A Learning Approach to Personalized Information Filtering

by

Beerud Dilip Sheth

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

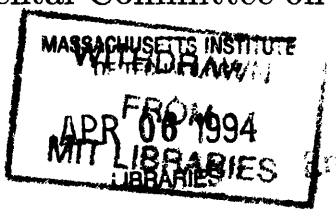
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1994

Author .....
Department of Electrical Engineering and Computer Science
Jan 14, 1994

Certified by .....
Pattie Maes
Assistant Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students



A Learning Approach to Personalized Information Filtering

by

Beerud Dilip Sheth

Submitted to the Department of Electrical Engineering and Computer Science
on Jan 14, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

A personalized information filtering system must specialize to current interests of the user and adapt as they change over time. It must also explore newer domains for potentially interesting information.

A learning approach to building personalized information filtering systems is proposed. The system is designed as a collection of information filtering interface agents. Interface Agents are intelligent and autonomous computer programs which learn users' preferences and act on their behalf — electronic personal assistants that automate tasks for the user. This thesis presents the basic framework for personalized information filtering agents, and describes an implementation, "Newt", built using the framework. Newt uses a keyword based filtering algorithm. The learning mechanisms used are relevance feedback and the genetic algorithm. The user interface is friendly and accessible to both naive as well as power users.

Experimental results indicate that Newt can be personalized to serve some of the news filtering needs of the user, in particular, those that are more regular and predictable. Relevance feedback is good for specializing to user interests. The genetic algorithm causes the system to adapt and explore for new types of information. This demonstrates that Interface Agents are a promising approach to the problem of designing personalized information filtering.

Thesis Supervisor: Pattie Maes
Title: Assistant Professor of Media Arts and Sciences

Acknowledgements

I would like to take this opportunity to thank all those who have contributed to this thesis, directly or indirectly.

I would first like to express my sincere gratitude to my adviser Prof. Pattie Maes for her constructive criticism, guidance, support and for sharing her immense wealth of knowledge, all of which have had a strong influence on this thesis. I would also like to thank her for providing me the appropriate environment for research, and for being accessible at all times.

I'd like to thank Yezdi Lashkari, Thuy (Cecile) Pham, Leonard Foner and Carl Feynman for providing comments on earlier drafts of this thesis. I would like to thank Anne Dudfield for help with some elements of the graphical interface and Cecile Pham for help with the icons. Thanks to Richard Marcus, Paul Resnick and Anil Chakravarthy for numerous discussions and Viet Anh for helping out with the news servers.

I also owe gratitude to the members of the Autonomous Agents Group (do we have a name yet? :-) for helping make work a real pleasure. These include Pattie Maes, Henry Lieberman, Bruce Blumberg, Abbe Don, Robyn Kozierok, Leonard Foner, Yezdi Lashkari, Max Metral, Christina Davidson, and my dependable bartok!

I'd like to thank News In the Future Consortium and the National Science Foundation (grant 9205668-IRI) for providing financial support for the work in this thesis.

I am deeply indebted to my parents for their encouragement, support and sacrifices without which I certainly would not have reached where I have. I'd especially like to thank my fiancee, Nipa, for her love, friendship, understanding and for being a constant source of inspiration. I look forward to joining you in May!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The problem

The number of networked users has increased rapidly with the widespread proliferation of computers and networks. If the Internet is any indication, the number of people who have started using online services has increased dramatically in recent years, . The number of Internet hosts, *i.e.* machines which have direct connectivity, is over a million [26] and is growing exponentially. There are many more which connect to the Internet indirectly through intermediary services such as America Online and Compuserve. This explosive growth has fed the growth in the amount of information resources available over the networks. As more information becomes available, it becomes increasingly difficult to search for information. However, as the number of users increases, newer users are likely to be less network savvy. Getting information should become easier, not harder, if newer users are to be able to meet their information needs. It is, therefore, critically important to build tools that help users serve their information needs better.

Information Filtering deals with the delivery of information that is relevant to the user in a timely manner. An information filtering system assists users by filtering the data stream and delivering the relevant information to the user. Information preferences vary greatly across users, therefore, filtering systems must be highly personalized to serve the individual interests of the user. A personalized filtering system

must satisfy three requirements:

- **Specialization:** A personalized filtering system must serve the specific interests of the user. The system selects the articles deemed to be interesting to the user and eliminates the rest. However, a filtering system might not be able to perfectly differentiate the articles that are actually relevant to the user from the ones that are not. The proportion of irrelevant articles delivered to the user should be as low as possible. The proportion of relevant articles eliminated should also be low. Since filtering involves repeated interactions with the user, the system should be able to identify patterns in the users behavior. The filtering system must infer the habits of the user and specialize to them, *i.e.* recommend as many relevant articles and as few irrelevant articles as possible.

- **Adaptation:** Since filtering typically involves interaction over long periods of time, user interests cannot be assumed to stay constant. When they change, the system must first be able to notice that the user interests have changed. Secondly, the system must adapt its behavior in response to the change. Anticipating and adapting to user needs helps make the system more user friendly. This is essential if more and more people are to use it.

- **Exploration:** A filtering system should also be capable of exploring newer information domains to find something of potential interest to the user. There are two motivations for exploration. One is that exploration helps match a presently unknown but real user interest. The other motivation is that it helps improve the adaptation process. This is the case because newer kinds of information need to be explored to serve the changing user interests.

On the one hand, users need to be able to control the huge amounts of information inflow and fulfill their information needs. On the other hand, users need to be relieved of repetitive computer tasks by providing higher level interface abstractions. A solution for both problems is to design an information filtering system that learns. The goal of an information filtering system is not to fully perform all information

filtering tasks, but to automate those that are more repetitive and predictable. Filtering systems can perform some of the filtering tasks and help the user manage the flux of information. Learning filtering systems can adapt to the tastes and preferences of the user and automate tasks for them. The proposed information filtering agents will be analyzed in detail in the following chapters.

1.2 Contributions

The work presented in this thesis uses techniques from the field of Interface Agents to solve the problem of information filtering. Interface Agents are computer programs that learn the tastes and preferences of users and automate repetitive and predictable computer related tasks for them [28]. An information filtering agent learns the information related preferences of the user and automates some of the filtering tasks for the user. The contributions made by this thesis are as follows:

- This thesis refines the statement of the personalized information filtering problem. Personalized filtering systems must be specialized to user interests, adaptive to preference changes and explore newer information domains. While the first two criteria have been addressed earlier [7, 39], exploration has not been sufficiently emphasized before.

- The vector space model for document representation has been generalized for use with documents containing more than just text. The distance metric for computation of document score and the effect of relevance feedback has also been generalized.

- Programming by demonstration [8] is proposed as an additional method of providing feedback to the filtering system. The user is no longer constrained to providing feedback only to the documents retrieved by the profiles, but can also pro-actively train the system using documents she found.

- This thesis validates the use of Genetic Algorithms [18] for modeling adaptive and exploratory behavior in Filtering systems.

- Experimental results show that using only relevance feedback is sufficient for specializing to user interests, but not satisfactory for modeling adaptive behavior.

1.3 Overview of this document

The rest of this thesis is organized as follows. The next chapter is a statement of the problem that is addressed in the rest of this thesis. It presents an overview of the previous work from which ideas have been drawn and introduces the proposed approach to solving the problem. Chapter 3 presents the basic algorithms used to build the information filtering agent. It describes in detail the underlying architecture using which adaptive agents could be built for information filtering. Chapter 4 describes a particular implementation, Newt (acronym for News Tailor), built using the algorithms described in Chapter 3. Efficiency issues are touched upon at the end of the chapter. The experiments performed with Newt and the results obtained are presented in Chapter 5. Chapter 6 contains the concluding remarks and Chapter 7 presents some directions for future work.

Chapter 2

Personalized Information Filtering

A tremendous amount of news and information is created and delivered over electronic media. This has made it increasingly difficult for individuals to control and effectively manage the potentially infinite flow of information. Ironically, just as more and more users are getting online, it is getting increasingly difficult to find information unless one knows exactly where to get it from and how to get it. Tools to regulate the flow are urgently needed to prevent computer users from being drowned by the flood of incoming information.

Information filtering systems can help users by eliminating the irrelevant information and by bringing the relevant information to the user's attention. Filters are mediators between the sources of information and their end-users. Belkin and Croft [7] provide a good description of Information filtering (IF) and identify the similarities and differences with Information Retrieval (IR). Filtering contexts typically involve a dynamic stream of information, as opposed to static data bases used in traditional IR systems. Due to the dynamic nature of the stream, timeliness of information assumes added significance in the filtering context. The amount of data involved in filtering environments is usually very large and unstructured.

IF involves repeated interactions over multiple sessions with users having long-term goals. This is in contrast to IR systems, where the user typically has a short term information need that is satisfied within a single session. This implies that an IF system must remember the user and individualize its performance for her. Filtering

systems maintain *profiles* which are representations of the user interests. Privacy of the user profiles is an issue that is of concern in filtering systems. Adequate precautions must be taken to ensure that complete control over the user profile rests with the user.

Filtering systems are much more likely to be used by a wider community of users than IR systems. The users of IF might not be highly motivated in their information seeking and may not have well defined interests. Therefore, IF systems need to be much more user-friendly than IR systems which have been mostly aimed at the highly motivated information seeker with very specific information needs.

A *Personalized* IF system should be highly responsive to the needs of the user. Since a filtering system involves repeated interactions with the user, the system should get a better feel for the user's needs over time. Assuming that a lot of the user actions are consistent, the system should get increasingly better at matching her needs over time. The system should be able to gradually converge to that part of user needs which is predictable and consistent. Furthermore, since the interaction could extend over a long period of time, the user's interests cannot be assumed to stay constant. The change in interest could be anything from a slight shift in relative priorities to completely losing interest in some domain and gaining interest in another. The system must be able to detect or must allow the user to indicate the change in interests and should respond by adapting to these changes. Finally, considering the possibility that the information seeker might not be highly motivated, the system should be capable of recommending new potentially interesting information based on what it already knows about her. The system must be able to explore newer domains and prospect for interesting information.

To summarize, the *personalized information filtering problem* is to design a system that can provide information that matches user needs in a consistent and timely manner. The system must also accommodate changes in user needs and adapt to those changes. Ideally, the system should be capable of entertaining not only the currently known needs of the user, but also exploring different domains to find articles of potential user interest. Thus, designing a system that is *specialized*, *adaptive* and

exploratory is the "holy grail" of personalized information filtering.

2.1 Previous Work

The work in this thesis is at the intersection of two historically distinct research areas, namely Information Retrieval/Filtering and Software Agents. Information Retrieval is a well established field of information science that addresses issues of retrieval from a large collection of documents in response to user queries. Information Retrieval literature has recently begun to address issues of Information Filtering [7, 11, 13]. By comparison, Agent research is a relatively new field of study which has grown out of Artificial Intelligence (AI). Agent research is concerned with issues of designing intelligent and autonomous software for a variety of tasks. This thesis draws upon work from both these areas.

2.1.1 Information Retrieval and Filtering

Information Retrieval

Three main retrieval paradigms can be identified [30] in the IR literature: (*i*) statistical (*ii*) semantic and (*iii*) contextual/structural. The first approach emphasizes statistical correlations of word counts in documents and document collections. Salton [38, 39] describes the use of statistical schemes such as probabilistic and vector space models for document representation and retrieval. The Smart system [42] is an example of a text processing and retrieval system based on the vector processing model. Latent Semantic Indexing (LSI) [5] is another example of a statistical method to capture the term associations in documents. The semantic approach to retrieval characterizes the documents and queries so as to represent the underlying meaning [17, 36]. It emphasizes natural language processing or the use of AI-like frames. The third approach, also known as "smart" Boolean, takes advantage of the structural and contextual information typically available in retrieval systems. For example, this could involve the use of thesauri in which relationships among terms are encoded [12]

or take advantage of context and structure generally available from the document terms [30]. CONIT [31] is an example of a system built in the "smart" Boolean framework.

The Internet is one of the largest publicly available "databases" of documents (among other things) and is a good testing ground for most retrieval techniques. With the Internet having seen an explosive growth in recent years, a number of services have arisen on the Internet to help users search and retrieve documents from servers around the world — WAIS, Gopher and World Wide Web to name a few. Wide Area Information Servers (WAIS) [21] is a networked based document indexing and retrieval system for textual data. The servers maintain inverted indexes of keywords that are used for efficient retrieval of documents. WAIS allows users to provide relevance feedback to further specialize an initial query. Gopher [32] is primarily a tool for browsing through hierarchically organized documents, but it also allows to search for information using full-text indexes. In the World Wide Web (WWW) [3], the information is organized using the hypertext paradigm where users can explore information by selecting hypertext links to other information. Documents also contain indexes which the user can search for.

A number of commercial retrieval systems are available in the market. Lexis/Nexis and WestLaw are well known information services which contain legal and other information. Both services retrieve documents in response to boolean queries from users. Hoover, marketed by Sandpoint, is an agent that acts as an intelligent librarian which knows how and where to look for information. Applesearch, released by Apple, is quite similar to Hoover.

Information Filtering

In contrast to IR, IF has only recently started to attract attention. Based on a survey of information sharing in organizations, three approaches can be identified [29]. Depending on the manner in which documents are selected for the user, filtering systems can be classified as cognitive, social or economic. *Cognitive* systems choose documents based on the characteristics of their contents. *Social* systems select documents based

on the recommendations and annotations of other users. *Economic* systems select documents based on some computation of cost-benefit to the user and through some pricing mechanisms.

A variety of approaches have been used to get at the semantic contents of the documents. *Oval* [29] is an example of a system which uses a keyword-based approach to match user defined rules to incoming documents. Foltz [10] demonstrates the use of LSI for information filtering and evaluates it for filtering Netnews articles. [11] performs a similar experiment for the domain of technical reports. INFOSCOPE [9] consists of rule-based agents which observe usage patterns and make suggestions to the user. The agents monitor the contents of the messages that are deemed interesting or uninteresting, make statistical correlations and suggest changes to the user.

Social systems typically select documents based on the ratings that other users assign to them. Users collaborate to help each other filter documents. Eager readers would be the first ones to read incoming articles and provide their endorsements, which will be used by passive readers to filter articles, as in GoodNews [45]. The selection could either depend on personal criteria (*e.g.* endorsement by a friend), or on aggregate criteria (*e.g.* endorsements by more than half the group members). Tapestry [13] is an example of a collaborative environment that accepts information from many sources, allows detailed endorsements and defines a query language to access endorsed articles.

The cognitive and social approaches are both just as valid for selecting documents. The difference lies in the fact that depending on the application area, one is likely to be more valuable than the other. If information is being gathered for keeping up to date with a certain community, social filtering is the way to choose documents. However, if information is being gathered based on the topic, independent of who the other users of the information are, then cognitive systems are more appropriate. Of course, sometimes a hybrid approach is necessary.

Commercial filtering services have recently entered the market. "First!" is a personalized news clipping service provided by Individual, Inc. The user profile is acquired by talking to the users and having them fill out templates. Personalized

news is delivered frequently through facsimile or email messages. The filtering is done by the Smart system [42], augmented by human supervision.

Learning and Adaptation in IR and IF

There is relatively little difference between IR and IF at an abstract level, since both are concerned with the task of getting information to people who need it. However there are certain aspects of the general problem that have been ignored by the IR literature but are especially relevant in filtering contexts. Learning and adaptation is an issue of significance in the filtering context that has not been emphasized as much in IR research.

IR has mostly dealt with the learning issue through relevance feedback. Relevance feedback refers to the reformulation of a search query in response to feedback provided by the user for the results of previous versions of the query. Work on Relevance Feedback methods in IR has a long history [38]. It has been used for the vector space model [37] and found to significantly improve performance [40]. Another approach to modeling adaptation is the use of genetic algorithms (GA) to devise adaptive algorithms for information retrieval. Yang and Korfhage [47] evolve a population of query individuals to optimize the search. Gordon [14], on the other hand, uses a method where competing representations are associated with documents. The representations are then altered over time using GA.

Learning and adaptation is, however, of much greater importance in filtering contexts. Filtering is concerned with repeated use of the system by users with long term interests. Filtering systems are often used by larger communities of people, a large number of whom might not be highly motivated information seekers [7]. Interests may not always be well defined or might not always be well expressed. In addition, the users' interests cannot be assumed to be constant, as mentioned before. Filtering systems must therefore be responsive to dynamic user interests.

Baclace [2] proposes a hybrid algorithm to evolve agents for information intake filtering. Information intake filtering (IIF) refers to "prioritizing objects in a conceptual in-basket". The IIF agents learn using a combination of a genetic and economic

algorithm.

A number of different techniques have been used for modeling the user. Doppelgänger [35] is a user modeling system that acquires information through many sensors. For example, the "badge sensor" transmits physical location of the user while a "login sensor" tracks when people log in to computers. It uses prediction techniques such as time series analysis and Hidden Markov models to make inferences about user behavior. Any client application can obtain these inferences by making a request to the user model.

