LEARNING TO SURF: MULTIAGENT SYSTEMS FOR ADAPTIVE WEB PAGE RECOMMENDATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Marko Balabanović

March 1998

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Yoav Shoham
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Terry Winograd

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Robert Barrett

Approved for the University Committee on Graduate Studies:

Abstract

Imagine a newspaper personalized for your tastes. Instead of a selection of articles chosen for a general audience by a human editor, a software agent picks items just for you, covering your particular topics of interest. Since there are no journalists at its disposal, the agent searches the Web for appropriate articles. Over time, it uses your feedback on recommended articles to build a model of your interests. This thesis investigates the design of "recommender systems" which create such personalized newspapers.

Two research issues motivate this work and distinguish it from approaches usually taken by information retrieval or machine learning researchers. First, a recommender system will have many users, with overlapping interests. How can this be exploited? Second, each edition of a personalized newspaper consists of a small set of articles. Techniques for deciding on the relevance of individual articles are well known, but how is the composition of the set determined?

One of the primary contributions of this research is an implemented architecture linking populations of adaptive software agents. Common interests among its users are used both to increase efficiency and scalability, and to improve the quality of recommendations. A novel interface infers document preferences by monitoring user drag-and-drop actions, and affords control over the composition of sets of recommendations. Results are presented from a variety of experiments: user tests measuring learning performance, simulation studies isolating particular tradeoffs, and usability tests investigating interaction designs.

Acknowledgements

This research was conducted under the supervision of Yoav Shoham, whose intellectual rigor and insight have been a constant source of encouragement and inspiration.

Valuable perspectives and guidance have also been provided by Terry Winograd and Rob Barrett, who both induced greater clarity of thinking and presentation. Daphne Koller and Cliff Nass supplied helpful pointers to related areas of research as members of my orals committee. A thorough grounding in AI systems and architectures was provided by earlier advisors Carole Klein at Cambridge University and both Barbara Hayes-Roth and Nils Nilsson at Stanford University; in addition, Andy Hopper at Cambridge University provided the initial encouragement to undertake doctoral research.

Over the nearly six years I have been at the Department of Computer Science at Stanford University, there have been constant valuable interactions with members of the various research groups focussed on creating AI agents (or 'bots) which reason or learn (AIbots, Bots, Nobotics and Nobots), as well as the People, Computers and Design group focussed on interaction design, and, most recently, the Digital Libraries Project, working on many different technologies in support of information seeking tasks. Outside of Stanford, I have gained a wider perspective on interaction design and prototyping from Mik Lamming and Mike Flynn at the Xerox Research Center in Cambridge, and a better understanding of real-world technology design and development from James Rucker and Marcos Polanco at Imana in San Francisco.

Funding was provided by NSF grant IRI-9503109, the NSF/DARPA/NASA Digital Library project (NSF IRI-9411306), NSF grant IRI-9220645, and ARPA grant F49620-94-1-0900. The implementations to be described were constructed with the help of a lot of free software. Thanks, therefore, are also due to the authors and maintainers of Apache[1], SubArctic [Hudson and Smith, 1996], Berkeley DB [Seltzer and Yigit, 1991], mSQL [Hughes, 1993], CHACO [Hendrickson and Leland, 1994] and above all the Python programming language [van Rossum, 1995].

A number of user tests were performed as part of this work. I am grateful to all of the people who took the time to participate, either on the Web or in person here at Stanford.

[1] Currently documented at http://www.apache.org

Contents

List of Tables

List of Figures

Chapter 1

Introduction

CORFE *(n.)*

An object which is almost totally indistinguishable from a newspaper, the one crucial difference being that it belongs to somebody else and is unaccountably much more interesting that your own—which may otherwise appear to be in all respects identical. Though it is a rule of life that a train or other public place may contain any number of corfes but only one newspaper, it is quite possible to transform your own perfectly ordinary newspaper into a corfe by the simple expedient of letting somebody else read it.

from [Adams and Lloyd, 1984]

Imagine a newspaper that is personalized for your tastes. Instead of a selection of articles and features chosen for a general audience by a human editor, a software agent picks items just for you, covering your particular topics of interest. You would be right in thinking that today's technology limits an electronic personalized newspaper. It cannot compete with its human-edited equivalent when it comes to completeness of coverage for a large audience. However there are information needs that a more personalized, automated newspaper can successfully meet. This thesis is about how to design such personalized newspapers.

Consider some of the problems in creating a *text recommender system*, a software system that recommends documents for readers. First, it must discover potentially relevant documents. Unlike a major print newspaper, our electronic version has no cadres of journalists at its disposal. Instead it is restricted to selecting among documents available on public networks, and in particular the World-Wide Web. Luckily the exponential increase in size of the Web over the last few years has made available a wealth of material, including, for instance, up-to-the minute news articles on a wide variety of topics. Nevertheless it is a resource-intensive task to track down documents that might be of interest to a particular readership, and this *collection* phase is one focus of the research reported here.

