agent can perform a certain operation and how much resource an agent can consume. For example there is a permit that specifies a maximum agent lifetime. Permits are *negotiated* between an agent and every machine that it visits. Despite its use in a commercial system Telescript has several weaknesses such as privacy concerns and inefficient execution of agents. A project is underway at Dartmouth to develop a transportable agent system that addresses these issues [Gra95]. This system is based around the the Tool Command Language (Tcl) which is a popular string-based scripting language.

It is likely that a successful implementation of the information architecture will support *transportable* agents. Such agents are more efficient – especially for distributed information retrieval in which large volumes of data must be examined at each site in unpredictable ways – and can make it much easier to write distributed applications. However transportation should not be overused. As opposed to the documents or feature vectors for example,it makes sense to transmit the domain descriptions to the agent since (1) there are relatively few descriptions per domain and (2) the descriptions can be cached on the home machine. Caching the domain descriptions is reasonable since an application or user is likely to search the same domains over and over again. In addition domain descriptions should change relatively slowly since they represent a document collection as a whole. This means that they can be kept in the cache for a long time before being thrown out due to age. Alternatively it takes an insignificant amount of bandwidth to verify that the cached domain description is the same as the remote domain description. A reasonable caching scheme will consume far less bandwidth than migrating the agent from domain description to domain description.

5. How can the agent discover the domain if the application developer or user does not specify the domain or does not completely specify the domain? This is a resource discovery problem in which an agent must identify the resources that are relevant to its task. The three general approaches are directory assistance, distributed object managers and automatic brokers. Directory assistance maps a description of the desired service to one or more resources that can provide the service. An example is the X.500 directory service. An X.500 directory is a distributed hierarchical database. Each object in the hierarchy is a collection of named fields that describe the object. These objects are stored in a collection of distributed servers. X.500 searches can access multiple servers. Objects can be replicated or shadowed to other servers in order to improve search performance. X.500 can provide directory for most applications although the object fields will differ from application to application. For example Aramino has developed a directory that contains meta-knowledge about online bibliographic databases. The directory is searched to identify the bibliographic databases that are most likely to contain the answer to a user query. These databases are then searched [BJR94]. Distributed object managers do not support resource discovery per se. Instead they provide transparent access to a distributed collections of objects. Message can be sent to an object without specifying its network location. The messages are automatically routed to the appropriate site. Distributed object managers include CORBA and DSOM [Lew95]. Distributed object management is a critical adjunct to directory assistance since it can provide transparent access to the resources that have just been discovered. Automatic brokers combine directory assistance with distributed object management. They identify an appropriate resource and then

provide transparent routing of messages to and from the resource. Automatic brokers include ToolTalk and the Publish and Subscribe Service on the Macintosh [GK94b].

Genesereth extends the automatic broker scheme in his work with KQML [GK94b]. Genesereth proposes a federated architecture in which each agent is associated with a facilitator. Agents communicate only with their facilitators. Facilitators communicate with each other. Agents post their capabilities and application-specific facts to their facilitator. A capability is a description of a service that the agent is willing to provide to the outside world. Capabilities and facts are expressed in KQML. When an agent needs information, it sends a request to its facilitator. The facilitator performs backwards inference on its collection of capabilities and facts in order to find an answer to the request. The inference process could involve decomposing the request into subrequest and invoking agents to handle each subrequest. The advantage of the facilitator approach is that each agent communicates with a system agent that appears to handle all requests itself. The main concern with the facilitator approach is *scalability* – i.e. the cost of the inference process and the size of the facilitator's knowledge base. These problems are addressed by limiting the amount and kind of information that each facilitator stores internally. The features that distinguishes this federated architecture from traditional directory assistance and automatic brokers is the amount of processing that is done in the facilitator. The facilitator engages in automated reasoning rather than lookup, pattern matching or standard database and text searches [GSS94].

The command thread in all of these approaches is that resource discovery is reduced to a search problem. The agent implicitly or explicitly searches a well-known resource in order to identify additional resources. This search can be performed by the agent itself or by a third party that provides directory assistance or automatic brokering. For example an agent might note that an e-mail message from a certain person contains an important keyword and add the person's World Wide Web page to its list of resources when it searches for relevant papers. On the other hand the agent could contact a directory assistance service in order to identify researchers in the specified field and then focus on these researches. The search can be general or application-specific and can be as complex as the agent desires. In short resource discovery for a particular application is nothing more than the problem of providing an appropriate database of resource descriptions and an appropriate search mechanism. The challenge is to develop resource descriptions and search mechanisms that can be used in a range of applications. In addition these resource descriptions could be distributed across multiple network sites such that an agent discovers more and more of the domain as it proceeds from site to site. Since it is likely that multiple copies of the agent are running simultaneously in order to gain the efficiency of parallel execution, we must make sure that the copies do not discover and search the same resources over and over again.

3.3 Task Specification

Information agents are models of their creator's or user's information management and organizational processes. Therefore, they have to be able to deal with the idiosyncrasies that arise in the way different individuals become accustomed to performing information organization. To have this flexibility, agents must be programmable or at least customizable. We will discuss some aspects of agent specification that should be defined for possibly fixed applications but for different users. In addition to these detailed parameter-like specifications, agents will require a general purpose programming language to allow developers, if not end-users, to program agents. Moreover, this list is not complete and other specifications are definitely conceivable for different applications.

- Domain specification – This is some description of where the agent must search for relevant data. The domain may be temporally indexed meaning there is some sense of time associated with the search wherein certain sources are to be search at certain times.

- Cost – If searches and data cost money, what amount is the agent authorized to spend? How should that cost be allocated among the possible sources?

- Information description - How is the requested information described? How is its form specified? Specifically: text, images, graphics, tables, charts, video, sound.
- Relevance threshold – How is relevance measured and specified? This allows for varying between a broad and very specific interpretations of the retrieval task.
- Redundancy measure – How is redundancy measured and specified? That is, how can agents be instructed to retrieve documents that contain information novel with respect to some existing or specified corpus of information?
- Action - Should the agent identify information sources and actually retrieve them? How are automatic actions specified to an agent?

None of these specification issues appear to have been treated in any systematic manner in the literature. All of them involve interface work but more importantly, they all require some basic research on quantifiable specifications for an agent's task. In the information retrieval literature, the information content aspect has been addressed only. We are not aware of any models or implementations at all for agents to "shop" for information or otherwise retrieve information that best compliments an existing corpus.

Some commercial collaboration enhancement software systems have a forms-based interface with a scripting language available to the user [lot93].

3.4 Predictive Agents

By prediction, we mean tying the operation and activity of an agent that gathers information to an application that can use the resulting information in some anticipatory way to assist the user. At risk of belaboring the obvious, information is valuable because it reduces uncertainty about the present or future. In some applications, such as financial forecasting and program trading, the link between gathering information and using it in a predictive model is well established. A different kind of simple example is to retrieve weather predictions, parse them and embed them into a calendar/appointment book. If rain is forecast, then generate a message that certain measures should be taken, such as carrying an umbrella or cancelling an outing.

Most information retrieval research has addressed the retrieval aspects solely. That is a complex and nontrivial problem in and of itself without the added complication of incorporating a model of how the actual information will be useful. In many cases, such models are nonexistent or primitive. In domains governed by widely accepted laws or equations (such as weather modeling), there is at least a framework that is not controversial or subjective. Other applications, such as decision support, can vary from the fairly routine (inventory control and management in retail sales) to the very complex (battlefield tactics and strategy). Some predictive applications (such as prognosis of illnesses) may not even have models that are quantifiable in terms of rules or laws yet and must rely on purely statistical techniques. We believe that the integration of information resources with applications that use that information to automatically model a complex system and act accordingly is the Holy Grail for information agent research. It may be too ambitious to do at present for hard, real-world problems but should be a goal for agent research.

agent.correlation.tex

3.5 Correlation

Feature selection extracts the useful information of the documents and abstracts it into vectors. Feature selection is a difficult and application specific problem so that text document features will be completely different from image or video features. Nonetheless, the goal of feature extraction is typically the same – namely to identify salient features of an object that can be used to meaningfully but efficiently represent a document. With numerical feature vectors, the detection of relations between documents can be obtained by comparing the feature vectors of the documents. There may be other useful relations between the documents, not only similarity relations. For example, the task of an information agent may not be to find the most similar documents to the current document collection, but to find the ones that will increase the amount of the useful information most in the collection (this is redundancy which is discused below). In a multimedia document collection, comparison may need to be made between different media, so the comparison may occur in different feature vector spaces. The nature of the required comparison will decide the features to be selected. So each document may associate with several different feature vectors for different comparison purposes.