2.1.2 Software Agents

An *agent* is a system that tries to fulfill its goals in a complex, dynamic environment. It is situated in the environment and interacts through sensors and actuators. Autonomous adaptive agents operate totally autonomously and become better over time at achieving its goals. Agents that are situated in the "cyberspace" environment are known as "software agents" or "interface agents" [27]. *Interface Agents* are computer programs that automate repetitive tasks to provide assistance to a user dealing with a particular computer application.

The idea of employing agents to delegate computer-based tasks goes back to research by Negroponte [34] and Kay [22]. The research in this field is directed toward the ideal of agents that have high-level human-like communication skills and can accept high-level goals and reliably translate these to low level tasks.

Two approaches have been traditionally used for designing interface agents. The first approach is to make the end-user program the agent. An example is the *Oval* system [25] which allows users to program rules that dictate the agent's behavior. The advantage of this approach is that it gives the user total control. This makes it easy for the user to trust the rules, since she herself created them. However, the disadvantage is that the burden of the programming task is on the user, which may not be desirable. The other approach is to knowledge-engineer the agent with substantial background knowledge about the application and the user. For example, UCEgo [4] helps the user to use the Unix operating system by making various suggestions to

correct user mistakes. The advantage is that the user no longer has to program the agent – the knowledge engineer will do that for her. However, the problem now is that the agent might not be customized to individual preferences. It might also be difficult for the user to trust the agent since she does not have a good idea of how it works and what its limitations are.

A third approach using machine learning techniques to overcome the shortcomings of the other two has been proposed by Maes and Kozierok [28]. The goal of this approach is to build Agents that acquire their competence and adapt to user requirements. In computer environments that involve a lot of repetitive tasks and where the individual differences outweigh the similarities, a machine learning approach has the potential to be very useful. The agent can learn by observing the user and imitating her, by reacting to feedback from the user and by learning from training examples provided by the user. A meeting scheduling agent using the above approach is described in [23, 24]. The meeting scheduling agent uses memory-based reasoning and reinforcement learning to automate actions for the user. It can also use rules provided by the user and learn from examples of meeting in hypothetical situations. An electronic mail agent is described in [28, 33].

A number of products are commercially sold as agents in the market. Magnet, from No Hands Software, can automate filing, scheduling and similar tasks for the user. "Open Sesame!", from Charles River Associates Inc., is a learning agent that monitors the keyboard and mouse watching for repeating patterns. BeyondMail, from Beyond Inc., can be taught to automate responses to incoming electronic mail.

2.2 Learning Agents for Information Filtering

2.2.1 The proposed approach

Drawing inspiration from the two above-mentioned fields of research, this thesis proposes the use of Interface Agents for Personalized Information Filtering. IF is an ideal and challenging environment for the use of interface agents. The information

space is complex and getting more so. It is also very dynamic and the new information coming in is highly unpredictable. To add to the complexity, the user needs could themselves be changing. Adaptive agents can help users cope with this flux of information. Furthermore, IF meets the two requirements for applicability of agents — repetitive computer interaction and differences in individual preferences.

The proposed idea is to build a set of adaptive autonomous interface agents that inhabit the user's computer and are assigned the goal of being responsive to the information needs of the user. The agents can sense user feedback as well as changes in the information environment. The agents are autonomous as they can take actions relating to news filtering on the user's behalf. The agents are adaptive as they learn the preferences of the user and adapt as they change over time. The learning mechanism used by the agents is relevance feedback and genetic algorithm. The profiles used by the filtering system consist of terms which are matched with the contents of the documents *i.e.* the agents use cognitive filtering. The algorithm used by the agent is described in the following chapter.

The learning mechanism used in the information filtering agents is motivated by research in Genetic Algorithms and Artificial Evolution [1, 6, 15, 18, 19]. IF is effectively a dynamically changing search problem. Searching a large and changing space involves a trade-off between two objectives: (*i*) *exploiting* the currently available solution and (*ii*) further *exploring* the search space for a possibly better solution. Hill Climbing is an example of a search technique that exploits the best known alternative. However, because of this very reason, it is likely to get stuck in local maxima. Random Search, on the other hand, is an extreme case of an exploration search technique: it is unsatisfactory as it does not make use of the best solution found so far. Genetic Algorithms manage the trade-off between exploration and exploitation in a near optimal way — they exploit the solution found so far, while Crossover and Mutation operations provide a way of exploring the search space for better solutions [18].

Several experiments have demonstrated that artificial evolution is helped by individual learning [1, 19]. This phenomenon is also known as the "Baldwin effect": if the organisms evolved are allowed to learn during their lifetime, then the evolution

towards a fitter species happens much faster. This is the case because every individual is able to explore a "patch" of the search space (find the maximum fitness in the local neighborhood of its genotype) rather than a single point (evaluate the fitness of its own genotype).

Drawing upon the above-mentioned ideas, the information filtering agent is modeled as a population of profiles. Each profile searches for documents that match itself and recommends them to the user. The user can provide feedback for the documents recommended. User feedback causes two effects. One, it changes the fitness of the profiles. If the user provides positive (negative) feedback for a document, the fitness of the profile which retrieved that document is increased (decreased). Second, the profile is modified in response to user feedback. Thus, each profile learns during its lifetime, taking advantage of the Baldwin effect. The population as a whole continually adapts to the dynamic needs of the user.

2.2.2 Comparison with previous approaches

A number of differences can be identified on comparing the proposed approach with previous approaches to the personalized information filtering problem. The differences are discussed below.

It is difficult to model exploration using rule-based systems which try to detect patterns in the user's behavior. For example, INFOSCOPE learns using rule based systems which remember interesting topics covered in the past. New recommendations of topics are made to the user based on recency, frequency and spacing of past topics [9]. The disadvantage of such an approach is that it is constrained to making recommendations of topics which lie within the realm of user's past interests. On the other hand, we explicitly model exploration. The information filtering agent searches new domains for information which could be of potential interest to the user. It is quite possible that user may not have seen the topic before.

Our approach is quite similar to the work described in Yang and Korfhage [47]. They evolve a population of query individuals, we evolve a population of profile individuals. However, they assume that user interests are fixed and strive towards

convergence. We assume dynamic user interests and our goal is to continually adapt. Another difference is that each profile individual in our system learns within it's lifetime taking advantage of the Baldwin effect described above. This is not the case in the other system.

Our approach differs in many respects from Information Intake Filtering (IIF) [2] mentioned earlier. The first difference is that IIF is a subset of the personalized information filtering problem as defined in this thesis. By assuming an in-basket, it scales down the problem considerably to that of prioritizing articles that have already undergone one level of filtering before reaching the in-basket. The feature space in IIF depends on the size of the intake and therefore the in-basket cannot be scaled up to the universe of documents. Another difference is that IIF implicitly assumes that user interests stay fixed once learned. Furthermore, it does not explore newer information domains, which is the function of the mutation operator in Newt. The use of an economic system with GA for assigning payoffs is interesting, however, and deserves further research.

Doppelgänger [35] is a user modeling system briefly described above. The most important distinction with our approach to user modeling and is that Doppelgänger is application independent. Calling it a user modeling *shell* would be more appropriate to distinguish it from user models that are domain dependent. Since Doppelgänger solves a more general problem, there cannot be a direct comparison with the domain dependent user model acquired in Newt. The following comparison is only in the context of the information filtering problem being addressed in this thesis. The advantage of an application independent user model is that it is able to identify patterns in user behavior that are not performed while reading news but is important nonetheless to the IF problem. An example of such an inference is that the user reads business news immediately after login and weather news just before logout. The disadvantage is that a lot of this information might be irrelevant, perhaps even counter-productive to filtering. By comparison, the representation of profiles in Newt has been strongly motivated by the domain. It has been found to be a sufficient user model for keyword based filtering. Another difference is that some of the inferencing techniques used in

Doppelgänger make it difficult to find reasons for the inference. The relatively simpler keyword based model can provide reasonable explanations for selecting or rejecting documents. There are further differences in the learning techniques used. Due to the nature of the data collected (locations, states, login data, *etc.*), Doppelgänger uses Markov models, time series techniques and clustering techniques. In our approach, we are interested in an efficient parallel search and have chosen genetic algorithms.

Chapter 3

The Algorithm

The Personalized News Filtering system is modeled as a set of Information Filtering Interface Agents. An information filtering agent assists the user with the task of finding interesting news articles in a particular domain. It has technical knowledge about the task involved, namely, information filtering. It is also aware of the interests and preferences of the user. Agents use this knowledge to automate filtering tasks for the user. Users typically have more than one filtering agent. For example, this may be the case when the user has non-overlapping news interests and would like different agents to satisfy each of these interests.

An *agent* is modeled as a *population* of *profile* individuals, each of which searches for articles in a small domain. While each of these profiles would typically satisfy a small part of user interests, the behavior of the population as a whole is more interesting. Together, all profiles of a population try to match the complete user interests and adapt to them. The profile contains information about where to search for articles and what kinds of articles to filter. Profiles search for articles that are similar to itself. Top-scoring articles are retrieved for presentation to the user. The articles recommended by each of the profiles are collected together and presented to the user. The user can provide positive or negative *feedback* for an article. User feedback has two effects. First, the profile which retrieved the article is modified based on the *relevance feedback* for the article. Secondly, the *fitness* associated with the profile increases or decreases for positive and negative feedback respectively. The

fitness of a profile represents its overall suitability in serving user interests.

The population of profiles is continually *evolving*, in response to dynamic changes in user preferences. When the population evolves from one generation to the next, the profiles with high fitness are retained in the *next generation*, while the unfit ones are eliminated. The vacancies created in the population are filled in by *genetic variants* of the fit ones. This introduces newer members into the population, which might potentially match user interests better than the unfit profiles they replaced. If so, they will attain high fitness values and will stay in the population. Else, they will be eliminated in succeeding generations.

This chapter presents the architectural framework of a filtering agent. It describes the representation of an agent as well as the learning algorithms used. The fundamental concepts discussed include the representations for documents and profiles, the filtering algorithm and the learning mechanisms, namely relevance feedback and the genetic algorithm. Each of these is described in detail in this chapter. Implementation issues have been dealt with in the following chapter.

3.1 Representation

The representation used for profiles and documents is based on the vector space representation, commonly used in the information retrieval literature [38]. In the vector space representation, documents and queries are both represented as vectors in some hyper-space. A distance metric which measures the proximity of vectors to each other is defined over the space. When a query is received, it is translated into its vector representation and document vectors in the proximity of the query vector are retrieved in response to the search. The advantage of using a common vector space for both documents and queries is that a document can also be used as a query itself *i.e.* one can find documents that are similar to a given document. Once the document-query is translated to a vector, the same distance metric can be used as for other queries. This property of vector spaces is quite useful for the current application, since users can provide samples of interesting articles as an alternative

to constructing intelligent queries.

A profile searches a part of the database looking for articles that are similar to it. Profiles are analogous to queries in information retrieval. The representations used for profiles and documents are described below.

3.1.1 Document

A standard method of indexing text consists of recognizing individual words, eliminating the commonly used words included on a word-exclusion list and using the remaining words for content identification of the texts. Sometimes, phrases are compacted and treated as a single term. Words may also be truncated to word-stems. Generally speaking, a "term" is used for text identification. Since the terms are not all equally important for content representation, importance factors (or weights) are assigned to the terms in proportion to their presumed importance for text content identification [41]. A text is then representable as a vector of terms

$$T_i =< w_{ij} >$$

where wij represents the weight of term t_j in text T_i.

In the information filtering context being considered in this thesis, documents contain more than just text. The documents can contain other information such as the author of the document, the source of information, the geographic origin of the news article, *etc.* It is therefore necessary to generalize the term-vector representation mentioned above. The generalized representation used is as follows.

A document consists of many *fields.* The text is just one of the many fields (and will be referred to as *keyword* field hereafter). The other fields could include *author*, *location* (the geographic origin of the news article), *date* (when the article was posted), (number of) *lines*, *etc.* There is no restriction on the number of fields, so long as it represents some attribute of the document. The *newsgroup* field needs special mentioning, since it is perhaps the only domain-specific field used in this representation. The news source dealt with in this thesis is USENET network news

available on the Internet. USENET is organized as a collection of newsgroups, each of which represents a broad category for the articles contained in it. Each document, therefore, has a newsgroup field indicating its category.

Each field is assigned terms to be used for identifying purposes. Since the terms are not all equally important, they are assigned weights. A field is thus represented as a term-vector similar to the one introduced above. In particular,

$$F_i^d =< w_{ij}^d >$$

where w_{ij} is the weight of term t_j in field F_i. The subscript i can be "a"(uthor), "k"(eyword), "l"(ocation) and so on. The superscript d indicates that F_i^d is a document field, (as opposed to a profile field described below).

Since a document consists of many fields, it is represented as a *set* of *field-vectors.* Formally,

$$D = \{F_i^d\}$$

where F_i^d is a field in document D.

3.1.2 Profile

The representation of a profile is similar to that of a document. A profile consists of a number of *fields*, like newsgroup, author, location, keyword, *etc.* Each field is a *vector of terms*, each of which is weighted in proportion to its importance for identification purposes.

However, there is a slight difference between profile and document representations. The profile stands for some user interest. Since the user might not have equal preferences for all the fields, importance factors are also assigned to the fields. Thus, a profile is a set of fields with field-weights,

$$P = \{(F_i^p, W(F_i^p))\}$$

where $W(F_i^p)$ gives the weight of the field F_i^p in profile P. The superscript p indicates

that F_i^p is a profile-field and not a document field. Each profile field F_i^p is represented identically to a document field

$$F_i^p =< w_{ij}^p >$$

3.2 Filtering Documents

The filtering process consists of translating documents to their vector space representations, finding documents which are similar to the profiles and selecting the top-scoring articles for presentation to the user.

3.2.1 Extracting Document Representations

Documents are represented as a set of fields where each field is a term-vector (see Section 3.1.2). The fields of the document representation must be extracted from the document itself. All document fields except the keyword field are directly extracted from the header lines of the article. The keyword field is generated from the text of the article.

For example, the location field can be created from the "Location" header line, if available. The terms available from the header line are added to the appropriate field and default weights are assigned. For *e.g.*, if the document has the following information:

Location: China, Taiwan

the vector representation in the Location field is:

$$F_l^d =< y, y >$$

where F_l^d is the location field in the document representation, y is the default weight and t_1 = "china", t_2 = "taiwan". Since the term vectors are normalized when documents are scored, it does not matter what the actual value of the default weight is. The weights just indicate the relative importance of the terms – in the above example both terms have the same relative importance.

The term-vector for the keyword field is obtained through a full text analysis of the documents. The weight of the term depends on its frequency of occurence in the text and the number of documents it appears in. This is a well known term weighting method for the vector-space model in the information retrieval literature [38] and has been adapted for the present use.

The weight of a keyword-term is the product of its term frequency and its inverse document frequency. The term frequency (*tf*) is the occurence frequency of the term in the text and is normally reflective of term importance. The inverse document frequency (*idf*) is a factor which enhances the terms which appear in fewer documents, while downgrading the terms occuring in many documents. The resulting effect is that the document-specific features get highlighted, while the collection-wide features are diminished in importance. The weight of the term is given as

$$w_{ik} = tf_{ik} \times idf_k$$

where tf_{ik} is the number of occurences of term t_k in document i, and idf_k is the inverse document frequency of the term t_k in the collection of documents. A commonly used measure for the inverse document frequency is

$$idf_k = \log(\frac{N}{n_k})$$

where N is the total number of documents in the "collection", of which n_k contain a given term t_k. The collection of documents is the context within which the inverse document frequencies are evaluated. A profile evaluates all documents in each newsgroup mentioned in the newsgroup field of the profile. It also searches some newsgroups which may not mentioned in the newsgroup field. These are the newsgroups which have some article belonging to it that received feedback through programming by demonstration. The implementation details are mentioned in Section 4.4.1.