But take a step back: how does a computer system even know which documents might be of interest to its readers? Just as a friend would learn of your likes and dislikes from your comments and behavior, a computer system can gather feedback concerning previously recommended documents. This can take the form of *explicit feedback*, where readers indicate, for instance, their degree of satisfaction with a particular document on some kind of scale. Or it could be *implicit feedback*, where inferences are made from observations of a reader's actions, for instance in using software to browse through or read recommended documents. What is a computer to do with such feedback? Perhaps you indicate that you particularly enjoyed today's article about the new Bordeaux vintage. Should tomorrow's personalized newspaper be completely concentrated on Bordeaux and wines? Is there a place for an article about a new strain of flu, when the computer knows nothing of your interests regarding health-related stories? What about a feature on Tuscany which has proven to be very popular with others who enjoyed the Bordeaux article? These questions concern the composition of the *recommendation set*, the collection of documents delivered in a single iteration, a single edition of the personalized newspaper. Methods of gathering reader feedback, and resulting tradeoffs and choices when composing the recommendation set are both important areas of investigation undertaken in this research.

Readers with knowledge of the academic fields of information retrieval, collaborative filtering, user modeling and machine learning will have recognized many of the issues raised so far. In the remainder of this introduction, the *text recommendation task* will be defined more precisely and with respect to other more commonly studied tasks such as retrieval, routing, filtering, and classification. Following this definition, the different research issues already alluded to will be revisited and explained at a more detailed level. What should be expected from the rest of this thesis? As befits the practical nature of this task, a series of working prototypes have been implemented and tested, both with real users and in simulation. By incrementally defining these increasingly complex systems, the reader will gradually be introduced to the original contributions of the research. These range from methods for the collection phase (discovering documents of interest to a community of users) to designs for the interactions between a recommender system and a user. A summary of these contributions will form the final part of this introduction.

1.1 The recommendation task

As is common for applications whose domain is the Web, this thesis draws heavily on the traditions of the information retrieval (IR) community. However the task outlined diverges from the original goal of IR systems—namely, retrieving documents to fulfill a short-term, specific information need as expressed in some query. In contrast, the goal of what have recently come to be known as *recommender systems* [Resnick and Varian, 1997] is to learn about a population of users in order to provide increasingly appropriate recommendations. When considering the specific case of a recommender system whose domain is documents, we define the *text recommendation task*, following [Jennings and Higuchi, 1993], as *helping the user follow threads of interest, by regularly* *recommending small sets of documents* (typically *small* means between 6–10). A text recommender system interacts with a user according to the following loop:

1. Select a set of documents to present to the user, based on an induced user model;
2. Elicit feedback from the user;
3. Improve the user model given the feedback on the presented documents.

The automatic layout of personalized newspapers is an ongoing area of research (e.g., see [Bender *et al.*, 1991]). This thesis, however, concentrates on the selection of articles for readers, rather than their final presentation and appearance as an electronic newspaper.

Several different IR tasks are defined by the Text REtrieval (TREC) conferences; it will prove useful to appeal to these specific definitions in order to ground a discussion of the differences between them and the recommendation task. In particular, TREC-5 [Harman, 1996], the latest of this series of conferences at time of writing, will be assumed.

The *ad hoc retrieval* task (sometimes called *retrospective* retrieval) requires the user to formulate a query for an IR system that accesses a particular corpus of documents. In response, the system ranks the documents according to relevance to the query. The query represents a short-term information need, which will hopefully be satisfied by the top-ranked documents. In contrast, the recommendation task does not require a query to be formulated. Furthermore, discrete, unordered sets of documents are delivered, rather than a ranking over an entire corpus. Finally, the iterative nature of the recommendation task is better suited to long-term, ongoing information needs. Ad hoc retrieval is the original task for IR systems, and is still thought of as canonical. It will be seen, however, that an IR "engine," a collection of algorithms and data structures developed originally for ad hoc retrieval, proves useful for a variety of other tasks.

More similar to the recommendation task are the *routing* and *filtering* tasks, where users are assumed to have long-term information needs [Belkin and Croft, 1992]. In the routing task, the system is not provided with a query, but rather a set of sample documents that are relevant to a user's information need. Using these documents as training data, the system can then rank the remaining documents in the corpus. For the filtering task, the system is instead required to classify the remaining documents into categories of "relevant" or "irrelevant". Variations of this task are often studied from a machine learning perspective, as instances of supervised classification problems [Yang, 1997]—from a training set of sample documents with attached category labels, a system learns to assign category labels to new documents (for the filtering task, the labels would be just "relevant" and "irrelevant").

In contrast, a recommender system must select its own samples of training data to present to the user for feedback, thus engaging in *active learning*. Furthermore, the fact that the process proceeds in an incremental rather than batch fashion means that many of the techniques used in routing or filtering, which rely on batch processing of the training data, are no longer as appropriate. The difference between these batch tasks and the recommendation task is analogous to that between supervised classification and reinforcement learning.

A more recent strand of research is *collaborative filtering* [Malone *et al.*, 1987]. A collaborative system will recommend an item based on the judgments of other users, rather than on the content of the item itself. This is consistent with the text recommendation task, and indeed Chapter 5 will present a way to combine collaborative filtering with IR techniques.