3.5.1 Quantitative Measures

Once a quantification for documents is selected in the form of feature vectors, it is necessary to determine some metric or correlation measure based on those vectors. Comparisons between sounds or images in restricted domains such as speech recognition and optical character recognition are well studied and use powerful pattern-recognition methods. They have been well studied with statistical models and signal processing methods and many efficient tools have been developed for them. The comparisons between text documents appear to be much harder in the sense that the semantic meanings of texts is important. Some basic ideas and methods in signal processing area may help us to find more efficient comparison methods than currently existing. Below is an example that demonstrates the potential of statistical model and signal processing methods in text processing area.

3.5.2 Text Correlation

In the vector space model of text representation [SM83], a document is regarded as a set of terms. The features are the term frequency counts in the documents. Feature vectors are formed whose components are the frequencies of different terms. Term weighting may use to improve the effectiveness of this representation. A similarity measure is commonly obtained by computing the cosine of the angle between two feature vectors x and y using the following formula:

$$s = \frac{x^T y}{||x||\,||y||} \tag{1}$$

To improve this measure, we look at methods form the signal processing area. There, different features are regarded as random variables and usually considered to be correlated to some degree; hence we can not consider them in isolation but must evaluate them in combination. The covariance matrix is used to improve the comparison between different feature vectors.

In the vector space model, the features are term frequencies which are correlated. Rather than covariances, correlation coefficients represent the inter-relationships between the terms. The computation of the correlation matrix is straightforward. Regarding the term frequencies in each document as random variables, let the number of documents in the collection be k and the frequency of term i in document j be f_{ij}. To simplify notation, let μ_i be the mean of all f_{ij} for term i:

$$\mu_i = \frac{1}{k}\sum_{j=1}^{k} f_{ij}$$

and let $F^* = \{f^*_{ij}\}$ be a matrix that has rows with zero means and identical unit norms:

$$f^*_{ij} = \frac{f_{ij} - \mu_i}{\sqrt{\sum_{j=1}^{k}(f_{ij} - \mu_i)^2}} \tag{2}$$

Then the correlation matrix can be computed by:

$$C = F^* F^{*T} \tag{3}$$

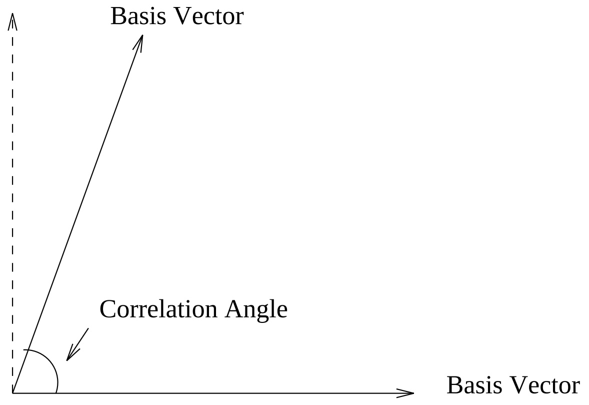


Figure 6: The basis vectors are not orthogonal.

We can interpret the correlation matrix in the following way. In the feature vector (document vector) space, the basis vector e_l corresponds to the single term l. The correlation matrix is not the identity matrix meaning that the basis vectors are not orthogonal to each other. The cosines of the angles between them form the correlation matrix. See 6. With this interpretation, the cosine of the angel s between two document vectors x and y should be computed by:

$$s = \frac{x^T C y}{\sqrt{x^T C x \, y^T C y}} \tag{4}$$

Using (3),

$$s = \frac{(F^{*T} x)^T F^{*T} y}{||F^{*T} x|| \, ||F^{*T} y||} \tag{5}$$

If we let C be the covariance matrix and not the correlation matrix, only (2) needs to be modified to a simpler form (delete the normalization operation):

$$f^*_{ij} = f_{ij} - \mu_i \tag{6}$$

Interpretation of the covariance matrix is the same as the one of the correlation matrix except that the basis vectors are weighted so that terms with high covariance are more important than those with low covariance. The effect is similar to term weighting methods used in information retrieval systems which assign large weight to terms that occur frequently in particular documents. Since term weighting methods can improve the system performance, using the covariance matrix rather than the correlation matrix seems to be a good choice.

It can be shown that the Latent Semantic Method (LSI) based on SVD [D+90], which has been proved to have better performance than simple cosine measure using (1), can be formulated in a form similar to (5) where F^* should be substituted by the reduced left singular matrix (term matrix) T.

The following is a demonstration of the above assertion. The term-document matrix X can be decomposed into

$$X = T_0 S_0 D_0^T$$

and X is approximated by X' where

$$X' = TSD^T$$

Here T and D are dimension-reduced left and right singular matrices (Term Matrix and Document Matrix). The similarity measure between document vectors x and y is

$$s = d_x S^2 d_y^T$$

d_x and d_y are columns of the right singular matrix D (right singular vectors) corresponding to x and y, so we have:

$$x' = TSd_x^T$$
$$y' = TSd_y^T.$$

Multiply the above equations by T^T, to get

$$Sd_x^T = T^T x' = T^T x$$
$$Sd_x^T = T^T x' = T^T x.$$

The right hand sides of the above two equations hold because x' and y' are projections of x and y onto subspace spanned by T. Thus

$$s = \frac{(T^T x)^T T^T y}{||T^T x||\,||T^T y||} \tag{7}$$

Letting $C = TT^T$, we have

$$s = \frac{x^T C y}{\sqrt{x^T C x\; y^T C y}} \tag{8}$$

which is the same as (2).

T and F^* have two basic properties in common:

- both T and F^* are linear transformation of matrix F;
- rows of F^* have zero means while the rows of T have *approximately* zero means.

The only thing not quite obvious here is that rows of T have *approximately* zero means. To prove this, we construct vector a with all its entries be one, i.e., $a = \ (1, 1, 1, \ldots)$.

Suppose $T \in R^{m\times k}(m \geq k)$ and $a \in R^k$. Let $\mu \in R^m$ be the mean vector of the column vectors of T. We have

$$\mu = \frac{Ta}{k}$$

Because the columns of T are orthonormal vectors, $||Ta|| = ||a||$. Thus

$$||\mu|| = \frac{||a||}{k} = \frac{1}{\sqrt{k}}$$

Clearly, no component of μ can exceed $\frac{1}{\sqrt{k}}$ which will be close to zero when k is large.

Thus, in the LSI-SVD method, C is also a covariance matrix. Instead of using the original term-document matrix, it is computed by the left singular matrix (*term matrix*) with dimension reduction which is a simple transformation of the original term-document matrix and preserves the relationships between terms.

It is interesting to consider the properties of the matrix C. Clearly it is symmetric and about half of the eigenvalues are positive. The components of C will in general have negative values.

To interpret this, consider the terms *computer, systems* and *political*. Because *computer* and *systems* will often appear in the same documents, the correlation coefficient between them is likely to be a positive value. The same can be said for the correlation coefficient between *political* and *systems*. On the other hand, because *computer* and *political* seldom appear in the same documents, the correlation coefficient between them is likely to be a negative value. Thus, if we calculate the similarity between the phrases *computer systems* and *political systems* , this negative value will reduce the score compared with (4) So, the possibility of an article talking about political systems to be judged relevant to an article talking about computer systems is less when using (4). In the general case, it is just what we need.