3.2.2 Scoring Documents

In the classical vector space representation, documents matching queries are retrieved by finding vectors in the proximity of the query vector. A commonly used similarity metric is the cosine of the angle between vectors. This can be calculated by taking a scalar product of the two vectors

$$Similarity(V_i, V_j) = \sum_k w_{ik} \times w_{jk} \tag{3.1}$$

The distance metric in equation 3.1 is generalized for the current application. The similarity between a document and a profile is a function of the similarities between the corresponding fields. The field similarities are first computed. Since each field is a term-vector, the metric used for measuring similarity between two fields of the same type is just as in equation 3.1. The similarity scores for the corresponding fields in the document and the profile are computed as shown in equation 3.2. The similarity between a complete document and complete profile is computed next. It is the sum of the field similarity scores weighted by field weights of the profile. This computation is shown in equation 3.3.

$$S(F_i^d, F_i^p) = \sum_k w_{ik}^d \times w_{ik}^p \tag{3.2}$$

$$S(D, P) = \sum_i S(F_i^d, F_i^p) \times W(F_i^p) \tag{3.3}$$

Scores of different fields are added together in equation 3.3, weighted by the field weights. However, this would tend to favor fields which have arbitrarily high term-weights, even when the field weights are low. Field scores must only be compared and added unless they are all on a unform scale. This can be achieved by normalizing the field vectors. The scalar product of two normalized vectors lies in the closed interval [-1, 1]. This constrains the field scores to lie in the same interval.

$$\mid F_i^d \mid = \mid F_i^p \mid = 1 \quad \Rightarrow \quad -1 \leq S(F_i^d, F_i^p) \leq 1 \quad \forall i \tag{3.4}$$

The implicit effect of normalization is that it is not possible to compare term-weights across fields *i.e.* if the weight of keyword t_i is higher than the weight of author t_j, it does not imply that t_i would contribute more to the document score than t_j. The weight indicates the *relative* importance of a term with respect to the other terms in the same field. Users creating profiles should be made aware of this fact.

A similar problem lies with the document similarity scores. Profiles go through different parts of the database and score different articles. The similarity scores of documents are with respect to different profiles. When the agent collects all the top-scoring articles retrieved by the profiles, it is not possible to compare the scores unless they are all on the same scale. Besides, a user would not be able to make sense of the similarity score if the scale is not known. Hence, document scores are constrained to be in the closed interval [-1, 1].

The highest score of 1 would only be assigned when the document and profile representations are identical. *i.e.*

$$S(P, P) = 1$$

Further, when fields are identical the field score will also be 1, because of the definition of field similarities and the constraint in equation 3.4

$$S(F_i^p, F_i^p) = 1$$

The last two equations imply that

$$\sum_i W(F_i^p) = 1 \tag{3.5}$$

Hence, the constraints in equations 3.4 and 3.5 together ensure that the field scores and document scores lie on convenient and uniform scales.

3.2.3 Selecting Documents

Each profile searches a part of the database and scores all the available documents. The documents which are finally presented to the user are selected from among the documents which score well with respect to the different profiles. A parameter that can be used while selecting documents, in addition to the document score, is the fitness associated with the profiles. The fitness is a measure of how successful the profile has been in the past in meeting user requirements. There are two variables which can be manipulated while deciding upon the final documents. One is the number of documents and the other is the ordering of the documents. There are a number of approaches to selecting the final documents:

- The number documents contributed by each profile is proportional to its fitness. The advantage is that more documents would be contributed by profiles with higher fitness. A potentially undesirable effect is that a low scoring document from a fit profile would probably be selected over a high-scoring document from an unfit profile.

- The number of documents contributed by each profile is the same. However, the document score is scaled by a factor proportional to the fitness. This has an impact on the order in which documents are presented to the user. The advantage of this approach is that the user sees the documents contributed by fitter profiles before the others. The potential disadvantage is that since the number of documents contributed by each profile is the same, some documents with low scores from unfit profiles will still be chosen in preference to some high scoring documents from fit profiles.

- A third approach is to use a threshold. Any document which scores above the threshold will be selected independent of the profile which scored it. The advantage is that the quality of documents is implicitly assured by having a threshold. A possible disadvantage is that a document which matches perfectly with an unfit profile, will be selected in preference to a document with a slightly weaker match with a fit profile.

The basic problem is that the "goodness" of a document is some function of the fitness of the profile as well as the document score. While there are merits to each of the above approaches, there is no winning approach which would be universally applicable to all users and situations. The right choice could well be a combination of one or more of the approaches and would depend on the user preferences and the kind of documents being filtered.

The reason why a scoring threshold might be inappropriate is that a threshold does not have much meaning in itself. The document scores are good for relative comparisons, but do not have an intrinsic absolute value. Therefore, the recommended approach is a combination of the first two approaches so that both the number of documents and the ordering depend on the fitness.

3.3 Learning from Feedback

The user can communicate with the agent by providing feedback about interesting articles in two ways. The first way is to provide positive or negative feedback for the articles retrieved by the agent (or one of its profiles). Secondly, the user can provide examples of articles that the agent did not retrieve, which is an example of programming by demonstration. In each of these cases, user feedback has two effects. One is that the appropriate profile is modified in response to the feedback. The other is that the fitness of the profile responsible for the article is appropriately modified.

3.3.1 Feedback for retrieved documents

Relevance feedback has been used to improve the performance of retrieval systems [40]. For vector space representations, the method for query reformulation in response to user feedback is vector adjustment. Since queries and documents are both vectors, the query vector is moved closer to the vectors representing documents which received positive feedback and away from the vectors representing documents which received negative feedback.

The feedback mechanism used here is a generalization of the above method. Each

of the field-vectors in the profile is modified in response to user feedback. The process of modification for each of the field-vectors is similar to the classical vector adjustment method.

Consider a profile P, which contributed document D for presentation to the user. The user provides feedback f, which is a positive or negative integer indicating the amount of feedback. Each field-vector in the profile is changed in proportion to the feedback received,

$$P = P + \alpha \times f \times D \tag{3.6}$$

i.e. the weight of each term in each field is modified proportional to the learning rate and the feedback. α is the learning rate which indicates the sensitivity of the profile to user feedback.

Addition in the context of equation 3.6 means

$$\forall i, k: \qquad w_{ik}^{p} = w_{ik}^{p} + \alpha \times f \times w_{ik}^{d} \tag{3.7}$$

where w_{ik}^{p} is the weight of term t_k in field i of the profile and w_{ik}^{d} is the weight of the same term in field i of the document. The resulting effect is that, for those terms already present in the profile, the term-weights are modified in proportion to the feedback. Terms not already in the profile are added to the profile. The fitness of profile P is also modified in proportion to the feedback received

$$f(P) = f(P) + \beta \times f \tag{3.8}$$

where β is the sensitivity of the fitness to user feedback.

When feedback is provided for the document, the implicit assumption is that the feedback refers to all the features of the document, as can be seen from equation 3.7. If the user provides selective feedback for certain terms in the document, only the fields corresponding to the term are modified. For example, if the user likes a certain author of the document and provides selective feedback, only the author field is modified using equation 3.7. Similarly, the user can give selective feedback for a

portion of the text.

3.3.2 Programming by demonstration

In the case of programming by demonstration, the user provides feedback for an article that was not retrieved by any of the profiles. When feedback is received for such a document D, not retrieved by any of the profiles, the agent has two choices. It can either create a new profile to accept feedback about D, or it can direct the feedback to one of the existing profiles.

If there exists a profile P, which already retrieves articles from the same newsgroup as the one that D belongs to, then the feedback is directed towards P. The justification is that, since D belongs to a domain already searched by P, it is the profile which can benefit the most from receiving the feedback. This results in the modification of P, just as if P had retrieved the document itself. The only difference from equation 3.7 is that the learning rate α would be higher in the case of programming by demonstration, since the user took special efforts to personally seek out examples of articles.

However, if there is no such profile which meets the above requirement, an empty profile is created and is then given feedback as if it had retrieved document D. This effectively creates a profile which looks like D. The change in fitness is as shown in equation 3.8.

3.4 Genetic Algorithm

As mentioned earlier, an agent is modeled as a population of profiles. The structure of an individual profile is described in the preceding sections. The rest of this chapter describes the population characteristics and behavior in response to changing user interests.

The formal definition of a Population is given in equation 3.9 It is defined as a

set, where each element of the set is a pair of profile and its fitness.

$$G = \{(P, f(P))\} \tag{3.9}$$

The profile representation is the same as described in Section 3.1.2, which is repeated here for convenience.

$$P = \{(F_i^p, W(F_i^p))\}$$

where i = a(uthors), k(eywords), l(ocations), n(ewsgroups), p(riority), s(ource), *etc.*

The genetic operators, namely crossover and mutation, refresh the population every generation by introducing new members into the population and flushing out the unfit ones.

3.4.1 Crossover

For the purposes of this section, assume that the fields in every profile are identically ordered. The exact order is not important, as long as it is the same in all profiles. So, for example, in every profile, the newsgroup field is the first, the keyword field second, and so on.

Let P_1 and P_2 be the parent profiles:

$$P_1 = \{(f_i^p, W(f_i^p))\}$$

$$P_2 = \{(F_i^p, W(F_i^p))\}$$

The crossover operator takes two parents and produces two offsprings. Each offspring inherits some attributes from one parent and the rest from the other. A *two-point crossover* is used by the crossover operator. Two points are randomly selected in the list of fields. All the fields lying between the two points are exchanged between the two parents to create two new offsprings. Since the fields in both the parents are in the same order, each offspring will have one field of each kind. The fitness of the

offsprings is set to a default initial value. Formally:

$$P_1 \otimes P_2 \Rightarrow P_3, P_4 \tag{3.10}$$

$$P_3 = \{(g_i^p, W(g_i^p))\}$$

$$P_4 = \{(G_i^p, W(G_i^p))\} \tag{3.11}$$

Equation 3.10 defines the crossover operation, where parents P_1 and P_2 produce offsprings P_3 and P_4. The two crossover points are generated as shown in equation 3.12, where the random number function gives an integer in range specified by the two parameters inclusive (the index of the first field is 0). If the offspring profiles are as defined in equation 3.11, each field is derived as shown in equations 3.13 and 3.14. The fitness of the offspring profile is set to its default value, as in equation 3.15.

$$k_1 = random(1, max(Fields) - 2)$$

$$k_2 = random(k_1 + 1, max(Fields) - 1) \tag{3.12}$$

$$g_i^p = \begin{cases} f_i^p & 0 \leq i < k_1 \text{ and } k_2 < i \leq max(Fields) - 1 \\ F_i^p & k_1 \leq i \leq k_2 \end{cases} \tag{3.13}$$

$$G_i^p = \begin{cases} F_i^p & 0 \leq i < k_1 \text{ and } k_2 < i \leq max(Fields) - 1 \\ f_i^p & k_1 \leq i \leq k_2 \end{cases} \tag{3.14}$$

$$f(P_3) = 0.5$$

$$f(P_4) = 0.5 \tag{3.15}$$

3.4.2 Mutation

Let the parent P be:

$$P = \{(f_i^p, W(f_i^p))\}$$

The mutation operator is defined so as to contribute to the exploratory behavior of the population of profiles. An offspring resulting from mutation searches a newer domain not covered by the parent or other members of the population. This helps

the population of profiles to adapt to changing user interests. It also introduces a serendipitious element to the profiles.

The offspring is slightly different from the parent, so that the offspring explores a newer domain. At the same time, the offspring retains some attributes of the parent so that it can exploit the knowledge of the parent. This effect is achieved by modifying the newsgroup field of the profile, since this field controls the search domain of the profile. One of the newsgroups is randomly selected for replacement. This newsgroup is replaced by a randomly selected "similar" newsgroup.

A similarity function is defined over the set of all newsgroups, as explained in the following paragraph. This similarity function is used to compute *a priori* similarities between all pairs of newsgroups. When the original newsgroup needs to be replaced, its nearest neighbors are the candidates for replacement. A pre-decided threshold, say n, is used to choose the number of candidates to be considered. Since the mutated offspring must be different from the parent, none of the candidate newsgroups should be already present in the parent. Once the set of n nearest neighbors not already present in the parent is found, one of the candidate newsgroups is selected at random. This is used as the replacement in the offspring.

To calculate the similarity between two newsgroups, both are first represented as a vector of terms. The document frequencies of terms in a newsgroup are used as the term weights. The document frequency for a term t in newsgroup $\mathcal{M}$ is the fraction of the total number of documents in M which contain the term t. It is calculated as shown in equation 3.16. This gives a vector of terms weighted by document frequency. A similar vector of terms is calculated for the other newsgroup $\mathcal{N}$. The similarity between the two newsgroups is given by the cosine product of the two vectors, as shown in equation 3.17.

$$df_t^{\mathcal{M}} = \frac{n_t^{\mathcal{M}}}{N^{\mathcal{M}}} \tag{3.16}$$

$$S(\mathcal{M} \mid \mathcal{N}) = \sum_t df_t^{\mathcal{M}} \times df_t^{\mathcal{N}} \tag{3.17}$$

where $df_t^{\mathcal{M}}$ is the document frequency of term t in newsgroup M, $n_t^{\mathcal{M}}$ is the number of

documents in M which contain the term t and $N_{\mathcal{M}}$ is the total number of documents in M. $S(\mathcal{M} \mid \mathcal{N})$ gives the similarity between newsgroups $\mathcal{M}$ and $\mathcal{N}$.

As in the crossover operator, the fitness of the offspring is set to the default initial weight of 0.5.

3.4.3 New Generation

As before, the population is a set of (profile, fitness) tuples. For the present purposes, assume that the profiles in the population are ordered in the decreasing order of their fitness. So let,

$$\begin{aligned} G &= \{(p_i, f(p_i))\} \\ i < j &\Rightarrow f(p_i) > f(p_j) \end{aligned}$$

When a new generation of the population is created, the fit members of the population are retained and the unfit ones are eliminated. Let r be the *retention rate*, namely the proportion of population members that will be retained in the new generation. The vacancies are proportional to (1 - r). Let c be the crossover rate, namely the proportion of vacancies filled in by crossovers. The remaining vacancies are filled in by mutations. The new generation can then be described as follows:

$$G_{new} = \{(P_i, f(P_i))\}$$

where

$$\begin{aligned} P_i &= \begin{cases} p_i & 0 < i < rN_G \\ p_j \otimes p_k & rN_G \leq i \leq rN_G + (1-r)cN_G \\ p_{j'} & rN_G + (1-r)cN_G < i \leq N_G \end{cases} \\ f(P_i) &= 0.5 \end{aligned}$$

where N_G is the size of population G and j and k are random numbers between 0 and rN_G. The probability that j (or k) being some number x is proportional to $f(P_x)$ *i.e.*

the fitness of P_x.

Since the members added by mutation and crossover search for newer kinds of information, the *serendipity* in the system is proportional to the new members added every generation *i.e.* proportional to $(1-r)$. The *stability* of the system is proportional to r, which indicates the proportion of the population that is stable across generations. These values can be appropriately modified to better suit user preferences.

Chapter 4

Newt: An Implementation

4.1 Introduction

This chapter discusses Newt (acronym for News Tailor), a personalized news filtering system implemented based on the algorithms described in the preceding chapter. The architecture of Newt is described in detail followed by a discussion of some efficiency issues.

The system is essentially composed of three modules which encapsulate fairly distinct functions.

- The graphical user interface module is responsible for displaying the agents and the articles retrieved by them and allowing the user to provide feedback.
- The learning module is responsible for maintaining the mapping between the actual interests of the user and the set of user profiles.
- The information filtering module uses the user profiles as its input and finds articles which match the profiles.

Each of these is explained in further detail in the following sections.

The database from which Newt retrieves articles is USENET. This is a collection of bulletin-board like systems (called "newsgroups") on the Internet. While most of the newsgroups are for discussion among users, there are a large number of newswire-like newsgroups. The basic difference between the two is that, while anybody can

post an article to the discussion newsgroups, access to the latter is typically restricted to a few (often, only the administrators). Newt works much better with newswire-like newsgroups. This is for a variety of reasons. The first reason is that a keyword-based system depends on the fact that words are spelled correctly. Secondly, keyword-based inferencing is much more powerful if there is consistency in the usage of terms. The third reason is that Newt performs a computation-intensive analysis on the articles in the newsgroups being searched. Moderated newsgroups typically have fewer articles, thereby being more amenable to intensive analyses. Due to these reasons, the examples that have been presented in the following sections mostly involve moderated newsgroups carrying newswire articles.

4.2 The Graphical User Interface

The motivation for having Agents in the Interface is that they provide a higher level of abstraction for user-interaction. The most commonly pervasive metaphor for computer interaction is the *direct manipulation* metaphor [43] as manifested in contemporary desktop systems. The direct manipulation metaphor requires incremental actions on individual objects. This is highly inefficient when users want to manipulate sets of objects not individual objects, *i.e.* they want organization above the level of individual objects [46]. All actions have to be initiated by the user, which places an enormous burden of knowledge on the user. It is also difficult to extract explanations from the system or carry out asynchronous acts. In contrast to desktop systems, agent-based interfaces involve more peer-to-peer interaction where the user and agent both initiate communication. Delegation of tasks is at a much higher level, where the user delegates high-level goals and the agent determines how to execute these without much user consultation.

It is the responsibility of the Newt Graphical User Interface (GUI) to demonstrate the full power of the Agent abstraction. The interface consists of visual representations of one or more Information Filtering Agents. The system is meant to supplement manual browsing, and not necessarily supplant it. As a result, there are two modes of

interaction with the system: one involves direct interaction with the Filtering Agents and the other direct manipulation through manual browsing with the agent watching "over the user's shoulder". The Filtering Agents learn based on feedback provided by the user. The user can provide positive and negative feedback for the articles presented by the filtering agents. Another way to provide feedback is to manually browse through the datastream and demonstrate examples of interesting or uninteresting articles to the agents. The user can train the agents using the interaction described above. If that is not sufficient, the user has access to the internal state of the agent. If necessary, the user can manually edit any variable affecting the agent behavior. This allows the system to be accessible to all kinds of users ranging from naive- to the power-users, who would rather "program" and "debug" an agent than "teach" or "talk" with it.