Finally, assisted browsing systems [Armstrong *et al.*, 1995; Lieberman, 1995] propose an alternative task formulation which simplifies the learning problem. By restricting themselves to the section of the Web just ahead of the user's current browser location, they recommend appropriate links to follow rather than arbitrary Web pages or documents.

1.2 Research issues

Let us return to the example of the software agent creating a personalized newspaper for a reader, where so far all that is known is that the reader enjoyed an article about a new Bordeaux vintage. This context exposes the two main issues to be investigated by this research.

1.2.1 How to exploit overlaps between users' interests

The user who enjoyed the Bordeaux article is merely one of many users of a recommender system. Perhaps there are a number of people who are interested in articles about wine. One challenge, therefore, is to find ways to allocate resources such that duplicated searching and discovery activities are eliminated—there could be a single "wine expert," perhaps, serving this community. But given the dynamic and adaptive nature of a recommender system, it is desirable that such experts are a spontaneous result of the system evolving; it would be impractical to allocate or plan such decompositions in advance.

Correspondingly, it might transpire that our Bordeaux lover is also an avid reader of articles about wildlife. Therefore it would be necessary for some of her articles to originate from a wildlife expert, if one exists. In general any user could have multiple topics of interest, arbitrarily overlapping with other users. An important research issue, then, is the design of systems that can take advantage of this structure to perform dynamic load balancing, and so deliver efficiencies of scale as the number of users increases.

In addition, it is interesting to investigate further consequences of pooling feedback and discovering clusters of interest. The concept of collaborative filtering has already been mentioned: a collaborative strategy could, for instance, discover other users who had also enjoyed the Bordeaux article, and then recommend other articles which they had liked. In contrast, a *content-based* strategy would recommend articles similar in content to the one about Bordeaux. By considering other users' opinions of an article as well as its content, a hybrid approach can make more informed decisions. Therefore, another aspect of this research issue is how overlaps between users' interests can be exploited to improve recommendations, combining the strengths of both collaborative and content-based techniques.

1.2.2 How to decide upon the composition of recommendation sets

Techniques for choosing individual articles for recommendation are well known, but deciding on the composition of a set of recommendations (or the selection of articles in a single edition of a personalized newspaper) is an important issue that has received little attention. A number of tradeoffs are involved. For instance, as in the earlier example: the system comes across an article about a new strain of flu. It has no information about whether or not the user would like this article, since it has never received feedback about any similar article. An *exploratory* strategy would recommend this article. In contrast, an *exploitative* strategy would stick to articles similar to the known Bordeaux example. This is known as the exploration/exploitation tradeoff [Berry and Fristedt, 1985].

Given the variety of articles selected for a user (both content-based and collaborative recommendations, both exploratory and exploitative choices, selections pertinent to different topics of interest), a significant problem is to winnow down and present a set of recommendations in an interface.

The issue here, in a nutshell, is to investigate ways of representing these tradeoffs, and to design interfaces that both leave the user in control of the composition of the recommendation set, and at the same time gather sufficient feedback to enable the system to learn a better user model.

1.3 Summary of contributions

The research issues described have been addressed through the design and implementation of a series of prototypes, which have been tested in a variety of experiments. The resulting original research contributions are summarized in the following four points.

An architecture linking populations of adaptive software agents, which takes advantage of the overlaps of interests among users of a recommender system to increase efficiency and scalability

This architecture dynamically links evolving and adapting populations of agents to a changing user community. It allows for scaling up the number of users while maintaining a fixed level of resource usage, providing a graceful degradation in the quality of recommendations. This is achieved by encouraging the evolution of agents to serve clusters of user interest, and pooling the feedback relating to such clusters. Its distributed nature ensures robustness and allows for users and agents to be added or removed dynamically.

A novel recommendation mechanism combining a content-based and a collaborative method

A method is introduced which combines both content-based and collaborative techniques, such that many of the disadvantages of using either approach alone are cancelled out. This method is applied to the text recommendation task in the context of the architecture described above. It can also be considered as a way for collaborative systems to be applied to previously problematic dynamic domains, such as recommendation of Web pages or news articles.

A new affordance allowing users to control the breadth or narrowness of their set of recommendations

A large-scale simulated setting was constructed primarily to investigate particular implementations for exploratory and exploitative strategies; the results of the experiments lead to a fuller understanding of how such strategies can be implemented and how they impact speed of learning user models, responsiveness to change in user interests and the composition of recommendation sets. The result of this research is an implemented user interface component allowing users to select "broad" or "narrow" recommendations, corresponding to exploratory or exploitative strategies, respectively.

A new interaction design allowing users to vary the proportions between topics of interest, where only implicit feedback is required

An implemented interface, based on a novel component composed of sliding panels, allows users to control the proportions between multiple topics of interest, thus providing a finer-grained control over the composition of recommendation sets. The interface further enables learning of user models through implicit gathering of feedback, by monitoring and interpreting user drag-and-drop actions (thus imposing much less cognitive load upon users).