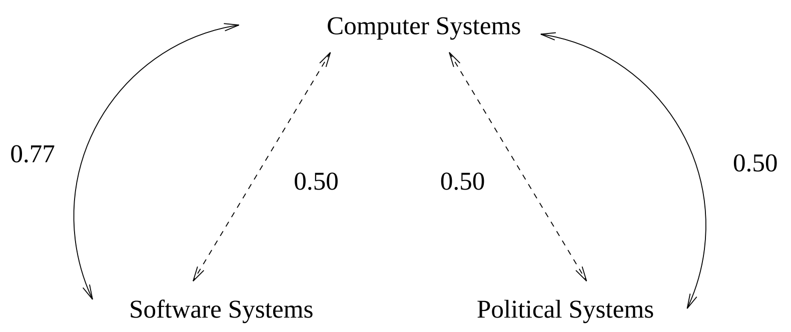


Figure 7: Similarity scores of two different methods. The numbers near the dashed lines are the scores obtained by a simple cosine measure while those near the solid curves are computed with the above correlation matrix.

As an example, suppose that the entries of the above three terms and another term *software* are in a correlation matrix as

	political	*computer*	*systems*	*software*
political	1	-0.1	0.1	-0.1
computer	-0.1	1	0.1	0.5
systems	0.1	0.1	1	0.1
software	-0.1	0.5	0.1	1

The similarity scores between the phrases composed of these words are illustrated in the following figure.

The numbers near the dashed lines are the scores obtained by a simple cosine measure using (1) while those near the solid curves are computed with the above correlation matrix and (4). Clearly, the latter one makes more sense if we use a higher threshold.

Use theory as proposed by Wittgenstein and others says that the meaning of a word is its use in the language. This suggests the use of rows/columns of C to represent the terms which will include information about the interrelations between the terms. With these representations, one can possibly find the *synonymy* and *polysemy* among terms.

3.5.3 Extensions

Formulae similar to (5) are used in pattern recognition to compare feature vectors of unknown patterns to the feature vectors in the pattern database. The matrix corresponding to C is the *inverse* of the covariance matrix of the features. In those cases, the feature vectors are observed in a noisy environment. To reduce the effect of noise signals, the inverse of the covariance matrix is used. It assigns low weight to the features with high covariance which contain more noise signals. Interestingly, it is opposite to what we do in term-weighting of text retrieval system. Generally, it is true that we can not consider the features in isolation but must evaluate them in combination. As for the whole current collection of the information agent, the useful features of the collection may be the documents themselves it contains. Again we can not consider the documents in the searching domain and current collection in isolation but must evaluate them in combination.

3.5.4 Building structure

With an efficient correlation computation, we can get a good metric of similarities between the documents. Usually, it can be easily modified to a distance metric. For example, the distance metric corresponding to the similarity metric in (5) should be:

$$s = (x - y)^T C(x - y)$$

In some cases such as vector space model of text processing, x and y need to be normalized first to have an identical vector norm.

Using this information, a hierarchical structure of the document collection can be built automatically. The basic idea is to hierarchically cluster the data points in the feature vector space which represents the documents. Various Clustering Analysis methods and Vector Quantization techniques can be used here. The structure of the resulting hierarchical clusters of the documents is a tree. If the clusters are concentrated enough, the search for relevant (similar) documents can be sped up by tree traversal. Moreover, we can use this structure information to improve the performance of information agents. For example, restricting the query results within some clusters may prevent the agent from finding irrelevant documents due to coarse similarity measures. The information revealed by the structure may be very useful to fulfill the main purpose of the information agent — find the necessary information to reduce the uncertainty.

This self-organizing hierarchical structure can be used for building domain hierarchies introduced in a separate section in this article. The domain specification is difficult because there is no way to generate meaningful summaries automatically with our coarse representations of documents. However, we can assign enough features to each domain for different purposes. For comparing and searching, we can use the centroid vector of the domain. For browsing, the users need to view the information that best describes the domain, so we can display the document whose feature vector is closest to the centroid vector of the domain.

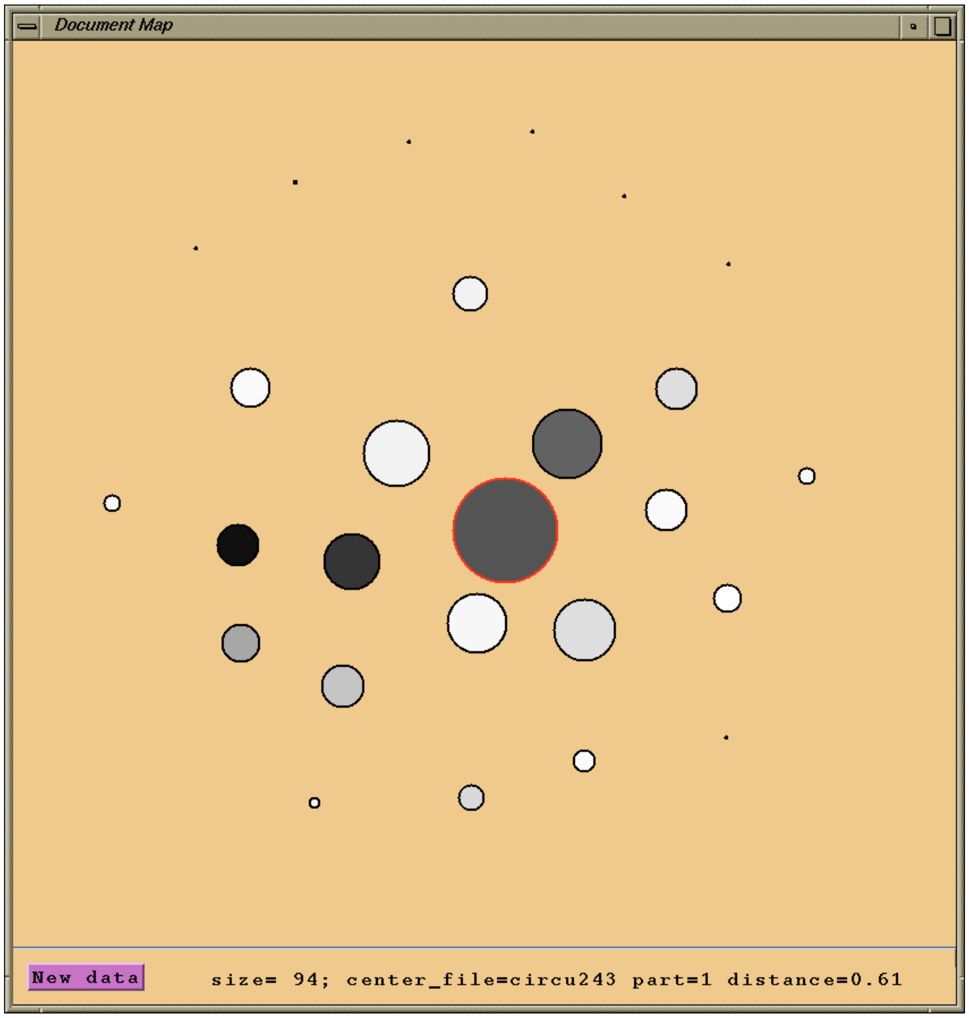
3.5.5 Visualization of Structure

To comprehend the document collection, the user should be able to visualize the relationships between documents. A good way to do this is to map the hierarchical clustering structure in high dimensional feature space to a planar, two dimensional space. For each cluster, we can compute a 2D map where the objects are the child clusters of that cluster. The size and density of clusters can be displayed by using different sizes and colors for the objects in the map. The density can be computed by the mean distance for the vectors to the center divided by the size of the cluster, for example. The distances between the objects in the 2D map should approximate the distances between the clusters in high dimensional space. See Figure 8 for an example of such a mapping that we have implemented. If users want the content information of the cluster, the document closet the centroid of the cluster can be viewed. When the user finds interest in one, he/she can browse the clusters close in the 2D map to others. Thus, it can overcome the drawback of most current IR systems where only a ranking list of the documents is provided. In such systems, the topics of the documents change randomly and it is not easy to return to the interesting documents that have been read.