4.2.1 The Main Window

The very first time a user uses this system, the user can either create an agent from scratch, or use off-the-shelf agents created by someone else. The way to do this is explained in Section 4.2.5. For the purposes of this section, assume that the user already has a set of agents.

At start-up, the Main Window consisting of a number of filtering agents appears on the screen (see figure 4-1). Each agent is visually represented by a graphical icon (the Agent Icon). Each agent is responsible for retrieving interesting news articles from some particular news domain. Below each of the Agent Icons is a set of 3 small icons, bearing the signs "+", "-" and "?" respectively. These are for providing positive/negative feedback and to ask the agent to explain its actions (see Section 4.2.3). Each agent is assigned a particular color, which is used to visually identify the graphical elements belonging to the agent. The background color of the three icons ("+", "-", "?") is the color code of the agent.

A menubar is provided to enable manual browsing and for manipulating agents (see following Sections on Manual Browsing and New Agents).

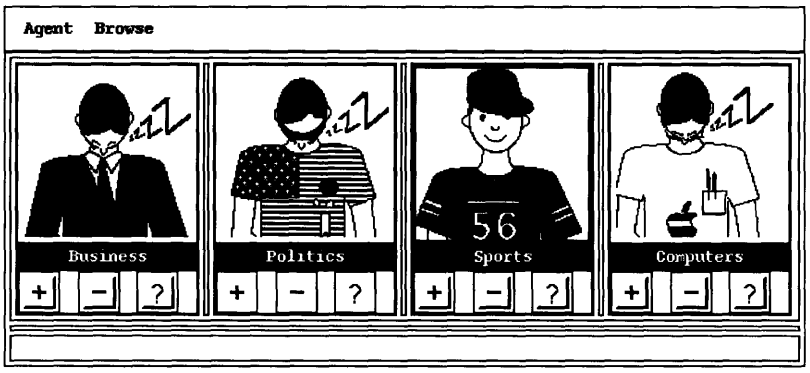


Figure 4-1: The main window of the graphical interface.

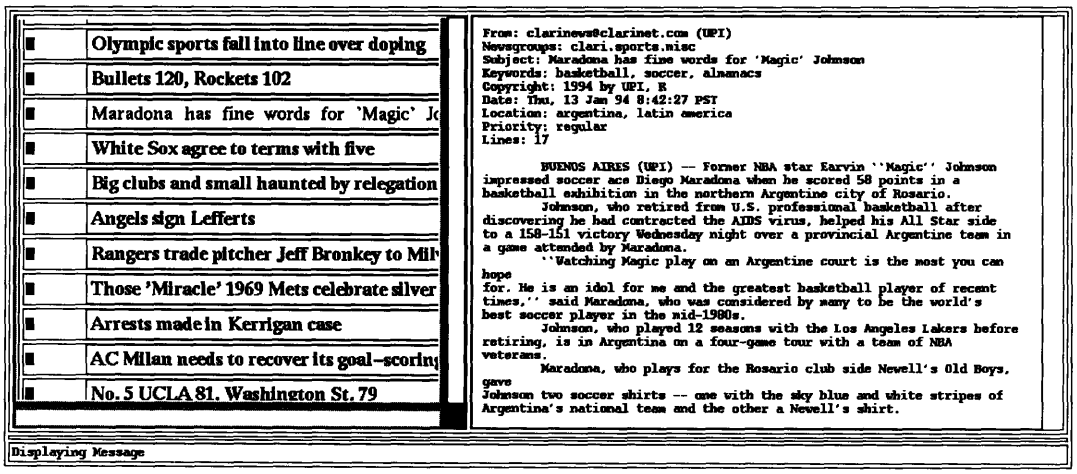


Figure 4-2: The news window of the graphical interface.

4.2.2 Reading News Retrieved By The Agent

Clicking on an Agent Icon brings up a window (News Window) for browsing through the news articles retrieved by that agent (see figure 4-2). In the left half of the News Window are the titles of the articles selected by the agent. The color code of the agent which retrieved the articles, is used as the background color for the list of titles. To the left of to each document title is a small bargraph indicating the score assigned to it by the agent. The titles are sorted in decreasing order of these scores.

To read the content of the article, the user clicks on the appropriate title. This brings up the text of the article in the right half of the news window. Scrollbars are provided to scroll through both the list of article titles, as well as the text of the article. Two kinds of fonts are used while displaying the list of article titles. Titles in "Bold" font indicate unread articles, while titles in "Normal" font indicate articles that have been read by the user. Clicking on the agent again, will make the corresponding News Window to disappear from the screen.

Some pre-created Agent Icons are provided along with the system. These display anthropmorphic cartoonish characters on the screen in various poses. The agent is "alert" when its News Window is currently open on the screen and is "asleep" otherwise. It makes a "thumbs-up" gesture when given positive feedback (figure 4-3) and "thumbs-down" when given negative feedback. In addition, the user is given the option of creating her own icons, if required. This is required when creating new agents and is explained in Section 4.2.5.

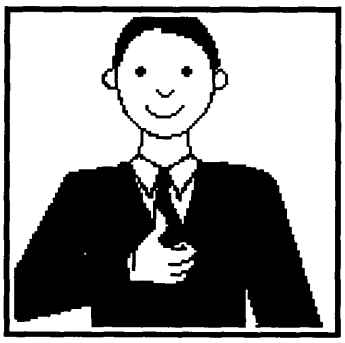


Figure 4-3: Agent response to positive feedback

4.2.3 Providing Feedback for Articles Retrieved

The user can provide feedback for the articles retrieved by the agent. Under each Agent Icon are small icons bearing the signs "+" and "-" to enable the agent to receive positive and negative feedback respectively. Clicking on one of these provides the appropriate feedback to the agent for the current article selection. "Current Selection" refers to the article that was last selected for reading in any of the News Windows. There is only one current article selection across all News Windows. This is so that any agent can be provided with feedback for any article (especially required in the case of manual browsing and programming by demonstration, see Section 4.2.4).

When feedback is provided as described above, the implicit assumption is that the feedback is for all the features of the article i.e. the keywords that were used to decide that the article should be retrieved, as well as the other fields in the article. All are used to modify the profile in consonance with the feedback. If the user wants to selectively provide feedback for a certain feature in the article (as opposed to the article as a whole), the user can highlight that portion of the text which contains the feature and then provide positive or negative feedback. For example, the user can highlight a paragraph in the text of the article and provide positive feedback. In this case, only the terms in the paragraph will be added to the profile instead of terms in the entire article.

The user can also seek explanations for selecting the current document. By clicking on the icon bearing the sign "?", the user can ask an agent why it recommended the current article selection. If the agent retrieved the document, it presents a brief explanation for recommending the document. In the current implementation, the only information provided as part of the explanation is the document score. Future versions are expected to have more detailed explanations.

4.2.4 Manual Browsing and Programming By Demonstration

As mentioned earlier, a direct interface to the datastream is provided so that the user can still manually browse through unfiltered data, bypassing the agents. The "Browse" menu in the menubar at the top of the Main Window brings up this interface. The menu allows the user to select a particular newsgroup. Some commonly used newsgroups merit a unique menu item, while the rest can be selected by choosing the "Other" menu item and typing in the name of the newsgroup.

All the articles belonging to the selected newsgroup are displayed in a new News Window. These articles are unfiltered and, therefore, unscored. In the case where one of the agents previously searched the newsgroup and retrieved some of the articles being displayed, the background of those article titles will be colored in the color code of the agent. This makes it easy to visually identify the agent(s) which searched the newsgroup and the articles it retrieved.

The user can browse through the articles in this window in the same manner as described in the previous section. Indeed, the user can also provide feedback just as before. For example, if there is an article that the user likes and she would like an agent to be able to recognize such articles in the future, the user can provide positive feedback for the article to the appropriate agent. This programming by demonstration will be particularly useful in the early stages of training and personalizing of new agents, as described below. Yet another motivation for programming by demonstration is that the user is not just constrained to providing feedback to what the agent already does. Instead, the user can also provide feedback to the agent about what it does not do, but should.

4.2.5 Adding New Agents and Training

The system allows for users to either create a new agent from scratch, or load an agent previously created and archived. An archived agent records the internal state of the agent at the time of archival, including the list of articles it had retrieved for

the user.

To load or create an agent, the user selects the "New Agent" menu-item in the "Agent" menubar of the Main Window. This pops up a dialog box which asks for the name of the agent and the directory in which to look for it. If an existing agent which meets this requirement is found, it is loaded, else a new agent is created. Before adding a new agent, an icon editor is presented to allow the user to create a personalized visual representation of the agent (the icon editor is a standard unix tool called bitmap). A new icon is added to the Main Window, a color code is assigned to the agent and the 3 icons for feedback and questions.

Once a new agent is added to the system, it is identical to the other agents in most respects. However, personalizing or training an agent assumes special significance in the case of new agents. This is because a new agent may not have any articles to present to the user for which it could receive feedback. The user will have to rely on other means to train the agent.

To compensate for the inability to provide direct feedback to new agents, the user can either demonstrate interesting articles to the agent, or manually edit the internal state of the agent. Programming by demonstration is particularly useful while training a new agent. By showing examples of interesting articles, the user is indirectly programming the agent to get "good" articles in the future. The other option to personalize new agents is to actually edit the internal state of the agent.

4.2.6 Population of Profiles

The interaction described above should suffice for most users most of the time. However, the user also has complete access to the internal state of the agent and can modify any state variables governing the behavior of the agent. On the one hand, this involves more work to make the necessary changes. On the other hand, direct access provides a better understanding of the agent's behavior and greater control over it.

An agent is modeled as a population of competing profiles which compete for the user's attention by presenting articles. The population of profiles can be viewed by

SHIFT-clicking (while the SHIFT key is pressed, click the mouse) on the appropriate Agent Icon. This brings up a window (Population Window) displaying the population of profiles that make up the agent (see figure 4-4). This window allows access to the population level features of the system. The Population Window displays the population of profiles. Each icon (Profile Icon) represents the profile and the scale adjoining it indicates its fitness. The fitness of a profile can be modified by manipulating this scale.

There are a number of operators provided in the lower section of the window. These operators allow the user to manipulate the members of the population.

- A blank profile can be introduced into the population by using the "Add" operator. A blank profile must be initialized immediately (see Section 4.2.7). This is because if no newsgroups are specified to be searched the profile would not be able to find any candidate articles. As a consequence, it would not receive feedback either.

- A member of the population can be eliminated by choosing the "Kill" operator.

- A "mutated" variant of a profile X can be added to the population by choosing operator "Mutate" and selecting profile X. Mutation modifies the set of newsgroups that a profile searches for articles. It does so by introducing a randomly selected newsgroup, similar to the newsgroup being replaced.

- Two profiles can be crossed over by exchanging some of the fields between them. This gives rise to two offspring profiles. This can be done by clicking the button labeled "Xover" and then selecting the two profiles to be crossed over.

- Instead of manipulating individual profiles, users can modify the whole population altogether. The next generation of the population is generated by retaining a certain number of the fit profiles, eliminating the rest and filling in the vacancies by genetic variants of the fit ones. This can be done by choosing the "doNextGen" operator.

4.2.7 Displaying Profiles

Each profile represents a subset of the total set of user interests. The profile can be viewed by SHIFT-clicking on the corresponding Profile Icon in the Population Window. This brings up the Profile Window (see figure 4-5).

The Profile Window can only display one field at a time. The set of values belonging to the currently displayed field is shown along with their weights. A few text editing operators (namely, "Add", "Delete" and "Edit") are provided to make editing changes to a profile. To change the current field being displayed, the user can click on the field name which pops a menu of field name choices.

It is also possible to view the news articles retrieved by an individual profile (as opposed to those retrieved by the population as a whole). Click on the button labeled "News" to display the news retrieved by this profile.

4.2.8 Archiving Agents

It is possible to archive agents, which can then be re-loaded or given to other users. Clicking on the button labeled "Archive" will archive the agent. The user needs to provide the Name of the agent and the directory to be archived in.

4.3 The Learning Module

The Learning module is responsible for ensuring that the set of profiles are personalized to the interests of the user. It handles the task of mapping user interests to the set of profiles and maintaining the correlation between the two. The set of profiles must fulfill two requirements. First, they must be specialized to the current information needs of the user. Second, they must dynamically adapt to user interests as they change. The Learning module thus encapsulates the Adaptive component of the system.

Each profile is stored in a different file. It is stored as an ASCII flatfile in a pre-determined format. Table 4.1 shows a sample profile. The top part of the table

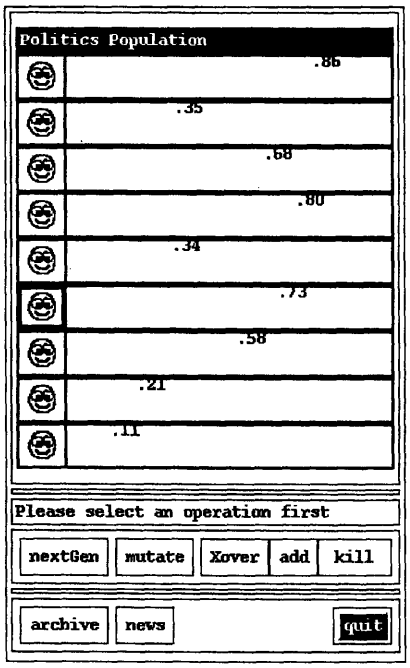


Figure 4-4: The population window displays the internal state of an agent.

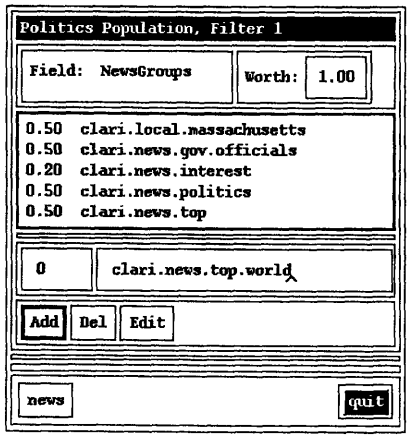


Figure 4-5: The profile window displays an interest profile.

is the profile representation described in the previous chapter. In addition to this, some implementation specific details are stored along with the profile in the same file for convenience. The other items in the file are used by the other modules and will discussed later.

4.3.1 User Feedback

User feedback has an effect at two levels. One is at the population level, where the fitness of members of the population increases or decreases depending on the feedback. This is required for genetic learning. The other effect is on the profile, which is modified based on the features of the article the feedback is given for. The user can also program the agent by demonstrating examples of documents and providing feedback for those.

As will be clear in the description of the Information Filtering Module, the document representation is generated while scoring the document and is not stored thereafter. Therefore, when feedback for a retrieved document is provided, its representation is no longer available for modifying the profile. This problem is much more acute in programming by demonstration, where feedback is provided for an article for which no representation has yet been created. The solution adopted in this implementation is that the pointers of documents are stored when feedback is provided, but the profiles are not immediately modified. The list of items labeled "ArtFeedback" in table 4.1 is the list of pointers to articles which received feedback in the last user session. Next to the pointers is a positive or negative number indicating the feedback for the article. The next time the Information Filtering module filters articles, it re-creates the document representation for all articles and the profile is then modified.

4.3.2 Genetic Algorithm

The population size needed for Newt is smaller than in typical GA applications for a number of reasons. First, each individual in the population is capable of learning

newsgroups:	0.2	
	clari.news.cast	1
	clari.news.gov.international	1
	clari.news.hot.east_europe	1
	clari.news.hot.ussr	1
	clari.news.top.world	1
locations:	0.1	
learningrate:	0.05	
numarts:	5	
keywords:	0.7	
	ukraine	0.475
	nuclear	0.294
	weapons	0.290
	soviet	0.239
	tactical	0.224
	somewhat	0.224
	inherited	0.215
	encouraged	0.207
	missiles	0.201
	kiev	0.201
	funds	0.164
	additional	0.158
	returned	0.149
	washington	0.148
	...	