1.4 Overview of the remainder of this thesis

Chapter 2 defines concepts and introduces notation which will be useful in the remainder of the thesis. In doing so it also discusses a number of assumptions made, thus situating this research relative to various academic communities, debates and schools of thought.

Chapter 3 specifies a very simple content-based recommender system. This serves to introduce the specific search and machine learning algorithms employed, and to highlight problems which are solved by later, more sophisticated systems. It corresponds to an early working prototype called "LIRA" (Learning Information Retrieval Agent), and results are shown from a user study conducted with it. The design of LIRA and its testing were originally published as [Balabanović and Shoham, 1995] and more fully as [Balabanović *et al.*, 1997]

In Chapter 4, alternative schemes based on collaborative filtering are introduced. This is a short chapter as a purely collaborative prototype was not constructed as part of this research.

A multiagent architecture as referred to in section 1.3 is introduced in Chapter 5, as implemented by the "Fab" prototype. This is the most complete of the implementations discussed, and was available for public use on the Web for several months, as well as for more controlled user tests. The first two contributions listed in section 1.3 are presented in this chapter; this work was originally published as [Balabanović, 1997] and [Balabanović and Shoham, 1997].

Chapter 6 goes on to discuss problems related to selecting a recommendation set and presenting it to the user. Accordingly, section 6.1 concerns an implementation and simulated experiments to investigate the properties of the exploration/exploitation tradeoff (to be published as [Balabanović, 1998a]), and section 6.2 concerns the design of a user interface to gather implicit feedback and give the user control of recommendations relating to multiple topics of interest (also described in [Balabanović, 1998b; Balabanović, 1998c]).

Finally, Chapter 7 wraps up with conclusions and revisits the contributions above. Related work is discussed throughout rather than in a separate chapter.

Chapter 2

Background and Definitions

The purpose of this chapter is twofold:

1. To introduce basic concepts, notation, assumptions and definitions which lay a foundation for the system designs and experiments to follow;

2. To situate the research relative to various academic communities and debates, and in particular to carve out its niche with respect to information retrieval, machine learning and user modeling.

These two goals will be tackled in the context of four basic desiderata for any text recommender system: how are documents represented, how are users represented, how is the system evaluated, and what are the possible human-computer interactions.

The reader is assumed to have a basic understanding of the protocols and formats forming the World-Wide Web [Berners-Lee *et al.*, 1992], such as HTTP and HTML. For more information, see reports and specifications as published by the World-Wide Web Consortium ("W3C")[1].

[1]Currently at http://www.w3.org

2.1 Representing text

This thesis is concerned with the recommendation of documents. An important prerequisite for this task is the ability to represent documents in a computer system. This enables algorithms which decide upon recipients for the documents, and also provides the basis for learning users' interests in documents. Investigation of ways to represent documents is a long-standing endeavor of the IR community. In this section the commonly used *vector space* method will be explained, along with the particular formulation used for all of the systems to be described.

2.1.1 Vector space model

There are many features of a document that could be useful to capture in a computer representation. For instance: visual appeal, intended audience, complexity of writing, quality of writing, genre or style, and, of course, the subject matter. The vector space model attempts to represent only the subject matter of a document, and it does so in a statistical rather than knowledge-based fashion. There are important repercussions resulting from these assumptions, as will be seen in section 2.2 when the representation of user interests is discussed.

The basis for the vector space approach, and indeed for most work within IR, is to consider a document as merely an unordered list of its constituent words and their frequencies of occurrence. However not all words are equally useful as indicators of document content, and so it becomes useful to introduce a weighting scheme. The method followed by all of the systems described in this thesis is summarized here, and the reader is referred to [Salton and McGill, 1983] for more general overview of the vector space approach.

Let $D = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n\}$ denote a set of documents or a *corpus*. Let $\{t_1, t_2, \ldots, t_p\}$

be the "dictionary," a set of all of the words of the corpus. We represent each document $\mathbf{d}$ as a vector in a p-dimensional vector space, so $\mathbf{d} = [d_{t_1} d_{t_2} \ldots d_{t_p}]$ where d_{t_i} is the weight for word t_i in document $\mathbf{d}$.

How are such vectors derived? The first step is to extract the constituent words from a document. If the document is a Web page, then all HTML tags, comments, images or other multimedia objects are ignored. Secondly, very common English words from a standard stop list of 571 words are ignored (some of the parsing methods employed are English language-specific, given the English-speaking target audience). Examples of stop words are `and`, `but`, `the`, etc. When dealing with Web pages, additional words very common on the Web are ignored, such as `page`, `web` and `click`.

Extracted words are reduced to their *stems* using the Porter suffix-stripping algorithm [Porter, 1980] (as implemented in [Frakes, 1992]). This reduces redundancy—for instance, `computer`, `computers`, `computing` and `computability` all reduce to `comput`.