The nonlinear mapping method [Jr.69] and self-organizing mapping method [Koh90] can be used for the mapping computation.

3.6 Redundancy

Correlation methods find relevant documents. The other important issue that information agents must consider is to prevent redundancy in the collection. Finding a metric for redundancy is a hard problem. Using the vector space model for text processing, there is a convenient interpretation of redundancy. Notice that the concatenation of two documents corresponds to the addition of the two document vectors which will not generate new information. One can conclude that linear dependence in the document vectors represent information redundancy in the collection. Thus the problem of preventing redundancy can be solve by finding an independent subset of the relevant document vectors.



```
size= 94; center_file=circu243 part=1 distance=0.61
```

Figure 8: Visualization of the cluster structure of a heterogeneous text document collection. Each sphere represents a cluster where sizes of spheres are proportional to sizes of clusters and the intensities of the colors of spheres are proportional to the densities of clusters. The distances between spheres approximate the distances between clusters in high dimensional feature space.

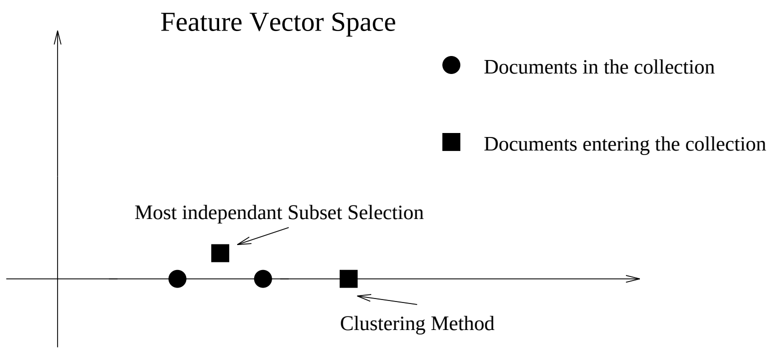

Feature Vector Space
● Documents in the collection
■ Documents entering the collection
Most independant Subset Selection
Clustering Method

Figure 9: Selection of documents avoiding redundancy: documents captured can differ when using different method

In practice, when the number of relevant documents the agent finds is large (this will happen more and more often with the growth of information resources), in order to reduce user effort and the computation and storage complexity of the application, we need to limit the number of documents presented. The same strategy can be used to find the subset of the relevant documents best covering the query or domain topic. That is to find the *most* independent subset of the relevant document vectors. It is consistent with the conclusion of Section 4 – to maximally reduce uncertainty we should find the documents that cover the coordinates in the feature vector space maximally.

This is a numerical linear algebra problem that can be solved efficiently by SVD and QR factorization with column pivoting [GL89]. This method is consistent with the conclusion of Section 4 – to maximally reduce uncertainty we should find the documents that cover the coordinates in the feature vector space maximally.

Another natural idea for finding the best representing vector subset is to find the vectors that are uniformly located in the vector space. This can be done by cluster analysis. After clustering, choose the vector for each cluster that is closet to the centroid of the cluster to represent the whole cluster. This method has great advantage when there is cluster structure in the collection and we want a new document to enter the collection. All that needs to be done is to choose the documents whose feature vectors will form new clusters. The problem is that it will not prevent linear dependency of the vectors. See Fig. 9. A hybrid method may help in this case: that is, using the centroid vector to represent the whole cluster, choose the vectors most independent of all the centroid vectors in the collection.

3.7 Authority

When a foreign information agent visits an information resource, should all its contents be opened up to the agent? Clearly not if the resource contains some secret issues or some contents that are not suitable for the user of the agent, for example a movie of violence for children. For a commercial resource, what information is available to the agent should depend on how much the user of the agent will pay for the service. Both cases invoke the idea of the authority of the information agent. That is agents of different users should be authorized with different access rights to the information. Authority issues are closely connected with the privacy issues that are discussed in a separate section.

3.7.1 Static cases

Generally, in the formal case introduced above, the rights of the agent are unchanged during the whole visit to the resource. Thus it is a static *Access Control* problem. It has been well studied in the literature of operating systems. Efficient mechanisms here might be the following.

- Group the user into different sets, such as *adults*, *children* or *people with privileges for the resource* and *people without privileges for the resource*.
- Assign each document or group of documents an attribute used to decide which group of users have access to them.

The identification of which group the users belong to is the critical issue. The accuracy of it will decide the efficiency of the mechanism. It can be done either by the information resource itself or by the agent. A safe approach is that the resource requests the registration of the user. For this approach, the resource will need a complete list of the authorized users and their privileges. (Different resources may share a single list.) Privacy issues have to be considered in this method because the user's information is available to the resource.

Another approach is to let the agent identify the privilege information of the user at the user's local network. The agent needs registration information of the user when started or users must select the agent with their privileges. We must make sure that the user is not able to modify the privilege information identified by the agent, i.e., the agent can not lie for the user.

3.7.2 Dynamic cases

In this case, whether or not the user will buy a piece of information in the resource is a decision that needs to be made after the connection being built. For a fully automatic agent without user's guidance, the agent is responsible for getting the most useful information at least expenses. Should the resource let the agent access the information before the agent decides to pay for it?

For a passive agent that only has the function of transmitting the useful information to the user without any predictive power, the answer is reasonable *yes*. After reading the information and considering the price of it, if the agent decides not to transmit this information to the user, it does not need to pay for it because the agent does not use the information in a productive way.

For an active information agent with predictive power or with the ability to increase its usefulness after reading the information, charges should be made. However, it will be difficult to decide whether to buy the information before reading it. Actually, we can let the agent read any information provided that the agent obeys the following rule:

> If the agent judged that the information read is useless, it must forget the information thoroughly and can not use it for any purpose. Otherwise the agent must pay for it.

The information architecture can be designed to make it possible! For example, a simple way to assure this is to let the agent run under the control of the resource. A part of the agent is sent to the resource. The resource runs it to do searching and any other operation except sending any information back. After that a list of selected documents are reported to the resource and the resource terminates the code. Then the resource sends the selected documents to the agent or applications . See Figure 10. The general reading authority of such an agent will become a strong motive of the user to use the agent. As we know, there is no way for humans to obey the above rule. Thus a human reader should pay for the information whenever he reads it. For the same amount of money a power agent can obtain much more useful information than human can do. Of course, the agent must be good enough to make the right judgment.

If the agent cannot make judgments without consulting the user, things become much more complicated. Because the users need to pay for the information before they have access to it, there must be some forms of advertisements associated with that information in order to persuade the user to buy it. These could be *abstracts*, some measure of popularities or particular side information such as *total numbers of current buyers*. In some circumstances, the resource and the user may make the agreement about which information in the resource the agent or user has access to before the actual visit of the agent. For example, the resource may give the user a chance of a free visit to some parts of it for advertising purposes. A method of authorization here is to use an analogy with tickets to some events[PW89]. The

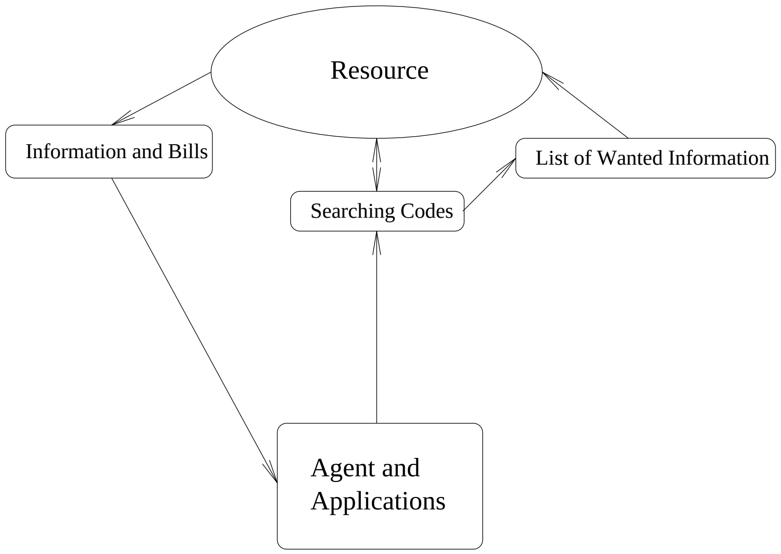


Figure 10: The agent can read any information in the resource but can only get the information that it is willing to pay for.

resource gives the agent a ticket that has the list of the information that can be visited in advance, and checks the ticket of the agent when its visiting to authorize the access rights.