————ArtScores————

clari.news.gov.international:<boutrosghali-norkorURecb_3DP@clarinet.com>	0.0902
clari.news.gov.international:<year-diplomacyUR4c4_3DN@clarinet.com>	0.0458
clari.news.gov.international:<boutrosghali-japanUM8f9-22b@clarinet.com>	0.0457
clari.news.gov.international:<boutrosghali-koreaUR819_3DO@clarinet.com>	0.0452
clari.news.hot.east_europe:<year-republicsURaa1_3DN@clarinet.com>	0.0450

————ArtFeedback————

clari.news.gov.international:<boutrosghali-japanUM8f9-22b@clarinet.com>	+1
clari.news.gov.international:<boutrosghali-koreaUR819_3DO@clarinet.com>	-1
clari.news.hot.east_europe:<year-republicsURaa1_3DN@clarinet.com>	+1

————ArtRead————

clari.news.gov.international:<boutrosghali-japanUM147-22c@clarinet.com>
clari.news.hot.east_europe:<russia-polandUR979_3DD@clarinet.com>
clari.news.gov.international:<france-natoUR349_3DE@clarinet.com>
clari.news.hot.east_europe:<ukraine-russiaUR55b_3DE@clarinet.com>
clari.news.hot.east_europe:<russia-usUR90c_3DG@clarinet.com>
clari.news.top.world:<poland-walesaUM902-227@clarinet.com>
clari.news.hot.east_europe:<russia-germanyUM94b-22a@clarinet.com>
...

Table 4.1: A sample interest profile

from feedback during its lifetime between two generations. The lack of diversity due to a smaller population will be compensated for by the learning mechanism. Considering the information filtering application domain, there is also much less change across generations than in typical applications of GA. A significant proportion of user interests is likely to stay stable, or change gradually at best. This application of GA is more like a slow evolution of population which continually adapts to environmental changes, as opposed to a directed search like, say, parameter optimization. Besides, the fundamental requirement of the population of specialized profiles is that they should at least cover the complete set of known user interests. In addition to this, they may have some extra profiles to explore the information space. Having a very large population will have the undesirable effect of retrieval of uninteresting documents. Finally, since the user decides the fitness of profiles by providing feedback, the greater the number of profiles, the greater the overhead of using such a system for the user. While it will vary across users, typical users would find few tens of profiles sufficient for most information needs. The actual size would depend on the capacity of the user to devour information, the amount of information available on the networks and the proportion thereof that is actually of (potential) interest to the user.

In the current implementation, all genetic operators are entirely user controlled. It is possible to either apply genetic operators to individual members of the population, or apply the "new generation" operator to the entire population. The effects of each of these operators is just as described in Section 3.4.

The mutation operator requires similarity measures between newsgroups. As described in Section 3.4.2, similarity is measured based on the correlation of frequently used terms in the two newsgroups. On typical USENET servers, the set of articles in a newsgroup changes on a continual basis. Strictly speaking, similarities should be measured at the time of mutation, so that the most up-to-date information is used. However, it is currently almost impossible to evaluate in real time similarity between two newsgroups, each of which could have up to thousands of articles. A more practical solution has been adopted by computing *a priori* similarities among

newsgroups. The similarities are re-computed only once a week. The implicit assumption is that while the articles in the newsgroups change quite frequently, the commonly used terms in the newsgroup change much more slowly.

Table 4.2 shows the results of one such computation. Similarity measures were computed for all pairs of newsgroups from a set of 27 newsgroups.

4.4 The Information Filtering Module

The Information Filtering module (called YAIF, for Yet Another Information Filter) is responsible for actually retrieving articles from the database of news articles. YAIF takes the profiles, scores articles with respect to the profiles and selects the high scoring articles to be presented to the user. YAIF is a time-intensive process and is run offline. The process is executed every night, so that filtered articles are available to every profile in the morning. This frequency is sufficient, since the data within a newsgroup does not change significantly within a day. In this section, the process of finding articles matching one profile is described. The same process is repeated for all available profiles.

Each profile is stored in a separate file. Two sets of newsgroups are searched for each profile. One set is the list of newsgroups specified in the newsgroup field of the profile. The other set consists of the newsgroups which may not be mentioned in the newsgroup field, but some articles belonging to it received feedback and are listed under "ArtFeedback" (see table 4.1). This is the case when users program by demonstration. For every profile, YAIF retrieves each article from the two sets of newsgroups and scores them with respect to the profile.

A typical article is shown in table 4.3. Each article contains some structured and unstructured information. The unstructured information is the actual information content of the article, namely the text. The structured part is the meta-information about the text of the article. This varies greatly depending on the source of news. There are a number of header lines that an article must have to adhere to the Standard for Interchange of USENET messages [20]. Some of the headers interesting for

clari.biz.invest	
8.99	clari.biz.market
8.31	clari.biz.top
6.88	clari.biz.mergers
5.96	clari.news.economy
5.41	clari.nb.telecom
5.39	clari.biz.finance
5.24	clari.sports.top
4.63	clari.nb.business
4.51	clari.tw.computers
4.48	clari.nb.ibm
4.21	clari.nb.general
3.95	clari.tw.space
3.68	clari.news.top.world
3.53	clari.news.election
3.50	clari.news.hot.east_europe
3.44	clari.news.gov.international
3.41	clari.news.politics
3.36	clari.news.aviation
3.35	clari.news.law.civil
3.25	clari.news.gov.officials
3.23	clari.news.europe
2.62	clari.news.hot.iraq
2.58	clari.nb.apple
2.31	clari.sports.misc
1.90	clari.sports.baseball
1.60	clari.sports.basketball

clari.biz.finance	
8.64	clari.biz.top
6.98	clari.news.economy
6.12	clari.biz.mergers
5.71	clari.sports.top
5.63	clari.biz.market
5.39	clari.biz.invest
5.31	clari.nb.telecom
5.01	clari.nb.ibm
4.69	clari.tw.computers
	...

clari.news.top.world	
9.89	clari.news.hot.east_europe
9.74	clari.news.election
9.60	clari.news.economy
8.73	clari.news.gov.international
8.53	clari.news.politics
8.28	clari.sports.top
8.25	clari.news.gov.officials
8.02	clari.news.hot.iraq
	...

clari.news.gov.international	
8.73	clari.news.top.world
8.65	clari.news.economy
8.17	clari.news.hot.east_europe
7.39	clari.news.hot.iraq
7.38	clari.news.politics
7.22	clari.news.gov.officials
7.17	clari.sports.top
	...

clari.sports.top	
12.40	clari.sports.misc
10.22	clari.news.economy
9.90	clari.sports.basketball
9.84	clari.biz.top
9.70	clari.nb.ibm
9.59	clari.nb.telecom
8.88	clari.tw.space
8.38	clari.sports.baseball
	...

clari.nb.apple	
17.00	clari.nb.ibm
14.21	clari.nb.general
11.93	clari.nb.telecom
11.59	clari.nb.business
7.00	clari.biz.mergers
	...

Table 4.2: Similarity measures for newsgroups

```
Xref:  news.media.mit.edu clari.news.gov.officials:3216
clari.news.gov.international:46214
From:  clarinews@clarinet.com (NICK DRIVER)
Newsgroups:  clari.news.gov.officials,clari.news.gov.international
Subject:  Economic takeoff masks pitfalls ahead for China
Keywords:  international, non-usa economies, economy, domestic economy,
government officials, government
Copyright:  1993 by UPI, R
Message-ID: <Xyear-chinaUR6f9_3DN@clarinet.com>
X-Supersedes:  <year-chinaUR6f9_3DN@clarinet.com>
References:  <yearURa3c_3DN@clarinet.com> <yearUR254_3DN@clarinet.com>
Date:  Thu, 23 Dec 93 16:48:40 PST
Location:  china
ACategory:  international
Slugword:  year-china
Priority:  release-at-will
Format:  annual, feature
ANPA: Wc:  880/828; Id:  z7864; Src:  upi; Sel:  xxief; Adate:  12-23-N/A;
V: atwill
Approved:  clarinews@clarinet.com
Codes:  yiefyxx., yiedfch., yigoyxx.
Note:  (850) release at will
Lines:  96

BEIJING (UPI) -- China's own predictions of its revival as a world power
moved toward reality in 1993 as its economic engine came to life,
triggering an impressive array of achievements.
While acknowledging the social dislocations and economic dangers their
capitalist-style market reform program has engendered, Chinese leaders have
avoided the difficult political questions that will be posed by the death
of frail 89-year-old senior leader Deng Xiaoping.
With a 1993 gross national product growth rate of 13.5 percent, China's
economic takeoff has captured the hearts, minds and wallets of domestic and
overseas investors.
Foreign investors have come flocking to the ''Middle Kingdom'' and to Hong
Kong's stock market in search of the riches that only 1.2 billion potential
consumers could produce.
Domestic consumers have reacted to the new economic realities by spending
more, saving less and establishing thousands of new private enterprises.
The private economy now accounts for more than half the country's
production by some official estimates.
But the economic boom also has driven up prices and increased urban...
```

Table 4.3: A sample USENET article

filtering purposes, that are mandated by the USENET protocol include Date, From, and Subject. Some suppliers of information provide additional information, which is optional. Such fields include pre-indexed keywords, Organization, (number of) Lines, Sender, Location and so on.

Each candidate article evaluated by YAIF is first converted to its representation before they can be scored. The method of translating documents to their representations is described in the next section.

4.4.1 Extracting Document Representations

As mentioned in Section 3.2.1, the inverse document frequency is calculated using the following formula:

$$idf_k = \log(\frac{N}{n_k})$$

where N is the total number of documents in the "collection", of which n_k contain a given term t_k. The collection of documents consists of articles in the set of newsgroups searched. A profile searches each newsgroup mentioned in the newsgroup field of the profile. It also searches some newsgroups which may not mentioned in the newsgroup field. These are the newsgroups which have some article belonging to it that received feedback through programming by demonstration. The pointers to the articles that received feedback are listed in the "ArtFeedback" list (see table 4.1). Therefore, the "collection" refers to the set of documents in any of these newsgroups and N is the total number of such documents.

The text-indexing algorithm has been implemented as a two stage process. This is because, while the *tf*'s in a document can be computed for an individual document, the *idf*'s cannot be computed until the whole collection of documents has been parsed once.

In the first stage, only the *tf*'s are computed. The text is first cleaned to remove punctuation marks and is then stripped down to individual words. A stoplist is used to eliminate the commonly used words (for *e.g.* "the", "and", "if", *etc.*). The frequency count of the words is computed and stored, as they will be needed to calculate the

final term weights. This is done for each of the N documents.

In the second stage, the *idf*'s are computed. Another parse over the accumulated data yields the document counts for all the terms, *i.e.* the number of documents each term appears in. The product of term frequency and the inverse document frequency gives the term weights and the term vector for each document can be generated.

4.4.2 Assigning Feedback

As mentioned in Section 4.3, the profile cannot be modified in response to user feedback without the document representation. At this stage in the Information Processing module, the document representation is now available and the profile can be modified. In fact, the profile must incorporate the changes due to user feedback before newer documents can be scored since the feedback predates the current round of document scoring.

The profile maintains a list of articles for which feedback was received, if any, and the amount of feedback for each, during the last user session. This can be seen in the items listed under "ArtFeedback" in table 4.1. Once the field vectors for a feedback article are available, the profile is modified as described in Section 3.3.1.

An implementation related issue is that the size of the term vector (especially keywords), could potentially grow infinitely long as new terms are introduced into the profile through feedback. The size of can be controlled by either having an upper limit on the number of terms to be retained, or by having a threshold for the ratio of the lowest term weight to the highest.

4.4.3 Scoring and Selecting Documents

The representations of all documents to be evaluated are now available. The field scores are computed using the scalar products of the profile and document term vectors. The fields considered by YAIF are Newsgroups, Location, Authors and Keywords. The document score is computed as a weighted linear sum of the field scores (see Section 3.2.2).

The number of articles selected for presentation by each profile is proportional to its fitness. The total number of articles to be presented by the agent can be specified by the user. The number of articles to be presented by each profile is calculated and stored in the profile. This is used by YAIF to select the right number of articles for presentation. Only the pointers to the selected articles are stored by YAIF in the profile, as in the "ArtScore" list in table 4.1. The article is not stored but is retrieved upon user request.

While selecting articles, care is taken to prevent presenting the same article twice in different sessions. The profile keeps a list of articles that have been presented to the user during previous sessions, as in the "ArtRead" list in table 4.1. The top scoring articles are compared to this list to prevent repetitions. The list of previously read articles can potentially grow infinitely long. However, USENET articles have expiry dates after which they are no longer available and their Message-ID becomes invalid. The growth of the list of previously read articles is prevented by retaining only those pointers which point to articles that have not expired from the database. A single parse over the Message-ID's collected by YAIF indicates the articles that are still "alive". The upper bound on the length of the list of article pointers is the total number of alive articles in the newsgroups searched by a profile.

Finally, YAIF writes back the updated profile as well as pointers to filtered articles into the profile. To summarize, the possible modifications to the profile caused by YAIF include changes to the profile due to feedback, the list of new articles filtered for the user and the list of previously read articles.

4.5 Efficiency Issues

As with any software system, during the course of developing Newt, a variety of performance tradeoffs had to be made. A discussion of these issues and the lessons learnt would be helpful when developing similar systems which may be developed in the future.

Not surprisingly, the most important tradeoff at the implementation stage was

the classical space-time tradeoff. Re-computing information is time-intensive but saves on storage. Storing information saves on computation time, but needs lots of memory and disk space. The most time-intensive component of Newt by far is the computation of term weights for indexing documents. When several profiles search a number of newsgroups each, it is possible that a newsgroup might be searched by more than one profile. If each profile computes the term weights independently, this would involve expensive recomputation. Instead, if all the indexes are stored, once computed, indexing and matching would take much less time.

As such, if there are enough users, economies of scale might be achieved by pre-processing all documents available in the database. There are two specific suggestions regarding pre-processing which could potentially accrue tremendous benefits. The first is computing term frequencies once and storing them for the lifetime of the article. Since the term counts can be computed for individual documents, these should only be performed once, probably when the article is introduced into the database. The other suggestion is regarding the computation of inverse document frequencies of newsgroups. Inverse document counts can be assumed to not change too much as articles enter and leave the database. Assuming that the articles within a newsgroup typically have similar kinds of keywords, incoming articles would compensate for outgoing articles. Inverse document frequencies for newsgroups could then be computed and maintained as approximations for long periods of time. In the context of USENET newsgroups, computing the inverse document frequencies once a week should be "good enough".

This would be economical if a large number of users are using these indexes for personalized filtering. In fact, it might even be worth exploring the possibility of creating an index server, quite like the NNTP (Network News Transfer Protocol) server. NNTP servers are the backbone of USENET – they are responsible for storing and forwarding articles and also responding to user requests for reading articles. A server similar to NNTP could be implemented which would provide article indexes instead of articles themselves. These indexes could be used to match with profiles and score articles. Once top-scoring representations are found, the article could be

retrieved from the NNTP server using the Message ID of the article.

Chapter 5

Experimental Results

The Personalized News Filtering Agents were tested in two different ways. First, nine real users were given access to the software and asked to use it on a daily basis over a period of two weeks. The other method of testing was by simulating user behavior in hypothetical situations. The methodology of the experiments and the results derived are presented in the following sections.

5.1 Tests with real users

Right from the initial conceptual stages, the intention of this project was to go beyond a "proof of concept". The objective was to actually implement a system which people could use on a daily basis and would want to make it a part of their news-reading habits. This goal was adopted because the underlying system as well as the abstract concepts cannot be well conveyed without an implementation which makes the presentation much more concrete.

"Real" user testing was, therefore, an important component of the performance evaluation. The project was advertised locally and nine volunteers were hired. Those who already spent a substantial amount of time reading Usenet newsgroups were preferred. Users were explained the various components of the system and were taught how to use it. They were then provided access to the system and were asked to use it on a daily basis. Since there wasn't sufficient time to port the code onto

different Unix machines, the system just resided on the host machine. Users were asked to remotely login to the host computer and display the output on the remote monitor. This is possible with the network transparent X Window system. The filtering for each of the users was performed on the host machine.

The users were provided with an initial set of agents so that they had a reasonable starting point in the system and so that they could build other agents by analogy. They were particularly encouraged to construct newer agents suited to their interests. They were asked to read the news retrieved by the agents and provide a lot of feedback. In addition to this, the users were also asked to go through the newsgroups searched by the agents and find articles that the agent should have retrieved but did not. Thus, the users were expected to actively provide as much feedback as possible. As regards to the genetic operators, the users were explained the purpose and consequences of these operators and were given the freedom to decide if and when to use them.

5.1.1 Results

While the system did survive the stress test for the most part, a number of bugs were discovered causing numerous system crashes. Some data were lost as a result of this. The system was used over a period of two weeks. The results for six of the users who used the system for more than half the time they were expected to are shown in figures 5-1 to 5-6. The results contain some noise for a variety of reasons. There was a mild learning curve involved in the initial stages. People also experimented with "what-if" scenarios, to get a feel for the system. Some data were lost due to unexpected crashes of the code, as mentioned earlier. Some users would read the interesting articles and forget to provide positive feedback, or vice versa. Besides, not all users were consistent in providing feedback, trying variations in the early stages of the interaction before crystallizing their interests. However, the results are fairly promising and are presented here nonetheless.