Words are then weighted using a "TFIDF" scheme: the weight d_{t_i} of a word t_i in a document $\mathbf{d}$ is derived by multiplying a term frequency ("TF") component by an inverse document frequency ("IDF") component:

$$d_{t_i} = \left(0.5 + 0.5\frac{tf_i}{tf_{max}}\right)\left(\log\frac{n}{df_i}\right) \tag{2.1}$$

where tf_i is the number of times word t_i appears in document $\mathbf{d}$ (the *term frequency*), df_i is the number of documents in the corpus which contain t_i (the *document frequency*), n is the number of documents in the corpus and tf_{max} is the maximum term frequency over all words in $\mathbf{d}$.

If word t_i does not occur in document $\mathbf{d}$, or if it is not among the l highest-weighted words, then $d_{t_i} = \oslash$, a special "missing" value. Note that $\oslash$ is treated as 0 for all vector calculations. The value of l has varied between 10 and 100 for the systems described in this thesis. Using all of the words from a document tends to lead to the "over-fitting" problem common in machine learning, and increases memory and communications load. Recent experiments described in [Pazzani *et al.*, 1996] have shown that using too many words leads to a decrease in performance when classifying Web pages using supervised learning methods, and similar results will be shown in section 3.4.3.

In a final step, the vector for each document is normalized to be of unit length. Variants of this formula are commonly used by IR systems—the aim is to give high weights to words which occur frequently in the document in question, but are rare in the rest of the corpus. For instance, the five top-weighted words from the IRS "Forms and Publications" Web page (as of March 1997) are `faint-of-heart` (0.33), `tax` (0.28), `regulations` (0.25), `tax-payer` (0.23) and `commissioner` (0.22).

A useful capability is the calculation of inter-document similarity. A standard metric for this in IR systems is to take the cosine of the angle between the two vectors. Since the vectors as defined above are of unit length, it is equivalent to use the dot product:

$$\textsc{sim}(\mathbf{d}_1, \mathbf{d}_2) = \mathbf{d}_1.\mathbf{d}_2 \tag{2.2}$$

Note that any weights whose value is $\oslash$ are treated as if the value was 0 in this calculation. On occasion it is more efficient to store a short *document signature*, such that if two documents have identical signatures, then it is likely that they have identical content. A document signature $\hat{\mathbf{d}}$ is defined to be the 10 highest-weighted words from the document vector $\mathbf{d}$ along with their weights. In practice even small changes to document content will lead to a change in the signature.

Although this vector space model may seem simplistic, it has been shown to be competitive with alternative IR methods [Harman, 1994]. Furthermore, it has been used and studied extensively, and forms the basis for many commercial Web search systems.

2.1.2 Special considerations when gathering Web pages

One of the difficulties of applying the scheme above to a large and dynamic corpus such as the Web is that it is impossible to accurately measure the document frequencies. Instead, these are estimated using a sample of randomly picked Web pages. Most of the systems to be described worked with a document frequency dictionary of approximately 70000 words taken from 5229 randomly picked Web pages (using the WebCrawler service to supply random URLs[2]). So in equation 2.1, $n = 5229$. When encountering a word t_i which is not represented in the document frequency dictionary, $df_i = 1$ is assumed. Note that, for consistency, this form of sampled document frequency dictionary is used throughout, even when a simulated setting would allow calculation of real document frequencies.

It is worthwhile noting a number of further difficulties specific to Web pages, as these are problems not commonly discussed in the IR literature (which more often assumes the more uniform domain of newswire or journal articles):

- When processing a large number of Web pages, it is highly likely that syntactically incorrect HTML or overly long pages will be encountered. Great care must be taken that the algorithms are robust when faced with unexpected forms or sizes of data. Implementations of the scheme above must restrict the maximum size of a page, and of any metadata fields monitored (e.g., the title), and further enforce timeouts on any connections.

- It is common to find the same document in several locations, perhaps with changes in formatting, or even an advertisement which is periodically refreshed.

[2]Currently at http://www.webcrawler.com

Comparing document signatures or vectors gives a more robust identity check than comparing URLs.

- In contrast, dynamic pages (the result of scripts or database queries) or those updated frequently are also common. The vector space scheme captures a static snapshot of a document, which may change by the time the user sees a recommendation. One solution is to also cache a copy of the document content, in order to be able to show the user the same document originally found by the agent. However, this has the disadvantage that the user will potentially be shown out-of-date information, and furthermore requires significant storage space.

- As a world-wide resource, the Web contains text in many different languages and character sets. However, the stop word list and stemming algorithm chosen are optimized for English, and furthermore the majority of test users of our systems have been English speaking. Rather than implement a language spotting algorithm (e.g., [Grefenstette, 1995]), we simply refrain from following links to domains of non-English speaking countries (e.g., `.fr` for France). This drastically reduces the number of recommended non-English documents.

- Over time, the proportion of easily-parseable ASCII text on Web pages is decreasing, with the advent of tools or formats which allow more flexible layout and design (Java, Shockwave, Acrobat PDF, etc.). In addition, professionally designed Web sites often favor text encoded in bit images, for full control over the appearance.