Payments can be done using techniques from electronic banking.

3.8 Privacy

There are two distinct security issues. Authority which is discussed in the previous section is the idea that there should be no unauthorized access to an information resource. A more subtle security issue is privacy. The internal structure of most agents will contain implicit and explicit information about the people or organizations on whose behalf they collect information. However even if the internal structure of an agent is invisible, a malicious party can gain significant knowledge about the agent's owner simply by examining the agent's output. If a well-designed agent considers a document relevant, the owner of that agent probably considers the document relevant as well. The malicious party can examine the documents that the agent deems relevant in order to build a partial picture of the owner's current situation and tastes. The results can range from mere annoyance to blackmail. Annoyance is when a company adds a person to a mailing list. Blackmail is when someone discovers that a senator likes x-rated movies and decides to use this fact to influence an upcoming vote.

A method is needed to maintain the privacy of the agent's owner. If the agent and information resources are contained entirely on the owner's home machine, privacy is simply a matter of denying unauthorized access to the owner's home machine. This problem has been solved in most modern operating systems. Unfortunately most situations will involve distributed information resources and many situations will require the agent to run all or a portion of its code on remote machines. There is no way to prevent the owner of the remote machine from examining the agent's code. There is no way to prevent the owner of the information resource from examining the documents that the agent deems relevant and transmits back to its owner. Even if it were technically feasible, the owner of a remote machine would never allow unexamined code to run on that machine. The results would be catastrophic if the agent were malicious. The owner of the information resource would demand to know which documents the agent transmitted so that it could bill for services rendered. Therefore any solution must be more administrative than technical. The true identity of the agent's owner must be hidden while still

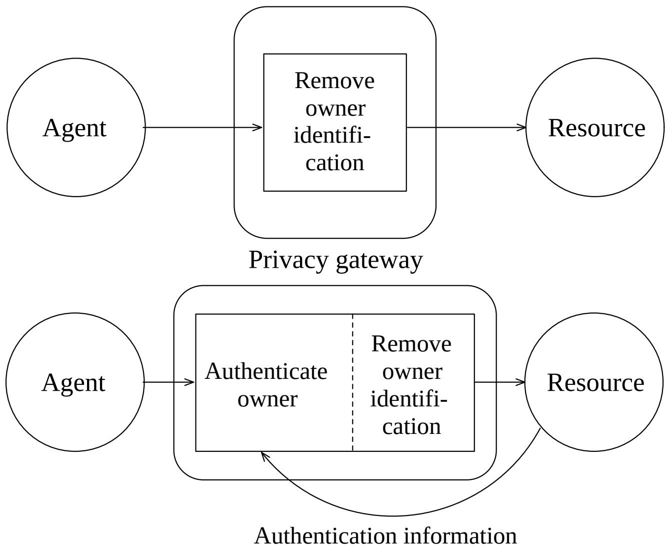

Agent
Remove owner identification
Resource
Privacy gateway
Agent
Authenticate owner
Remove owner identification
Resource
Authentication information

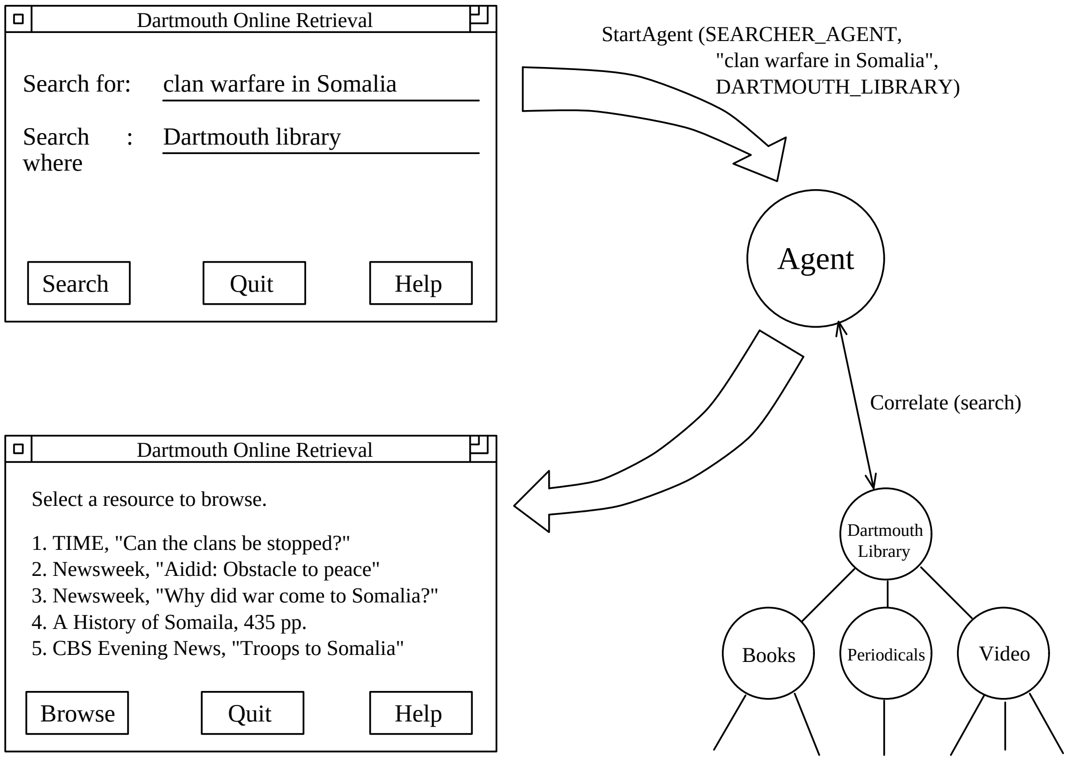
Figure 11: The two simplest privacy schemes. In the top scheme no authentication is required. In the bottom scheme authentication is required but both the agent and resource provider trust the third party.

providing a way for the information providers to perform authentication and bill for services rendered. A solution might be as simple as Nicholas Negroponte's "Swiss bank accounts" in which a trusted third party runs the agent on the owner's behalf so that no other user knows who owns the agent [Neg94]. Bills are sent to the trusted third party which then forwards the bills to the appropriate owners.

Authentication is more difficult since unfortunately privacy and authority are contrary goals. The privacy mechanism discussed above relies on a third party removing all owner identification from the agent. The third party then runs the agent on the owner's behalf. Authority relies on the resource provider knowing exactly who owns the agent so that proper authentication can be performed. Both security and privacy can be provided easily if both the agent's owner and the resource provider trust the third party. The third party can perform the necessary authentication checks on behalf of the resource provider. Figure 11 illustrates this scheme. The situation becomes much more difficult if the resource provider does not trust the third party and demands to perform all authentication checks itself. Providing privacy in such situations is the most important research issue in this area. Many ideas from operating system and network theory will be useful.

It should be noted that many situations do not require both privacy and security. For example privacy is not much of an issue in military environments but authentication is critical. Conversely authority is not much of an issue in many commercial environments but privacy is important. For example an online magazine service does not care who owns the agent as long as it can bill the owner for the articles provided. However the agent's owner might care deeply about privacy since many people view choice of reading material as highly personal. The third party can easily provide the necessary billing services and privacy.