Each graph plots the performance of the whole system over time for each user. For each session, the graph plots the proportion of articles that received positive feedback, the proportion of articles that received negative feedback and the difference

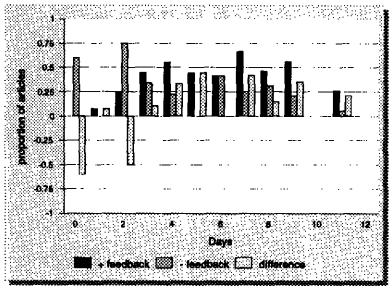


Figure 5-1: Tests with real users: User 1

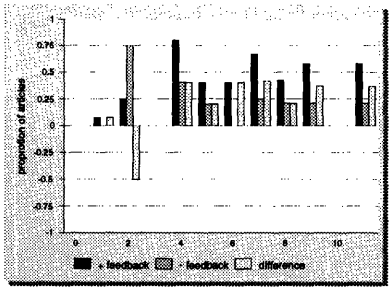


Figure 5-2: Tests with real users: User 2

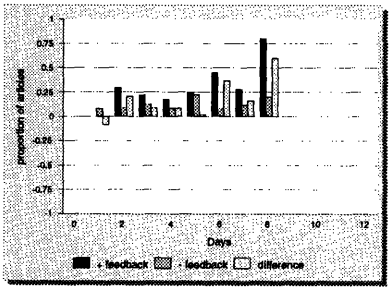


Figure 5-3: Tests with real users: User 3

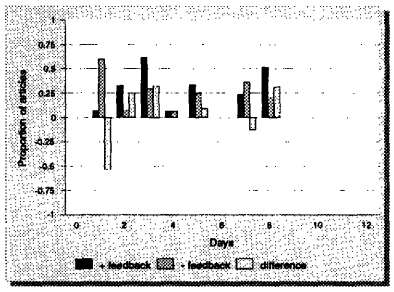


Figure 5-4: Tests with real users: User 4

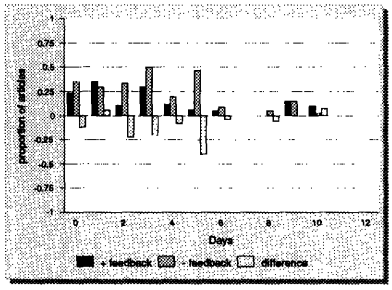


Figure 5-5: Tests with real users: User 5

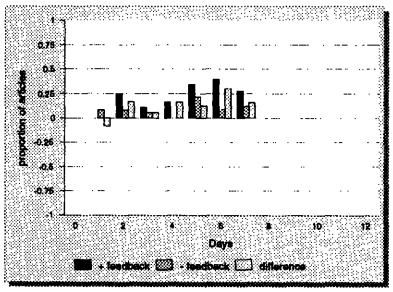


Figure 5-6: Tests with real users: User 6

of the two. For some users, there appears to be a definite pattern of consistent and improving performance (figures 5-1, 5-2, 5-3 and, to a lesser extent, figure 5-6). For others, there is no observable pattern. Agent-based systems are highly interactive systems and their performance greatly depends on how the user uses them, which explains some of the variance in the the results.

5.1.2 Questionnaires

In addition to the numerical data collected above, a questionnaire was also distributed to the users to get feedback on the subjective aspects of the system. A copy of the questionnaire is shown in Table 5-7. Some of the questions were directly pertinent to the system and dealt with ease of use and user-friendliness. The other questions dealt with the larger concept of Interface Agents for Personalized filtering and what the users felt about the broader issues involved. The answers to the questionnaire are summarized below. The users were candid about the fact that their responses were mostly influenced by their background and original expectation levels. This did produce varied responses from differing perspectives.

People responded quite positively to the graphical user interface. The interface was easy to use for the basic tasks of reading news and providing feedback. The use of colors to indicate association of articles to agents was well received. An area where more needs to be done is in explaining why an article has been selected by the agent. An interesting suggestion is to even provide some of the articles which have been rejected with reasons.

The development of a good "agent model" by the users is as important as the agents developing a sophisticated user model. A prerequisite to trusting the agent is understanding its behavior. People had mixed reactions when asked if they could develop good agent models. The visual representation of anthropomorphic agents was quite enjoyable, and easy to understand. However, since the internal state of the agent is quite complex, better tools appear to be necessary to explain the agent's behavior. Users who had previous knowledge of the underlying text processing module found this system to be a powerful tool and could make good use of it. On the other hand,

Performance Evaluation

(Please answer each question on a scale of 1-7. You are also encouraged to provide additional comments explaining your score.)

Usability Issues

- Graphical interaction: Did you find the graphical elements on the screen (agents, menus, lists etc) intuitive enough to use?
- Graphical Communication: Were you comfortable interacting with the agent (checking filtered news, feedback, querying the agent)?
- Did you have access to all the information you needed about the agent?
- Did you develop a good understanding of why the agent would get the articles it would? do you feel you were able to develop a good ''agent model''?

Agency Issues

Agents

- Personalized: do you think your agents kept presenting a higher proportion of interesting articles over time?
- Continually adaptive: would you say, your news interests changed over time? could you give an example, if so?
- If yes in the last question, did you think your news agents adapted to this change?
- Serendipity: Did your agents ''surprise'' you with information from a certain category you didn't expect, but found really interesting?

Communication

- Do you have a good enough model of the agents? Were they communicative enough?

Competence and Trust

- Trust: Do you trust your news agents enough to let them serve all your information needs?
- Are you comfortable with the concept of Agents doing information gathering for you?
- What privacy concerns, if any, would you have concerning the fact that the agents have access to the news interests of the user?

Figure 5-7: Testing with real users: the questionnaire

users with no such background had problems correlating the agent's behavior to its internal state and found some agent actions inexplicable.

Two weeks of daily interaction was not a sufficient enough time for some users to get used to the system, understand the mechanism, train the agent(s) and see the effects of a well-trained agent. Users who had previous knowledge extracted a lot of use in a very short time (see figures 5-1, 5-2). Some users found the system started to get useful towards the end of the testing phase (see figures 5-3, 5-6), while others could not judge or perceive any improvement.

In communicating with the agent, one of the problems was the delayed response of the agent. This problem exists because any changes made to the profile causes the agent modify its behavior only after a day, since the new data arrives on a daily basis. This lack of immediate response prevents efficient communication.

There were mixed responses to the question of trusting the agent. Users who understood the agent actions well and knew their abilities and limitations were very comfortable with the idea of trusting their agents for some of their information filtering needs. On the other hand, users who could not understand agent behavior had reservations trusting the agent.

As regards privacy concerns, for the most part users would like to keep the profiles private and only release information that they deliberately wish to. The information maintained in the user profiles can find a number of malicious uses, not the least of which is junk mail. The profiles should probably be safely encrypted and stored locally, to minimize the chances of reaching undesirable destinations.

To summarize the lessons learned from testing in a real environment, this system received an encouraging response from users and the approach appears to be pointed in the right direction. At the implementation level, some more work needs to be done before a system like this finds widespread use. In particular, good explanations have to be provided and the agent internal state needs to be better visualized.

5.2 Tests with simulated users

A learning filtering system must specialize to current user interests, adapt as they change over time and explore newer domains prospecting for potentially relevant information. In this section, the performance of Newt is evaluated for its ability to specialize, adapt and explore. A number of experiments have been performed using simulated user input under controlled situations. This approach allows us to focus on and study the behavior of each sub-component of the news filtering system which would otherwise be impossible in an unconstrained environment. The experiments and the results are described in the following. A sample scenario is presented along with each of the experiments as a vehicle for introducing the user context.

5.2.1 Specializing to User Interests

When the user has a known specific interest and is providing consistent feedback, a responsive system should converge to the set of terms which define that interest. The following experiments were conducted with the aim of evaluating the ability of Newt to specialize to user interests.

Scenario 1

Lee is a graduate student who is writing a term paper on the present political situation in China. He is interested in reading current articles which report on news pertaining to China. For the purpose of his term paper, only articles referring to China are relevant for him. All other articles are irrelevant. He diligently provides positive feedback for the relevant articles.

Experiment 1

The only agent in the system is a "China" agent, with a population of one profile. Effectively, the GA is not used at all and only the feedback learning system is tested. The initial profile is nearly empty and is shown on the left hand side of Table 5.1. The system is iteratively used for a number of user sessions. The number of articles

shown to the user is fixed and the articles are presented in a prioritized order. Each session involves reading the articles retrieved by the profile and providing positive feedback for articles about China.

Note that the location fields are not used in this experiment nor are individual keywords selected while providing feedback. The only feedback provided is for the document as a whole. This constraint is imposed so that the result can hold for most situations. In most filtering environments, at least the text of the article is always available. Therefore, no pre-indexed fields are used to avoid loss of generality.

Results

The performance measures used are precision and recall. These measures are well known and commonly used to evaluate the performance of information retrieval systems. *Precision* is the number of retrieved articles that were relevant. *Recall* is the number relevant articles that were retrieved. This experiment evaluates the behavior of precision and recall over successive user sessions. The database does not change over the duration of this experiment. Recall is calculated by manually going through each of the articles in the newsgroups searched by the profile. Each article is classified as relevant if the major topic of the news story was associated with China. Fortunately, the articles were straightforward to classify and there were barely any borderline cases.

Figure 5-8 shows the results of the experiment. The figure consists of four graphs. Each graph is a plot of precision versus recall, at different points in time, as indicated by the label. When a search is performed, the articles are effectively sorted by their similarity scores. If all the articles are retrieved for the user, the recall is 1, but the precision is very low. If none of the articles are retrieved, recall is 0, but the precision is 1. For intermediate numbers of articles retrieved, different values of recall and precision will be observed. By using the number of articles as a parameter, a graph can be drawn connecting the set of all possible precision-recall pairs. The graphs help us compare the performances of the each search for all possible values of the parameter.

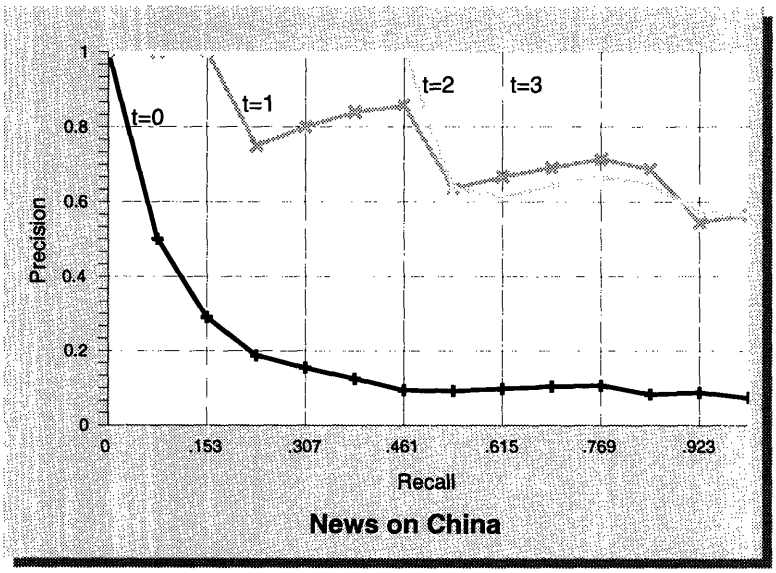


Figure 5-8: Specializing to user interests: empty initial profile.

Initial Profile	*Final Profile*	
newsgroups: 0.2	newsgroups: 0.2	
clari.news.gov.international	clari.news.gov.international	1
learningrate: 0.1	learningrate: 0.1	
	keywords: 2	
	china	0.580
	pakistan	0.263
	hong	0.254
	bhutto	0.209
	kong	0.169
	deng	0.163
	jiang	0.155
	chinese	0.154
	weapons	0.149
	myers	0.147
	worker	0.135
	dee	0.135
	chinas	0.133
	allegation	0.130
	deny	0.130
	missile	0.120
	death	0.118
	beijing	0.117
	reforms	0.106
	shipments	0.104
	accompanied	0.104
	visit	0.103
	peasant	0.103
	cooperation	0.095
	efforts	0.093
	communist	0.092
	...	

Table 5.1: Static user interests: effect of feedback

Initially, when the profile is empty, the articles retrieved are in a random order. The graph for the initial profile (labeled "$t = 0$") is the result when all articles are scored zero since none of them match the empty profile. Since this iteration is a random ordering of articles, it is possible that none of the retrieved articles are relevant. In that case, the user needs to program the agent by demonstration by providing a few examples of relevant documents. After feedback is provided to the relevant articles, the profile is modified and the search is performed again. The search is much improved and the result is shown by the next line graph (labeled "$t = 1$"). Successive iterations keep improving the performance of the search.

The profile is quite successful at converging on the terms commonly occurring in articles relating to "China". This can be seen from the final (at $t = 3$) profile as shown in Table 5.1.

It can be seen that given an empty initial profile (so there is nothing to unlearn) and with consistent user feedback the system is successful at specializing to user interests. The caveat is that the user interest must be amenable to a keyword-based description (see Chapter 6).

Scenario 2

Lee is now working on a similar paper, except this time he is interested in articles on Russia. He decides to continue using the same agent which was used earlier and has been specialized to China, expecting the system to specialize now to Russia.

Experiment 2

The profile specialized for news on China is now expected to focus on Russia. The initial profile now is the final profile of the previous experiment. This experiment would evaluate the ability of the system to specialize when the initial profile is not empty, but has a previous bias.

Results

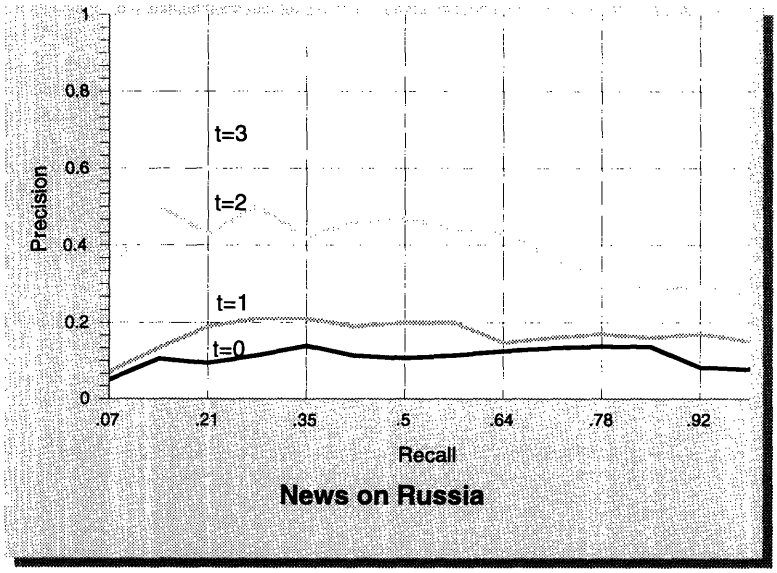


Figure 5-9: Specializing to a new interest.

The results are shown in figure 5-9. The graph is a precision vs. recall graph at different time points. The graph does show an improvement over time. However, this time around, the improvement is much slower. This is primarily because the previously learnt "Chinese" terms are still in the profile which affects the system performance from the "Russian" point of view.

Another potential problem is that in the transition period, the profile is neither specialized to Chinese articles nor Russian articles. Besides, there is an implicit "AND"ing of terms — an article which refers to both China and Russia will be rated higher than one referring to either one alone. This might not be what the user desired.

When an empty initial profile is used, the system converges relatively quickly to the desired set of interests. However, if the initial profile has some terms which are not related to the new user interests, then the profile takes longer to converge. This is because the profile needs to unlearn the terms which are already in there, but are of no use.

As can be seen from the two preceding experiments, a profile is capable of narrowing down the search and focusing on the current user interest. It specializes quite well. However, if user interests change, the profile has strong inertia to adapting to the change. Also, in the transition phase the profile could be specialized to neither the old nor the new interest. This approach using relevance feedback appears to work well for specializing but not as much for adapting when user interests change.

5.2.2 Adapting to Dynamic User Interests

To bolster the adaptive capabilities of the system, the system has been modeled as a team of specialized profiles. While each of the profiles is highly specialized in a narrow domain, the team as a whole reacts to changing user interests by controlling the numbers of each kind of specialized profiles. When user interests change, some of the profiles specialized to the old user interests are replaced by other profiles which serve the new user interest. This provides an optimal combination of specialization and adaptability.

In this section, the performance of the GA in evolving the population of specialized

profiles is tested and the results presented.

Scenario

Lee now wants to use Newt for all his news interests. He has a variety of interests and they keep changing over time. However, before using it for an extended period of time, he wants to test it over a static data stream, by simulating multiple days usage on the same set of newsgroups and documents.

Experiment

Since there many competing user interests, a population size greater than one is necessary to model the dynamics of user interests. The system is set up as a single-agent system, with a population size of twenty. Initially, the population is oriented to the current major user interest, namely "Business". Most of the profiles in the population specialize in Business news, while a relatively fewer number specialize in Politics. The user interests then gradually changes towards Politics. The profiles which retrieve political articles receive positive feedback and their fitness increases. At the same time, profiles which retrieve other kinds of articles (including Business articles) receive negative feedback and the corresponding fitness decreases.