Even apart from the concerns above, there are a number of difficulties caused by social issues underlying Web usage:

- The phenomenon of "spamming" reduces the probability that words in HTML are indicative of document content. In an effort to generate publicity, Web page authors "spam" well-known Web indexes by adding to their pages words that are common in user queries. In this way they hope that their pages will more often be returned as relevant to user queries, even if in reality they are not.

- Finally, the existing voluntary standard for preventing computer programs from accessing and indexing a site [Koster, 1994] is under-specified. "Robots" (sometimes called "spiders") are discouraged from visiting sites or parts of sites by posted `robots.txt` files. However, there is no way in such a file to specify a maximum allowable frequency of page requests, or the length of time for which the file itself can be cached. Thus, when fetching a large number of Web pages, it is highly likely that a number of irate Web-masters will be encountered.

2.1.3 Alternative features

The vector space scheme described has been used for all of the research to be described. However, alternative sets of features and formalisms have been studied—they are briefly surveyed here.

- Attempts have been made to ascribe interpretations to different HTML markup tags, in an effort to extract words more indicative of the content of a Web page. So, for instance, in HTML `<B>this</B>` would appear in bold face, and could be assumed to be more indicative of the document content. In addition, titles, headings and link anchor texts have been assigned higher weightings by various researchers.

- Given a large corpus of documents ahead of time, a matrix singular-value decomposition can be used to automatically extract the most important dimensions, each expressed as a vector [Deerwester *et al.*, 1990]. This *latent semantic* *indexing* (LSI) has been shown to have a number of advantages: by reducing dimensionality, a wider range of machine learning techniques are applicable; additionally, frequently cooccurring words tend to end up clustered in a single dimension, thus reducing the problems caused by synonyms.

- The graph structure of the Web is a source of information whose use also requires a preliminary analysis phase. For example, the "citation importance" [Page and Brin, 1998] of a page is a global feature computed not from its content but from its location in the graph. Pages have a high citation importance if many other pages link to them, and even more so if those pages also have a high citation importance.

- Finally, there is a large ongoing research effort addressing the distribution and use of metadata, either explicitly specified by the page author or a third party, or inferred automatically from properties of the page. An example of the latter case is to assign classification labels to pages based on supervised machine learning methods. These labels then become features for the page. The former case, explicitly assigned metadata, is largely dependent on agreed-upon protocols and infrastructure. The PICS system [Resnick and Miller, 1996] and the Stanford Digital Library metadata architecture [Baldonado *et al.*, 1997] are efforts along these lines, although no system has seen widespread adoption on the Web as yet.

In addition, within the field of IR, the *probabilistic model* forms an important alternative to the vector space approach [Salton and McGill, 1983; Belkin and Croft, 1987]. A variety of formulations exist. The basic idea is to estimate the probability that a document is relevant to a query, given probabilities over individual words (specifically, the probability that a word occurs in a document given that the document is relevant, or given that it is irrelevant). Independence of the individual word probabilities is usually assumed. In practice, comparisons show little difference between state-of-the-art ad hoc retrieval systems using vector space and probabilistic approaches [Harman, 1996].

2.2 Representing users

This section concerns two kinds of user models: those learned by the recommender system in order to recommend more appropriate documents, and those used by an optional simulation system to simulate user responses. In both cases the models represent the users' documents of interest, and do not capture all of the information of more richly structured formalisms sometimes employed by user modeling systems. For instance, stereotypes [Rich, 1979; Brajnik *et al.*, 1987] are frame-based user models that have been applied to similar tasks, and which might additionally encode users' age, education level, religious background or other demographic factors.

To avoid confusion, the phrase *user profile* will be used to denote the user model learned by the recommender system, whereas *user model* will be reserved for referring to the user model used by the simulation system. Both kinds of model, according to the dimensions defined by Rich [1979], are individual user, long-term and implicitly acquired. According to the classification of Carbonell [1983], they are analytical cognitive models.

2.2.1 Preference rankings: Definition and notation

Preference rankings are a fundamental concept underlying both the user representation and the evaluation metrics to be described. A preference ranking $\succ$ is a binary relation on the set of documents D:

$$\text{for } \mathbf{d}, \mathbf{d}' \in D,\ \ \mathbf{d} \succ \mathbf{d}' \Leftrightarrow \mathbf{d} \text{ is preferred to } \mathbf{d}' \tag{2.3}$$

which can also be written:

$$\succ \; = \; \{(\mathbf{d}, \mathbf{d}') \mid \mathbf{d} \text{ is preferred to } \mathbf{d}'\} \tag{2.4}$$

It is possible for two documents to be tied with respect to a preference ranking. We say a ranking $\succ$ is *indifferent* about $\mathbf{d}$ and $\mathbf{d}'$ (written $\mathbf{d} \sim \mathbf{d}'$) if neither $\mathbf{d} \succ \mathbf{d}'$ nor $\mathbf{d}' \succ \mathbf{d}$ holds. This can be interpreted as meaning that both documents are equally preferable, or simply that they are not comparable.