[Nor94] discusses privacy briefly and underscores that the challenges are as much ethical as technical. Most technical work has dealt with authority rather than privacy. Some work uses the term "privacy" in the sense that the privacy of a user is maintained by denying unauthorized access to the entities owned by that user. We emphasize that this is not how the term is used here. *Authority* is the problem of denying unauthorized access to some entity. *Privacy* is the problem of preventing a malicious user from determining who owns a completely visible and accessible entity while allowing valid users to perform appropriate authentication checks and send necessary information to the owner.


Dartmouth Online Retrieval
Search
Quit
Help
StartAgent (SEARCHER_AGENT, "clan warfare in Somalia", DARTMOUTH_LIBRARY)
Agent
Correlate (search)
Dartmouth Library
Books
Periodicals
Video
Dartmouth Online Retrieval
Browse
Quit
Help

Search for: clan warfare in Somalia

Search where : Dartmouth library

Select a resource to browse.

1. TIME, "Can the clans be stopped?"
2. Newsweek, "Aidid: Obstacle to peace"
3. Newsweek, "Why did war come to Somalia?"
4. A History of Somaila, 435 pp.
5. CBS Evening News, "Troops to Somalia"

Figure 12: A simple interface for information retrieval

3.9 Interface

The interface provides a boundary between the user and the agents that operate on the user's behalf. The interface starts up agents in response to user requests and presents the results of agent execution to the user. Many interfaces will be shells whose sole purpose is to simplify agent startup. However other interfaces will be large applications that use agents to perform specific subtasks. Most interfaces will be intuitive, user-friendly graphical user interfaces (GUI). It is generally recognized that a well-defined graphical interface is far easier to use than an equivalent text-based interface.

An obvious example of an interface is an information retrieval application. The interface asks the user for a query and then starts up an agent that will find all documents relevant to the query. The agent searches some collection of information resources returns the relevant documents to the interface. The interface then presents the documents to the user. Figure 12 illustrates an information retrieval interface. The interface asks for various search parameters and passes these parameters to the agent that actually performs the search. The "search where" parameter is the domain specification. Agent technology is not limited to "traditional" information retrieval however so we can expect many other kinds of interfaces. For example it is easy to envision a calendar application in which the interface consists of a graphical representation of a paper calendar. The user schedules his meetings on this electronic calendar. A few days before each meeting the interface starts up an agent that will collect information relevant to the topic of the meeting and the people who will be attending the meeting. This information might include biographies, descriptions of a company's products and recent press articles. The morning of the meeting the interface presents the information to the user.

Another example of an interface is a medical database application. The interface would be a standard database engine – hopefully more intuitive and user-friendly than the database engines currently on the market – which would allow the user to display, edit and search patient records. However a wide range of agents could run "behind the scenes" and provide a wealth of additional services to health care workers. One agent could collect all information relevant to a patient and present this information to the patient's doctor the day before the patient comes in for an appointment. A second agent could automatically generate hyperlinks from patient records to relevant information. Hyperlinks could go to descriptions of drugs that have been prescribed for the patient, recent articles on the patient's condition and the records of patients with similar diseases. A third agent could cluster the patient records into groups. Each group would consist of patients with similar symptoms. Such clustering would help health care workers quickly determine that there has been an outbreak of some disease. There are many other agent possibilities.

The design of an appropriate interface will draw on all the work that has been done in human-computer interaction. However agents add a new facet to the traditional problems. Not only will the quality of the interface determine whether the application is accepted and used, the quality of the interface will determine whether the *underlying agent technology* is accepted and used. Effective presentation of agent activity will leave the user feeling that the agent has relieved him of a burdensome task and allowed him to make more efficient use of his time. Ineffective presentation of agent activity will leave the user feeling that he has lost control. Effective presentation of agent activity is particularly critical when the agent takes some automatic action on behalf of the user such as deleting and forwarding mail messages. The user must understand the action that was taken and must feel that he would have taken the same action if he had performed the task manually. Other factors besides an effective interface contribute to psychological acceptance of agents. Pattie Maes has noted that trainable agents are readily accepted [Mae94]. Users are comfortable with agent decisions since the agents learn which decisions to make by observing user behavior. The July 1994 issue of *Communications of the ACM* contains several articles that discuss the psychological acceptance of agents. These articles include [Mae94], [Bat94] and [Nor94].

4 The Shannon Machine - A Formal Model for Agents

The goal of this section is to describe a formal model, called a Shannon Machine, for an agent's activity using concepts from computation and information theory. Such a formal model is useful for characterizing the problems that information agents are meant to solve which up until now have been only vaguely defined. Specifically, while agents are computational procedures that manipulate information, the problems they ostensibly are solving have not typically been defined in terms of computational or information theoretic terms.

We begin by describing an information gathering problem as one that involves reducing uncertainty about some entity. Uncertainty is measured in terms of the conditional entropy of the entity given the information collected. This measure of uncertainty can be justified in economic terms by showing that returns on investments (or bets) are optimized if this measure of uncertainty is reduced. Much of the following material is based on known information theoretic constructs and our contribution is mainly in the application and interpretation of it to the information retrieval and intelligent agent context. All the requisite background is contained in standard references [CT91, LP81].

A *Shannon Machine* is based on the notion of a Turing Machine. It consists of a sextuple, $(K, K', \Sigma, \delta, \rho, s)$ where

- K is a finite set of states not containing the halt state, h;
- K' is a finite set of partial states with $K \subseteq K'$;
- Σ is an alphabet with a blank symbol, b, but not not the symbols L or R; $s \in K$ is the initial state;
- δ is a function from $K \times \Sigma$ to $(K' \cup \{h\}) \times (\Sigma \cup \{L, R\})$;
- ρ is a function from K' to K.

The operation of (K, Σ, δ, s) is identical to a standard Turing Machine except that a state transition is from a state within K to a state within K'. The operation of ρ is to map K', the set of partially defined states, to K, the set of completely defined states.

The action of ρ is to reduce uncertainty about states and this requires a model for states, their partial specification and the notion of uncertainty. The starting point for formalizing partial states and uncertainty is to define a *feature space*, $\mathcal{F}_i$, which is taken to be finite. Here i is again a finite index, $1 \leq i \leq n$. The product space, $\mathcal{F} = \Pi_i \mathcal{F}_i$, is the domain of possible tuples. Each element of $\mathcal{F}$, say d, has a probability of being true, $p(d)$, but p itself is not a probability distribution on $\mathcal{F}$. In other words, $\mathcal{F}$ does not necessarily consist of mutually exclusive events with respect to p.

We use $\mathcal{F}$ extended by wildcard values $*$ to represent the partially specified state space K'. The mapping ρ augments a partial state by retrieving elements from K' that serve to complete the state probabilistically. When the uncertainty in a state is sufficiently reduced, a state is accepted and the Shannon Machine takes a Turing Machine-like step.

The extended space $\mathcal{F}^* = \Pi_i(\mathcal{F}_i \cup \{*\})$ is the Cartesian product space of the original $\mathcal{F}_i$ with a distinct element added, namely *, which represents a wildcard element, or unspecified coordinate.

We start with some element $q \in \mathcal{F}^*$ which is called a *query*. A query is a partially specified element of $\mathcal{F}$ in the sense that some number of coordinates are specified, the rest being wildcarded. Without loss of generality, we assume that the specified coordinates are f_j with $1 \leq j \leq k$ because for the purposes of this presentation, the query is fixed.

An element of $\mathcal{F}^*$ is called *relevant* to q if the k specified coordinates of q match the first k coordinates of q, with the remaining coordinates arbitrary.