Results

The results of the experiment are shown in Fig 5-10 and Fig 5-11. The graphs show the evolution of the population of profiles as user interests change over time. The profiles have been grouped together based on their specialization. The initial population in each of the graphs is identical. Fig 5-10 shows the effect of varying the rate of change of user interest. Fig 5-11 shows the effect of varying the retention rate of the population.

Users would typically provide positive and negative feedback to the agents, while using the real system. For this experiment, the same effect is simulated by assuming an imaginary user with a certain bias, which represents the change in favor of the new interest. The bias is a parameter introduced solely for the purpose of this experiment

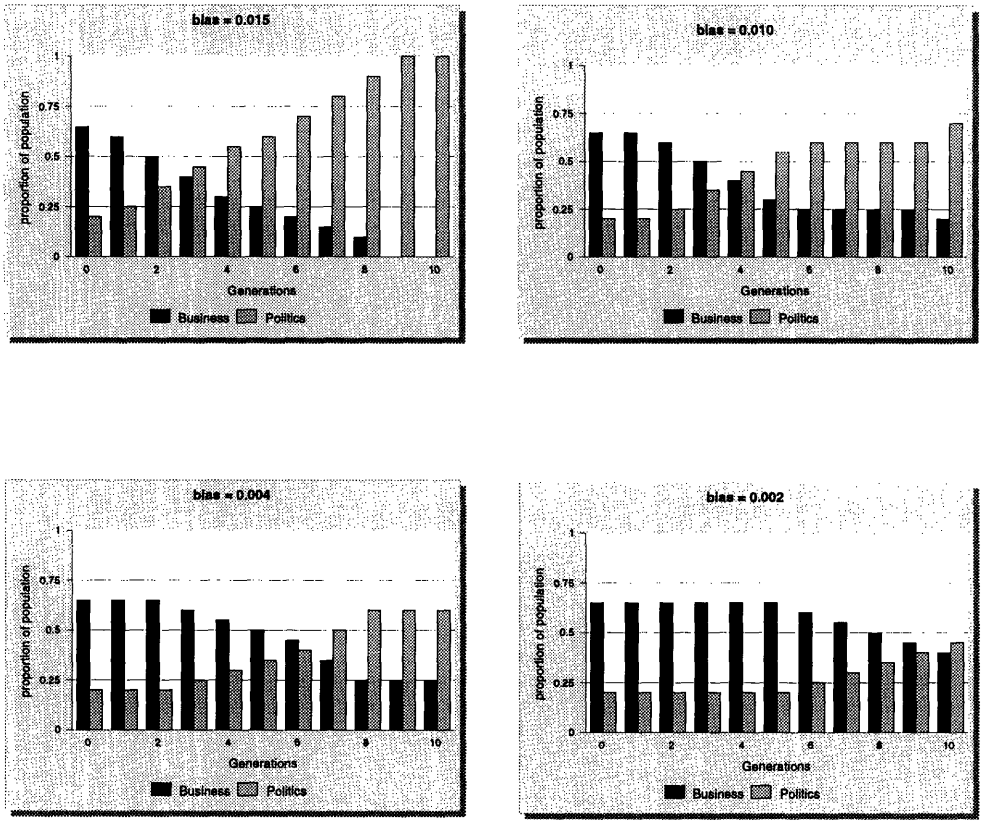


Figure 5-10: Adapting to user interests: varying the bias

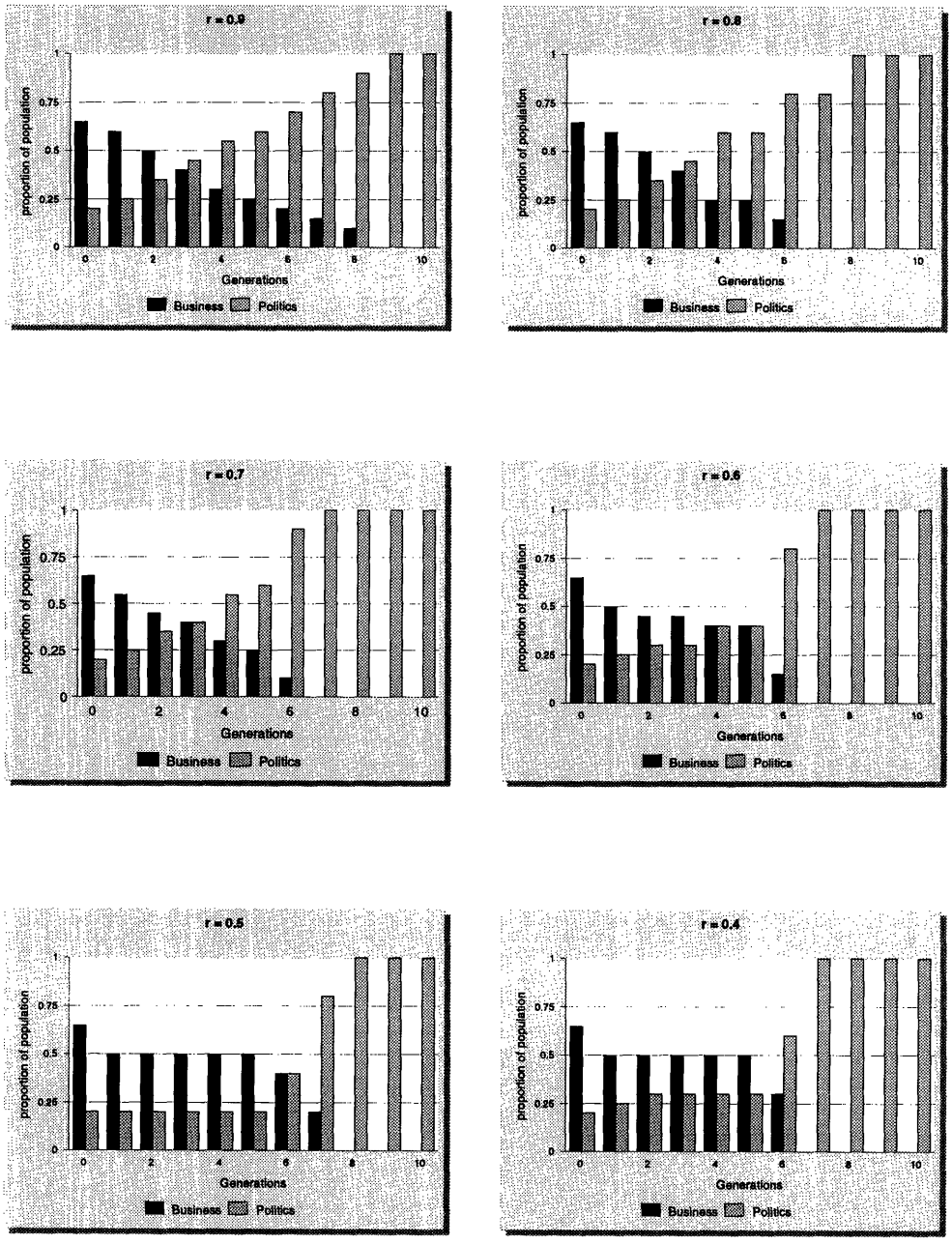


Figure 5-11: Adapting to user interests: varying retention rate

which measures the rate of change of user interest. Bias is defined as the amount by which the fitness of certain profiles increases and that of the other profiles decreases every generation. Thus, if the bias is 0.01 in favor of politics, the fitness of profiles retrieving politics articles increases by 1% every generation and the fitness of the others decreases by 1%. The effect of a real user providing consistent feedback is simulated by setting the appropriate bias for the simulated user.

The retention rate (r) is set to 0.9. Offsprings of profiles are assumed to specialize to the same category as their parents. For simplicity, the crossover genetic operator is not used since the category of the offspring would be difficult to determine. Only the mutation operator is used. Figure 5-10 shows the evolution of the population for different biases. The bias is held constant for the entire duration plotted. Each graph plots the proportion of profiles in the population which specialize in business and politics articles, respectively, in successive generations. When the bias is sufficiently high, the selective pressure in favor of politics is much stronger and the constitution of the population changes relatively quickly. On the other hand, when the bias is lower, the change in the population is much more gradual and takes longer.

In this case, the problem of profiles being half interested in two different topics does not arise as before. This is because feedback for each topic goes to the appropriate specialist. A Business profile would retrieve Business articles and only receive feedback for those. When the user interest changes, it will receive less positive feedback and will eventually be eliminated. However, the case where a Business profile will receive feedback for a Politics document will not arise. Thus, a profile will not have to specialize in two topics — it will either specialize in one, or be eliminated from the population.

The effect of varying the retention rate is shown in Fig 5-11. The bias in this case is fixed at 0.015. The higher the retention rate, the more stable the population is likely to stay across successive generations and more gradual will be the change. When the retention rate is high, the change is very gradual since the maximum change across two generations is constrained. Lowering the retention rate sets the stage for potentially large changes in the population. In the figure, as the retention rate is

lowered, the change in the constitution of the population is much more drastic and discontinuous when it takes place.

The advantage of lowering the retention rate is that the population can react much faster. The disadvantage is that the changes are much more erratic and less controllable. The actual value should be set to achieve an optimal combination.

5.2.3 Exploring Newer Domains

To help with both adaptation as well as serendipity, the system should explore for new kinds of information. Exploration for new information domains could either precede or succeed a change in user interest. When it precedes the change in user interests, the system apparently exhibits the serendipity behavior. Conversely, when the change in user interest precedes the exploration, it appears as if the system is exhibiting better adaptive behavior in response to the change in user interests. In either case, however, the system needs to be able to guess the future interests based on currently known habits.

Scenario

Lee is spending a leisurely Sunday afternoon at home and wants to be entertained and educated with interesting tidbits of information. He wants to be surprised with new information, but preferably related to something he already is interested in.

Experiment

To better understand the process of mutation and focus on its exploratory behavior, the initial profile is successively mutated over several generations. The initial profile provides a starting point and the lineage of mutated offspring represents the exploration process. This helps to evaluate the exploratory behavior of Newt. The mutation operator is defined in Section 3.4.2 and the table of similar newsgroups used is shown in table 4.2.

Results

Table 5.2 shows the results of the experiment. The newsgroup field determines the search domain of the profile and indicates the kinds of articles it is likely to retrieve. Hence, mutation only operates on the newsgroup field thereby directing the exploratory process. The results are shown with different initial profiles.

The initial profile for each test run is shown in the left column. The initial profile is successively mutated over many generations. This process is repeated a large number of times for the same profile to average out the random effects of the mutation operator. So for every initial profile, a probability distribution for the final profile is derived. The distribution gives the probability of a newsgroup being in the final profile.

For each test run the initial profile, the final profile probability distribution and the number of generations is indicated. Continual mutations produce some interesting results. The system appears to be exploring in the neighborhood of the initial profile and coming up with interesting newsgroups that are quite similar to the ones in the initial profile. The implication is that if the user likes a newsgroup in the initial profile, say, *clari.biz.finance*[1], then it is quite likely (with probability $p = 0.19$, see Table 5.2) that he is also interested in the newsgroup *clari.biz.mergers.*

5.2.4 Testing the complete system

Having tested the individual components of the system in the preceding sections, the complete system is now tested for its overall behavior. In this test the precision and recall measures are used to evaluate the performance. These measures provide a precise indication of the system behavior.

Scenario

Encouraged by the preceding results, Lee wants to use the system so that it would simultaneously explore and specialize. He decides to use the system to read news

[1] clari.* news hierarchy is a pay service provided by ClariNet Communications Corp.

Initial Profile	*Final Profile Distribution*	
clari.biz.finance	clari.biz.market	0.24
clari.biz.invest	clari.biz.mergers	0.19
clari.biz.top	clari.news.economy	0.18
	clari.sports.top	0.17
	clari.biz.top	0.14
	clari.biz.invest	0.04
	clari.biz.finance	0.03
clari.sports.top	clari.nb.telecom	0.22
clari.sports.baseball	clari.sports.top	0.22
clari.sports.misc	clari.nb.ibm	0.19
	clari.sports.basketball	0.16
	clari.sports.misc	0.16
	clari.sports.baseball	0.06
clari.news.gov.international	clari.news.hot.east_europe	0.26
clari.news.gov.officials	clari.news.hot.iraq	0.19
clari.news.top.world	clari.news.politics	0.17
	clari.news.top.world	0.17
	clari.news.economy	0.17
	clari.news.gov.international	0.03
	clari.news.gov.officials	0.02
clari.news.europe	clari.news.politics	0.23
clari.news.hot.east_europe	clari.news.hot.iraq	0.22
clari.news.gov.international	clari.news.hot.east_europe	0.20
	clari.news.top.world	0.14
	clari.news.economy	0.13
	clari.news.gov.international	0.03
	clari.news.europe	0.03

Table 5.2: Dynamic user interests: effect of mutations

about the situation in former Yugoslavia.

Experiment

The system is set up with a single agent which retrieves news articles pertaining to former Yugoslavia. The profile population size is five. One of the international newsgroups is randomly added to the newsgroup field of each of the initial profiles. A single keyword, "bosnia", is added to the keyword field of each profile, so as to provide a reasonable starting point for the system. Every 3 sessions, a new generation of the population is created.

The performance measures used are precision and recall. Recall is calculated by manually identifying the documents that are relevant. Documents pertaining to former Yugoslavia in any of the newsgroups are classified as relevant. The scope of this experiment is restricted to the newsgroups which carry newswire stories. An arbitrary threshold (0.05) for document scores is set. Only documents scoring above the the threshold are retrieved for presentation. To achieve the desired goal of high recall and high precision the similarity scores of relevant documents must lie above the threshold and the scores of irrelevant documents must lie below the threshold.

Results

The results of the experiment are shown in figure 5-12. The graph reveals some interesting patterns. Recall typically changes only when the next generation is created. At other times, it tends to stay steady. Within a generation, precision tends to rise upto a certain point and levels off after that. Precision sometimes shows a slight dip when the next generation is created.

The behavior of the system recall can be explained as follows. A higher proportion of the relevant articles will be retrieved only if the profiles search all the right newsgroups *i.e.* newsgroups which contain the relevant articles. If the set of newsgroups being searched is limited, recall is constrained and tends to stay steady. The set of newsgroups being searched changes when the mutation operator is applied. The mutation operator is applied only when a new generation is created. If the newly in-

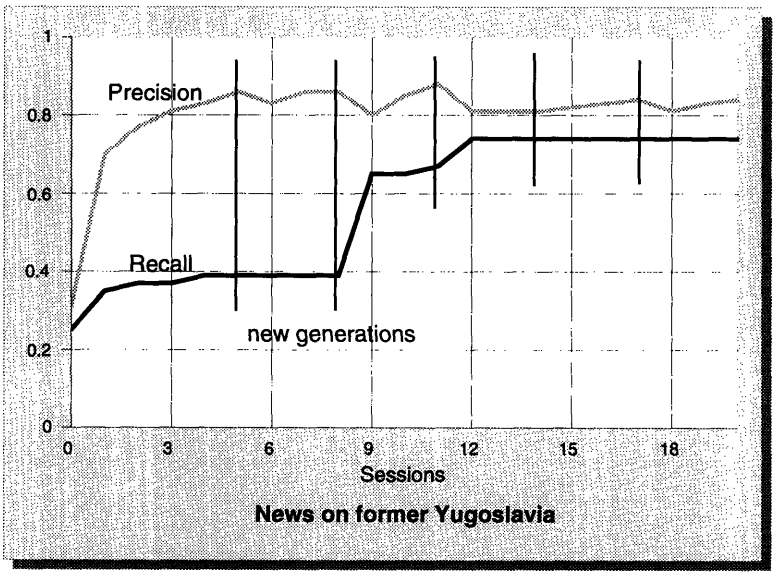


Figure 5-12: Testing the complete system.

troduced profiles contain the right newsgroups, recall will show sudden improvement (see the new generations created during sessions 8 and 11 in figure 5-12). If the new profiles search newsgroups which do not contain any irrelevant documents, recall will not show much change in the new generation. The only exception to the preceding analysis is in the initial stages of the experiment where the profiles are almost empty and do not retrieve many relevant documents. In the extreme case when the profile is completely empty, it would not recommend any document for reading and the recall would be zero. Relevance feedback in the initial stages helps both precision and recall.

The behavior of system precision depends on relevance feedback. The feedback provided by the user helps the system to distinguish between relevant and irrelevant documents. In particular, relevance feedback helps the system to assign lower scores to irrelevant documents, thereby improving precision. This effect is similar to the specialization described in Section 5.2.1. When a new generation is created, the precision may suffer a bit. This is because the newly introduced profiles are likely to recommend some irrelevant documents. However, relevance feedback in the succeeding sessions helps to "fine tune" the new profiles. This restores the precision to its original level.

Thus, experimental evidence suggests that relevance feedback helps improve the overall precision of the system while genetic operators help improve overall recall.