Preference rankings are further restricted to be *weak orders*, in other words they satisfy the following two axioms, for all $\mathbf{d}, \mathbf{d}', \mathbf{d}''$ in D:

$$\text{Asymmetry} : \mathbf{d} \succ \mathbf{d}' \Rightarrow \neg(\mathbf{d}' \succ \mathbf{d}) \tag{2.5}$$

$$\text{Negative Transitivity} : \neg(\mathbf{d} \succ \mathbf{d}') \wedge \neg(\mathbf{d}' \succ \mathbf{d}'') \Rightarrow \neg(\mathbf{d} \succ \mathbf{d}'') \tag{2.6}$$

Note that these axioms imply the positive transitivity of $\succ$, and also that $\sim$ is an equivalence relation [Yao, 1995]. A weak order can be visualized as a number of levels into which documents are arranged—documents in a higher level are preferred to documents in a lower level, and documents at the same level are indifferent.

This formulation of preference rankings was first described in the context of IR by Bollman and Wong [1987], and has since been studied both from the standpoint of machine learning [Wong *et al.*, 1988; Wong and Yao, 1990; Wong *et al.*, 1991] and evaluation of IR systems [Yao, 1995].

2.2.2 Information need and relevance

The concept of a user's *information needs* is the basis for their representation within the systems to be described. Taylor [1962], based on interviews with academic librarians, provides a useful breakdown of information need, positing that it progresses through four stages of "question formation": visceral, conscious, formalized and compromised (the latter stage corresponds to a query constructed to submit to a retrieval system). In seeking a more direct correspondence to different kinds of support an information system might be required to supply, Ingwersen [1992] suggests a different breakdown: verificative or location information needs (the user wants to verify or locate known items), conscious topical needs (the user wants to clarify, review or pursue aspects of known subject matter) and muddled topical needs (the user wants to explore new concepts outside of known subject matter).

The centrality of the concept of information need to IR is consistent with the *uses and gratification* behavioral theory of news reading [Dozier and Rice, 1984], which is based on the assumption that "media use, including news reading, serves some ulterior purpose external to the communication behavior itself." An alternative is the *play* or *ludenic* theory [Stephenson, 1967], which explains news reading in terms of its intrinsic pleasurableness as an activity. Given the specific, ongoing information needs implied by the task of following threads of interest, the systems to be described fit better within the realm of the uses and gratification theory. However, it is worthwhile to realize that reasons for usage vary over time and between users, and it is important to also support "recreational" reading in the interface [Watters *et al.*, 1998].

The *topical relevance* of a document is defined as its relevance to a formalized information need (which is also a conscious topical need, to use Ingwersen's classification), as determined by a human judge. This is the definition of relevance used for evaluating systems at the TREC conferences [Harman, 1996], and in general corresponds to the regular usage of the word *relevance* in the IR community[3]. An example of an intensional definition of a formalized information need from TREC-4 is "What are the prospects of the Quebec separatists achieving independence from the rest of Canada?" Unlike utility, which is a relation between a user and a document, topical relevance is a relation between a "topic" and a document.

Classifications and hierarchies based on clusters of topical relevance are prevalent in information systems. *Editorial categories* are defined as clusters of topical relevance to which human editors can assign documents according to their content, and which have been chosen by human editors so as to be useful for particular audiences. Commonly found examples are categories the news services assign to their news stories (e.g., the Reuters service assigns labels such as "gold," "cocoa" and "money supply"), or newspaper sections like "politics," "sports," "financial news," etc. An example of an editorial category analogous to the aforementioned formalized information need is "Canadian Current Events: National Unity," a category used for classifying news articles by the Yahoo! Web service[4]. One way to think about editorial categories is as extensional definitions of long-term information needs, as defined by editors rather than end-users.

In the remainder of this thesis the word *topic* will be used to mean a cluster of topical relevance, as defined either by an editor or an end-user. This is almost the same as its usage at the TREC conferences, where *topic* is synonymous with *formalized information need.* A topic is denoted by T_i, where $T_i \subset D$.

[3]Since there is an ongoing debate about the nature of relevance, for example see [Saracevic, 1975; Saracevic, 1995], *topical relevance* is used as a more precise phrase, defined as above.

[4]The Yahoo! hierarchy is currently available at http://www.yahoo.com.

2.2.3 Users' judgments of documents

Why do people decide to read particular news articles and not others? How do they decide which documents best satisfy their information needs? In this section we examine questions such as these to gain an overview of the space of possibilities. The following section will describe the particular representation for user interests chosen for this research, one based on topical relevance.

Researchers from both user modeling and IR have performed experiments with human subjects to investigate the questions above. They have uncovered a wide variety of reasons cited by users for their preferences among documents. Many of the pertinent studies agree, however, that topical relevance is a crucial factor influencing user judgments. To give a flavor for the results: In a study where people were asked for criteria they used in evaluating the value of a document in relation to a specific information need, 82% of them cited topical relevance as a primary reason [Schamber and Bateman, 1996]. In another study, analysis of think-aloud protocols from an information-seeking task revealed topical relevance to be the major factor [Wang and Soergel, 1996]. In a study of why people chose to read particular Usenet news articles, topical relevance was shown to play a part in almost all of the decisions [Stadnyk and Kass, 1991].