A query induces a conditional probability distribution on $\mathcal{F}$ according to

$$\mu_q(d) = \frac{p(d)}{(\sum'_d p(d'))}$$

where the summation is over relevant $d' = (f_1, ..., f_k, ...) \in \mathcal{F}$ and $d = (f_1, ..., f_k, ...)$. If d is not relevant to q then $\mu_q(d) = 0$. This induced probability distribution has an interpretation as a Bayesian conditional probability if we assume a uniform prior on all of $\mathcal{F}$. In general, the denominator $\sum'_d p(d')$ will be nonzero and we treat the case where it is zero shortly.

The entropy of a query is the entropy of μ_q:

$$H(q) = H(\mu_q) = -\sum_d^{\prime} \mu_q(d') \log \mu_q(d').$$

This entropy captures the uncertainty of the query within $\mathcal{F}$. H is large when the distribution μ_q is close to uniform and is small when the probability is concentrated on a few d'.

A *document*, y, is an element of $\mathcal{F}^*$ and is represented as $y = (\eta_j)$ where each $\eta_j \in \mathcal{F}_j$ or $\eta_j = *$, the latter case meaning that η_j is unspecified in F_j. Again, the document y is *relevant* to the query q if $y = (\eta_j)$ satisfies $\eta_j = q_j$ for $1 \leq j \leq k$.

The conditional entropy of q given a relevant document y is the entropy of the modified probability distribution

$$\mu_q(d|y) = \mu_q(d)\chi(d, y)/p$$

where $\chi(d, y) = 1$ if d and y agree on all unwildcarded coordinates , $\chi(d, y) = 0$ otherwise and $p = \sum_d \mu_q(d)\chi(d, y)$ is the normalization. If y is relevant with no wildcarded coordinates, this entropy is 0. Such a document specifies all coordinates.

Specifically, the entropy is

$$H(q) = -\sum_d \mu_q(d) \log(\mu_q(d))$$

where logarithms are taken base 2.

If we have a number of documents, $y_1, y_2, ..., y_k$ then the conditional entropy is given by the entropy of the conditional distribution

$$\mu_q(d|y_1, y_2, ...y_k) = \mu_q(d)\chi(d, y_1, y_2, ..., y_k)/p$$

where $\chi(d, y_1, y_2, ..., y_k) = \Pi_j \chi(d, y_j)$ and p is the normalization again. Formally, we have

$$H(q|y_1, y_2, ..., y_k) = -\sum_d \mu_q(d|y_1, y_2, ...y_k) \log(\mu_q(d|y_1, y_2, ...y_k)).$$

Relevant information reduces uncertainty according to the following result.

Theorem 1 *Let y be relevant to q. Then for relevant $y_1, ..., y_{k+1}$, we have*

$$H(q) \geq H(q|y) \geq H(q|y_1, ...y_k) \geq H(q|y_1, ..., y_k, y_{k+1}) \geq 0.$$

Proof This follows from properties of conditional probabilities and conditional entropies.

The order of presentation of relevant documents does not play a role in entropy, only the total ensemble of documents. That is,

$$H(q|y_1, ..., y_k) = H(q|y_{i_1}, ..., y_{i_k})$$

which can be checked by appealing to the definitions above.

The reduction in uncertainty as measured by entropy can be justified in economic terms by applying results from gambling and portfolio optimization according to the following sequence of ideas. Let $I(q, y_1, ..., y_k) = H(q) - H(q|y_1, ..., y_k)$ be the mutual information in the query q and the relevant documents y_j. By the above, $0 \leq I(q, y_1, ..., y_k) \leq H(q)$ so that the uncertainty in q can be reduced by at most $H(q)$.

Let $b(d)$ be the investment (bet) made in d with a payoff of $r(d)$ if d happens. The investments are normalized so that $\sum_d b(d) = 1$. The expected doubling rate on an investment strategy is then

$$W(q) = \sum_d \mu_q(d) \log(b(d)r(d))$$

which is optimized for $b(d) = \mu_q(d)$ in which case $W^*(q) = \sum_d \mu_q(d) \log(\mu_q(d)r(d))$. The doubling rate given the information in y is accordingly

$$W(q|y) = \sum_d \mu_q(d|y) \log(b(d)r(d))$$

with optimal value $W^*(q|y) = \sum_d \mu_q(d|y) \log(\mu_q(d|y)r(d))$.

The increase in the doubling rate given the information y is then

$$W^*(q|y) - W^*(q) = I(q, y) = H(q) - H(q|y)$$

which justifies using entropies as measures of uncertainty.

We now relate these results to concrete computational operations in information retrieval. The first step is finding relevant documents, d, which involves matching the specified coordinates of the query, q. This step involves correlation operations whereby searches are made for documents d that maximally match the query q. The measure of similarity between the document and the query is given by

$$S(q, d) = \sum_{j=1}^{k} \chi(q_j, y_j)$$

which can be expressed as an inner product according to the expanded representations of q and d as vectors. Introducing the index set $X = \cup_j \mathcal{F}_j \cup \{*_j\}$

$$\overline{q} = (\overline{q}_x)$$

where $\overline{q}_x = 1$ if $x \in X$ is one of the coordinate values of q and $\overline{q}_x = 0$ otherwise. The same extension of a document d is made with the convention that a wildcarded coordinate in $\mathcal{F}_j$ is handled by setting all the corresponding coordinates of $\overline{d}$ equal to 0.

In this vector representation, $S(q,d) = \overline{q} \cdot \overline{d}$ where $\cdot$ denotes vector inner product. Finding relevant documents is thus equivalent to finding documents whose inner products, or correlations, with q are maximal.

We now turn to the problem of identifying documents, d, which reduce uncertainty in the query in some maximal way. To this end, we introduce a partial ordering on documents via

$$y = (\eta_j) < z = (\zeta_j)$$

providing $\eta_j = \zeta_j$ whenever $\zeta_j \neq *$. That is, y has more coordinates specified. In this case, we have the following results both of which follow from the properties of conditional entropy.

Theorem 2 *If $y < z$ then*

$$H(q|y) \leq H(q|z).$$

Theorem 3 *If $y < z$ then for any $y_1, y_2, ..., y_n$ we have*

$$H(q|y_1, y_2, ..., y_n, y) \leq H(q|y_1, y_2, ..., y_n, z).$$

5 Summary

Rapid progress in telecommunications and computing made networking a must. This networking imperative has brought about an *informational watershed*, when readers searching for data are focused on minutia of the search process instead of specifying their domimnant needs based on the structure of the information required, and the actions which are to be taken according to its content. Agent technology is called to break through the watershed, mediating information search, management and usage steps to encapsulate logistics of search and provide readers with a higher abstraction level of information resources. Agent technology presented here describes an agent program architecture and structure of its resources. Interaction of the agent program, on one hand, with the user, and, on the other, with a data frame of the resources hierarchy manifests specific issues to resolve. On the agent-user side of the watershed it is agent specification, relegation authority to an agent while protecting privacy of the user, and predictive interface to incorporate agents' activities into broader context of the user's business. Resource handling for search and update is described as building and traversing knowledge domain hierarchy. Intelligent agent correlates data segments with each other on its way of traversing the hierarchy, taking decisions about the further search and possibly real-time interaction with the user, such as warning or advice. Agent accesses information selectively to get the most from the data hierarchy, optimizing the time, commercial information fees and other resources over the search. To enable this selectivity, a measure of dissimilarity of a whole document collection, called *redundancy* is introduced. Agent computes it along with correlating pairs of document, trying to decrease *uncertainty* in the information space being managed, with respect to the user-specified task. A model of generic agent machine, called Shannon machine, provides information-theoretical basis to fight uncertainty.

6 Acknowledgements

The authors wish to acknowledge Vishal Gupta for starting us down this path; Victor Mcgee and Alok Kumar for insight into the world of finance; Keith Kotay and David Kotz for useful discussions on remote execution; and our families and friends who provide constant support and encouragement.

A Glossary

Agent - An information agent is software that manages an information space according to specifications defined by a user. The information space can be distributed and can vary over time. An agent can search document collections and make decisions based on the information in a document collection.