Chapter 6

Conclusions

A personalized information filtering system must be able to specialize to user interests, adapt as they change and explore the domain for potentially relevant information. The implementation of the personalized information filtering agents demonstrates that Relevance Feedback and the Genetic Algorithm can be used together to build personalized information filtering systems. The results of real user tests were promising. Users who could get familiar with the system used it as a powerful news filtering tool. However, we discovered that some more work needs to be done to help the users understand the agent better. Results of simulated tests under controlled environments were more conclusive. They show that Relevance Feedback as a technique is sufficient for profiles to specialize to static user interests. However, it cannot be used to adapt as user's interests change. Results further show that genetic algorithms are a promising approach for modeling adaptation and exploration in an information filtering system. More precisely, relevance feedback helps improve the overall precision, while genetic algorithms help improve the systemic recall.

Several criticisms of our approach have been made. One of the questions asked most frequently, by those watching demonstrations of Newt is that of *serendipity*. This question is of most concern to people from the media industry, especially newspapers. The question is, how would the system be able to recommend relevant articles that the user could not possibly have known to ask for in the first place? The genetic algorithm approach provides at least a partial solution to the serendipity problem. If

a user's preferences are extremely stable, she can set the mutation rate low enough so that the articles retrieved are mostly from the current set of newsgroups. On the other hand, a user can turn the "serendipity knob" higher, if she really likes to continually receive information about different topics. An advantage is that a user can have serendipity when she wants, as much as she wants. The solution is limited, however, since the only serendipity the system is capable of producing is searching for the same keywords as before in other newsgroups.

Another criticism of this system is that there are shortcomings to a keyword based approach to filtering. Needless to say, there are limitations to the kinds of concepts that can be expressed in terms of keywords. As a result, there are some news articles which will be difficult to retrieve using keyword-based searches. A concept that is difficult to express using just keywords is point-of-view. For example, it is almost impossible to decide if a news editorial on the budget represents either the liberal or the conservative point of view based only on the keywords used in the editorial. Other concepts that cannot be conveniently expressed merely in terms of keywords include humor, rhetoric, satire, gossip, *etc.* These expose the shortcomings of a keyword based system. However, keyword based systems work fairly well for most newswire articles which typically deal with a particular event, person, place, or thing and have enough keyword clues in the articles that can be exploited (see Section 5.2.1). Notice that Newt is not wedded to a keyword-based search engine. A system very similar to Newt could be built based on the framework described earlier in Chapter 3 which would use a different search engine. The document and profile representations and the effect of feedback on the representation would change.

Another criticism is that our research is mostly focussed on newsgroups which carry newswires. How would the approach generalize to other kinds of information? As mentioned earlier, these constraints were imposed mainly due to the quality of information and computing power. Some newsgroups were excluded from the analysis because of poor quality of keywords, in terms of spelling and consistency of usage as well as for computer processing constraints. The former is definitely a greater problem, since the latter can be overcome with approximate but efficient algorithms.

Given sufficiently good quality of data, the filtering approach described in this thesis should be generalizable to other data streams as well.

A brief discussion of the impact of personalized information filtering agents on the conventional media industry would not be inappropriate. The impact of systems such as Newt (or possible commercial grade versions) on the traditional media industry, as we have known, can be potentially quite significant. The trend towards automating substantial components of a news editor's filtering responsibilities is likely to continue, even if the ideal goal of complete automation may be quite distant, if not unreachable. A traditional newspaper serves many purposes — information, entertainment, communal needs, general interest, setting the national agenda, *etc.* The niche for personalized, prioritized information is likely to be the first one for which electronic surrogates for traditional newspapers will succeed. Other niches may take longer to occupy and are likely to be more difficult to automate. Nonetheless, the locus of control is tending to move closer to the consumer, who is likely to have a much greater say in regulating the incoming flow of information. It empowers producers of information to reach a large number of consumers and vice versa *i.e.* there will be fewer middlemen in the process. Personalized Information Filtering Agents take us a step closer towards managing the information rich world of the future.

Chapter 7

Future Work

7.1 The Filtering Engine

YAIF is the filtering engine in the current implementation of Newt. It is inadequate on a number of counts and can be improved upon. There are two interesting directions for future work concerning the filtering engine. One area of future work is to make incremental improvements to the existing filtering engine without changing the basic keyword based search engine. The other direction for future work is to make fundamental changes to the engine by replacing it with another that uses a different filtering method. Both these directions for future work are discussed below.

A disadvantage of YAIF is that text full indexing and matching takes too much time. Hence, agents cannot search for articles in real time. For example, if the user creates a new profile, it is difficult to test it right away since the filtering module is executed off-line. Even if the filtering module is executed on demand, it would take too much time to index all the articles in a newsgroup. One way to solve this problem is to keep pre-computed indexes for all the articles available for as long as the article is alive. An interesting challenge is to devise an efficient and incremental text indexing algorithm. The term frequencies can be computed for individual documents as they enter the database. However, the inverse document frequencies need to be recomputed incrementally as documents enter or leave the database.

Instead of trying to make improvements to the current keyword-based filtering

engine, a new engine could be implemented using a completely different filtering method. For example, social or economic filtering could be used instead of the current cognitive filtering engine. The modular design of Newt makes it quite easy to replace the current filtering engine with another one. Taking this one step further, different agents in the system could have different filtering engines for recommending documents to the user. Each agent would filter documents using a different approach. However, the documents would be presented to the user through a single interface. The advantage is that the user will have a common interface to support her diverse filtering needs.

Keyword-based filtering schemes have their limitations as mentioned in Chapter 6. An option to overcome these limitations is to use techniques from the fields of natural language understanding and knowledge representation. For instance, FRAMER [17] is a knowledge representation library which can draw analogies in natural language texts. So, for example, if FRAMER infers that the news story A is analogous to the news story B, if the user likes document A, the agent could also recommend document B. Another option is to use the contextual/structural retrieval paradigm [30]. In this framework, the user can also provide qualitative relevance feedback explaining *why* a document is not relevant. It is worth exploring the possibility of using these approaches for building better filtering engines. There are certain issues that one must bear in mind while designing filtering systems. Representations of user interests must be sufficiently flexible to allow incremental modifications. The search engine must also be able to provide good reasons for selecting or rejecting a document. The representations must also satisfy the "clustering" requirement *i.e.* people whose interests are similar would also have similar representations. This is necessary for exploration, where, by slightly modifying the profile, the required result is a commensurate change in the documents selected.

7.2 Genetic Algorithm

The Genetic Algorithm promises to be an interesting approach for modeling adaptive filtering systems. However, the GA approach was not fully exploited in this work due to time constraints. There are many directions for future work relating to GAs for information filtering.

One area for future research is in automating the application of genetic operators based on user actions. If the the user starts providing too much negative feedback, increasing the frequency of newer generations may perhaps speed up the adaptation. The GA used in this system involves a number of parameters, such as the retention rate, mutation rate, *etc.* These could also be automatically modified based on user behavior. For example, if a user consistently provides positive feedback for documents retrieved serendipitiously, perhaps the the number of mutations should increase.

Another interesting problem is to maintain diversity in the population by preventing profiles from getting too "close" to each other. Different profiles should ideally cover non-overlapping regions of the information space. Calculating the proximity of profiles in the vector space representation is easy. The interesting problem is to devise a scheme which applies selective pressure on neighbouring profiles to move away from each other. The economic model for optimizing computational resources, as used by Baclace [2], could possibly provide a solution.

Users typically have special needs that must be integrated into the GA model. For example, a user might create a special high priority profile which must never be eliminated by the population, no matter what. Some users might not want any profile to be eliminated unless it gets explicit negative feedback. The GA model described in this thesis must be extended so that the "special" members of the population are handled in a special way. These issues need to be addressed in future work.

7.3 Agent model

Inexplicable agent behavior is perhaps the biggest obstacle to users accepting agent-based systems. It is critically important that the user develops a good understanding of the agent behavior. An area of future research involves creating tools to help users develop satisfactory agent models. There are two approaches to solving the problem. First, the internal state of an agent should be accessible and comprehensible to the user. Second, the agent has to be able to provide meaningful explanations for all its actions.

More work needs to be done in providing users with a better view of the internal state of the agent. One way to facilitate this is to allow better views and interfaces. For instance, consider the following situation that arose when testing the system with real users. Sometimes the user wanted to see the list of newsgroups searched by each profile separately, while at other times the user liked to see all the newsgroups searched by the entire population of profiles. The user thus needed different views of the internal state. In future implementations, perhaps, the user should also be allowed to experiment with "what-if" scenarios. For example, the user might want to know what kinds of articles will be retrieved if an extra keyword is added to the profile.

The second method for enhancing the understanding of the agent is to provide explanations for the agent's behavior. In the current implementation, the only information stored after the scoring process is done is the final score of the document. An alternative, for instance, could be to remember the five terms which contributed the most to the document score. Thus, a reasonable explanation would be available to justify why the document was selected. Interfaces must also be provided for the user to query any action of the agent.

7.4 The User Interface

The user interface is another interesting area for future research. The amount of user interaction required can be further reduced by use of more intelligent interfaces. "Smart badges" have been used for user modeling [35] to detect the physical location of the user. A possible application to our filtering system is to automatically assume positive feedback if the user spends a long time in front of the computer when the document is displayed. This would lessen the burden of interaction for the user.

The interface could also take advantage of multiple media. For example, some users would like to *read* textual news, *hear* it if they're driving, *view* it if it involves video and so on. An interesting problem is to design an intelligent agent that can communicate using multiple media and can offer the user seamless navigation across media.

Bibliography

[1] Ackley, D., Littman, M., Interactions between Learning and Evolution, *Artificial Life II*, Edited by C. Langton, C. Taylor, J. Farmer and S. Rasmussen, Addison Wesley, 1991.

[2] Baclace, P. E., Personal Information Intake Filtering, Bellcore Information Filtering Workshop, November 1991.

[3] Berners-Lee, T., Cailliau, R., Groff, J., and Pollermann, B., World-Wide Web: The Information Universe, *Electronic Networking: Research, Applications and Poilicy*, Meckler Publications, 2(1), Spring 1992, pp. 52-58.

[4] Chin, D., Intelligent Interfaces as Agents, Intelligent User Interfaces, ACM Press, 1991, pp. 177-206.

[5] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science, Vol. 41, No. 6, 1990, pp. 391-407.

[6] DeJong, K.A., Adaptive System Design: A Genetic Approach, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 10 No. 9, 1980.

[7] Belkin, N.J., Croft, W.B., Information Filtering and Information Retrieval: Two Sides of the Same Coin?, *Communications of the ACM*, Vol. 35 No. 12, pp. 29-38, 1992.

[8] Cypher, A., ed. *Watch what I do: Programming by Demonstration*, MIT Press, Cambridge, MA, 1993.

[9] Fischer, G., Stevens, C., Information access in complex, poorly structured information spaces. *Human Factors in Computing Systems CHI'91 Conference Proceedings*, 1991, pp. 63-70.

[10] Foltz, P. W., Using Latent Semantic Indexing for information filtering, Proceegings of the ACM Conference on Office Information Systems, ACM/SIGOIS, New York, 1990, pp. 40-47.

[11] Foltz, P.W., Dumais, T., Personalized Information Delivery: An Analysis of Information Filtering Methods, *Communications of the ACM*, Vol. 35 No. 12, pp. 51-60.

[12] Gauch, S., Smith, J. B., An Expert System for Searching Full Text, Information Processing and Management, 25(3), 1989, pp. 253-263.

[13] Goldberg, G., Nichols, D., Oki, B.M., Terry, D., Using Collaborative Filtering to Weave an Information Tapestry, *Communications of the ACM*, Vol. 35 No. 12, pp. 61-70.

[14] Gordon, M., Probabilisitc and Genetic Algorithms for Document Retrieval, *Communications of the ACM*, Vol. 31 No. 10, Oct. 1988.

[15] Grefenstette, J.J., *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, The Robotics Institute of Carnegie Mellon University, Pittsburgh, 1985.

[16] Grefenstette, J.J., Optimization of Control Parameters for Genetic Algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 16 No. 1, 1986

[17] Haase, K. B., FRAMER: A Persistent Portable Representation Library, Submitted to: European Conference on Artificial Intelligence, 1994.

[18] Holland, J.H., Adaptation in Natural and Artificial Systems, *An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, University of Michigan Press, Ann Arbor, 1975.

[19] Hinton, G.E., Nowlan, S.J., How Learning can Guide Evolution, *Complex Systems*, 1: 495-502, 1987.

[20] Horton, M., Standard for Interchange of USENET Messages, RFC-1036, USENET Project, December 1987.

[21] Kahle, B., An Information System for Corporate Users: Wide Area Information Servers, Thinking Machines technical report TMC-99, April 1991.

[22] Kay, A., Computer Software, *Scientific American*, 251(3), 1984, pp. 53-59.

[23] Kozierok, R., Maes, P., A Learning Interface Agent for Scheduling Meetings, ACM-SIGCHI International Workshop on Intelligent User Interfaces, Florida, January 1993.

[24] Kozierok, R., A Learning Approach to Knowledge Acquisition for Intelligent Interface Agents, SM Thesis, Department of Electrical Engineering and Computer Science, MIT, May 1993.

[25] Lai, K., Malone, T., Yu, K., Object Lens: A "SpreadSheet" for Cooperative work. ACM Transactions on Office Information Systems 5(4), 1988, pp. 297-326.

[26] Lottor, M., Internet Growth (1981-1991), Request for Comments 1296, Network Information Systems Center, SRI International, Jan 1992.

[27] Maes, P., Modeling Adaptive Autonomous Agents, To appear in: Artificial Life Journal, Vol. 1, Nos. 1 & 2, MIT Press, Summer 1994.

[28] Maes, P., Kozierok, R., Learning Interface Agents, Proceedings of AAAI, 1993.

[29] Malone, T. W., Grant *et al*, Intelligent Information-Sharing Systems, *Communications of the ACM*, Vol. 30, No. 5, May 1987, pp. 390-402.

[30] Marcus, R. S., Computer and Human Understanding in Intelligent Retrieval Assistance, Proceedings of the 54th American Society for Information Science meeting, Vol. 28, October 1991, pp. 49-59.

[31] Marcus, R. S., Reintjes, J.F., A Translating Computer Interface for End-User Operation of Heterogenous Retrieval Systems; Part I: Design; Part II: Evaluations, Journal of the American Society for Information Science, 32(4), 1981, pp. 287-303, pp. 304-317.

[32] McCahill, M., The Internet Gopher: A Distributed Server Information System, *ConneXions - The Interoperability Report*, Interop, Inc., 6(7), July 1992, pp. 10-14.

[33] Metral, M., Design of a Generic Learning Interface Agent, SB Thesis, Department of Media Art and Sciences, Massachusetts Institute of Technology, May 1993.

[34] Negroponte, N., *The Architecture Machine: Towards a more Human Environment*, MIT Press, 1970.

[35] Orwant, J., Doppelgänger Goes To School: Machine Learning for User Modeling, SM Thesis, Department of Media Art and Sciences, Massachusetts Institute of Technology, Sept 1993.

[36] Rau, L. F., Conceptual Information Extraction and Retrieval from Natural Language Input, Proceedings of RIAO 88: User-Oriented Content-Based Text and Image Handling, March 1988, pp. 424-437.

[37] Rocchio, J. J., Jr., Relevance Feedback in Information Retrieval, *The Smart System — Experiments in Automatic Document Processing*, ed. Salton, G., Prentice-Hall Inc., 1971, pp. 337-354.

[38] Salton, G., McGill, M. J., *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[39] Salton, G., *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison Wesley Publishing, 1989.

[40] Salton, G., Buckley, C. Improving retrieval performance by relevance feedback. *JASIS 41*, 1990, pp. 288-297.

[41] Salton, G., Buckley, C. A note on Term Weighting and Text Matching. TR 90-1166, Department of Computer Science, Cornell University, 1990.

[42] Salton, G., ed. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison Wesley Publishing, 1989.

[43] Schneiderman, B., Direct Manipulation: A Step Beyond Programming Languages, IEEE Computer, Vol. 16(8), 1983, pp 57-59.

[44] Sheth, B., Maes, P., Evolving Agents for Personalized Information Filtering, Proceedings of the ninth IEEE Conference on AI for Applications, 1993.

[45] Suchak, M.A., GoodNews: A Collaborative Filter for Network News, SM Thesis, Department of Electrical Engineering and Computer Science, MIT, Feb 1994.

[46] Whittaker, S., Stenton, P., User Studies and the Design of Natural Language Systems, Proceedings of 4th Conference of the European Chapter of the Association of Computational Linguistics, 1989, pp. 116-123.

[47] Yang, J., Korfhage, R. R., Query Optimization in Information Retrieval Using Genetic Algorithms, Proceedings of the 5th International Conference on Genetic Algorithms, Urbana, IL, 1993, pp. 603-611.