Authority - This refers to the automatic actions that an agent can make on behalf of the user.

Correlation agent - An agent that searches a domain in order to find all documents that are related to an input set of feature vectors.

Document - Any set of data.

Document collection - Any set of logically related documents. Examples of document collections include a collection of outdoor scenes, all articles in the most recent Reader's Digest and all files with a certain owner.

Domain - A set of document collections and possibly other domains.

Feature vector - A general way of representing a document. Each element of the vector describes some feature of the document. A feature vector can either be lossy or lossless. A lossless feature vector contains enough information to recreate the original document. A lossy feature vector does not provide enough information to recreate the original document but instead can be viewed as a summary of the document.

Predictive agent - An agent that makes a decision based on an input set of feature vectors and its own internal rules. The output of the agent can be viewed as a set of feature vectors that describe the decision.

Privacy - An agent and its output reveals much about the agent's owner. Other people and organizations should be unable to collect this information or unable to associate this information with the agent's owner.

References

[And73] Michael R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York and London, 1973.

[Bat94] Joseph Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7):68–71, July 1994.

[BJR94] Paul Barker, Thomas Johannsen, and Colin Robbins. A survey of current and possible future uses of x.500 directory services. *Journal of Information Networking*, 1(3), March 1994.

[BN84] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, February 1984.

[Boe94] Personal communication, August 1994.

[Bus45] Vannevar Bush. As we may think. *Atlantic Monthly*, pages pp101–108, July 1945.

[C+92] A.M. Cummings et al. *University Libraries and Scholarly Communications.* Association of Research Libraries, New York, 1992.

[Cic64] M. T. Cicero. *De senectute. De amicitia. De divinatione.* Classical Library. Loeb, London; Cambridge, MA, 1964.

[CT91] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* Wiley, New York, 1991.

[D$^+$90] S. Deerwester et al. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[E$^+$94] Ernest A. Edmonds et al. Support for collaborative design: Agents and emergence. *Communications of the ACM*, 37(7):72–76, July 1994.

[EW94] Oren Etzioni and Daniel Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7):72–76, July 1994.

[FW91] T. Finn and G. Wiederhold. An overview of KQML: A knowledge query and manipulation language. Available through the Stanford University Computer Science Department, 1991.

[G$^+$92] M. R. Genesereth et al. *Knowledge Interchange Format Version 3 Reference Manual, Logic-92-1.* Stanford University Logic Group, 1992.

[G$^+$94] Y. Gong et al. An image database system with content capturing and fast image indexing abilities. In *Proceedings of the International Conference on Multimedia Computing and Systems*, Washington, May 1994. IEEE, IEEE Computer Society Press.

[GK94a] Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53,147, July 1994.

[GK94b] Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, July 1994.

[GL89] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore and London, 1989.

[Gra95] Robert S. Gray. Transportable agents. Ph.D. Thesis Proposal, 1995. Available from the author.

[Gre94] Irene Greif. Desktop agents in group-enabled projects. *Communications of the ACM*, 37(7):72–76, July 1994.

[GSS94] Michael Genesereth, Narinder Singh, and Mustafa Syed. A distributed and anonymous knowledge sharing approach to software interoperation. In Yannis Labrou and Tim Finin, editors, *Proceedings of the CIKM Workshop on Intelligent Information Agents, Third International Conference on Information and Knowledge Management (CIKM 94)*, Gaithersburg, Maryland, December 1994.

[Gup94] Vishal Gupta. Dartmouth associative retrieval kernel (DARK). Master's thesis, Dartmouth College, 1994.

[Hay88] J. P. Hayes. Building hypertext using automatic text processing. In *Proceedings of the AAAI-88 Workshop on AI and Hypertext*, St. Paul, Minnesota, 1988.

[Jr.69] John W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, May 1969.

[JvRS95] Dag Johansen, Robbert van Renesse, and Fred B. Scheidner. Operating system support for mobile agents. In *Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems*, 1995.

[Kem62] et al. Kemeny, J. G. A library for 2000 a.d. In Martin Greenberger, editor, *Management and the Computer of the Future.* The MIT Press and Wiley, 1962.

[Khr94] Alexy V. Khrabrov. Modern information society calls for intelligent agents. Available via the author (alexy.khrabrov@dartmouth.edu), 1994.

[KK94] Keith Kotay and David Kotz. Transportable agents. In Yannis Labrou and Tim Finin, editors, *Proceedings of the CIKM Workshop on Intelligent Information Agents, Third International Conference on Information and Knowledge Management (CIKM 94)*, Gaithersburg, Maryland, December 1994.

[Koh90] Teuvo Kohonen. The self organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[Kun93] S. Y. Kung. *Digital Neural Networks*. PTR Prentice Hall, Englewood Cliffs, New Jersey 07632, 1993.

[Lew95] Ted G. Lewis. Where is client/server software heading? *IEEE Computer*, pages 49–55, April 1995.

[lot93] Lotus development corporation: Notes user's guide, 1993.

[LP81] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[M$^+$94] Tom Mitchell et al. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):72–76, July 1994.

[Mae94] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40,142, July 1994.

[Neg94] Nicholas Negroponte. Bits about bits. Keynote address at the IEEE International Conference on Multimedia Computing and Systems., May 1994.

[Nor94] Donald Norman. How might people interact with agents. *Communications of the ACM*, 37(7):68–71, July 1994.

[PW89] James R. Pinkert and Larry L. Wear. *Operating Systems*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1989.

[R$^+$93] D.E. Rose et al. Content awareness in a file system interface: Implementing the 'pile' metaphor for organizing information. In *Proceedings of ACM-SIGIR '93*, pages 260–268, 1993.

[Rea91] T. C. Rearick. Automating the conersion of text into hypertext. In E. Berk and J. Devlin, editors, *Hypertext/Hypermedia Handbook*. McGraw-Hill, 1991.

[Rei94] Andy Reinhardt. The network with smarts. *Byte*, pages 51–64, October 1994.

[Rie94] Doug Riecken. Introduction. *Communications of the ACM*, 37(7):72–76, July 1994.

[Rin91] R. Riner. Automated conversion. In E. Berk and J. Devlin, editors, *Hypertext/Hypermedia Handbook*. McGraw-Hill, 1991.

[RWG94] A. Duda R. Weiss and D. K. Gifford. Content-based access to algebraic video. In *Proceedings of the International Conference on Multimedia Computing and Systems*, Washington, May 1994. IEEE, IEEE Computer Society Press.

[Sal73] G. Salton. Recent studies in automatic text analysis and document retrieval. *Journal of the ACM*, 20(2):258–278, 1973.

[SCS94] David Canfield Smith, Allen Cypher, and Jim Spohrer. Kidsim: Programming agents without a programming language. *Communications of the ACM*, 37(7):72–76, July 1994.

[Sel94] Ted Selker. Coach: A teaching agent that learns. *Communications of the ACM*, 37(7):72–76, July 1994.

[SG90] J. W. Stamos and D. K. Gifford. Remote evaluation. *ACM Transactions on Progamming Languages and Systems*, 12(4):537–565, 1990.

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Computer Science Series. McGraw-Hill, New York, 1983.

[Spa89] Eugene H. Spafford. The internet worm: Crisis and aftermath. *Communications of the ACM*, 32(6):678–687, 1989.

[SS94] Mukesh Singhal and Niranjan G. Shivaratri. *Advanced concepts in operating systems: Distributed, database and multiprocessor operating systems.* McGraw-Hill Series in Computer Science. McGraw-Hill, New York, 1994.

[Sto94] A. D. Stoyenko. SUPRA-RPC: SUbprogram PaRAmeters in Remote Procedure Calls. *Software-Practice and Experience*, 24(1):27–49, January 1994.

[Val84] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:11:1134–1142, 1984.

[Whi94] James E. White. Telescript technology: The foundation for the electronic marketplace. General magic white paper, General Magic, Inc., 1994.