All of the studies cited above also give *novelty* as an important factor, often second only to topical relevance. Novelty here is loosely defined as the existence of information in a document which is new to a user. A variety of further criteria have been identified which can be thought of as inherent in the content of the document. These include the "type" of the document (e.g., "request for information," "advertisement," "discussion," etc.), the source of the document, the quality of writing, the sensationalism of the document. In addition, a large number of factors dependent upon the user's context have been identified in many studies, and users will also bring their own set of criteria to bear. Commonly cited factors include: knowledge of domain, time constraints, emotional appeal, societal importance, media cueing, credibility, popularity. Many of these context-specific factors would be difficult or impossible to model given current methods.

It is useful to distinguish *instance-based* factors, which can be derived by analyzing single documents independently, from *set-based* factors, which may require analysis of a set of previous recommendations or retrieval results. For instance, topical relevance is an instance-based factor, whereas novelty is a set-based factor, since a document can be judged to be less novel if the preceding document is very similar.

The *probability ranking principle* [Robertson, 1977] states that the optimal output for an IR system is a list of documents ranked in order of estimated probability of relevance to the supplied query. Since most IR systems adhere to this principle and evaluate each document independently, they are ill-equipped to model set-based factors like novelty. In practice, many systems (e.g., Web search engines) take the first steps towards addressing this factor by removing duplicates from a results list. Often there is a tradeoff between topical relevance and novelty—the most topically relevant documents for some information need would usually also be very similar to each other.

In contrast to an IR system, a newspaper must take great account of novelty as a factor—readers would be puzzled by the existence of two articles covering the same subject matter. In addition, a newspaper enters into an implicit "social contract" regarding completeness of coverage—readers assume omitted stories are less important, relative to the genre or specialization of the newspaper, than those printed. This is consistent with findings showing that perceived recall is highly correlated with user satisfaction for ad hoc retrieval systems [Su, 1992][5].

[5] *Perceived recall* corresponds to Su's *user's satisfaction with completeness of search results.*

2.2.4 User profiles

A variety of factors relating to the representation of users have been discussed. The studies cited above show that topical relevance is a vital factor. Recall that the text recommendation task has been defined as helping the user to follow threads of interest. Broadly speaking, this can be a blend of two kinds of help: automatically selecting articles for users, and providing facilities to help users select articles for themselves. The choice of representation for users determines the boundary between these two kinds of goal.

Rather than explicitly modeling users, the systems described in this thesis instead model just the users' documents of interest. The simplest way to achieve this is to represent user profiles in the same vector space as documents. Topical relevance is the only factor captured by this representation, although in a limited way it is possible to also measure novelty—a document can be deemed similar or dissimilar, relative to the vector space, to other documents a user has been recommended.

The consequence of this decision is that articles are selected for users almost entirely on a topical relevance basis. Users can then apply their own idiosyncratic criteria by browsing, selecting and interacting with the recommender system. Thus the decision to model only topical relevance sets the boundary between the system's article selections and the user's article selections. In general the location of this boundary is a key difference between recommender or information filtering systems (where the system makes many choices on the user's behalf), and ad hoc retrieval or information visualization systems (where the users are provided tools but must make their own choices).

User profiles are denoted $\mathbf{m} = [m_{t_1} m_{t_2} \dots m_{t_p}]$.

This single-vector representation may at first appear similar to the standard way to represent a query for an ad hoc retrieval system. However, the underlying assumptions are quite different, due to the fact that a query is specified by a user directly, whereas a user profile is learned by the system given feedback on recommended documents. For a query `Bordeaux wine`, it is correct to assume that the user is not interested in documents about the flu, using the implicit closed-world assumption of IR systems. However, in a user profile for a recommender system, missing words are merely words about which there has been neither positive nor negative feedback yet. Thus, in this example, there is no information about the user's preferences for documents about the flu. As for document vectors, elements m_{t_i} are permitted to have a "missing" value, denoted $\oslash$.

Given such a vector $\mathbf{m}$, the topical relevance of a document $\mathbf{d}$ is measured by taking a dot product, after appropriately normalizing each vector to be of unit length:

$$\text{Topical relevance of } \mathbf{d} \text{ to } \mathbf{m} = \textsc{sim}(\mathbf{m}, \mathbf{d}) \tag{2.7}$$

We say a preference relation $\succ$ is *weakly linear* [Wong and Yao, 1990] if there exists a user profile $\mathbf{m}$ such that for any $\mathbf{d}, \mathbf{d}'$ in D,

$$\mathbf{d} \succ \mathbf{d}' \;\Rightarrow\; \textsc{sim}(\mathbf{m}, \mathbf{d}) > \textsc{sim}(\mathbf{m}, \mathbf{d}') \tag{2.8}$$

One of the disadvantages of such a single-vector representation is the confusion caused when trying to represent multiple topics of interest. For instance, it is impossible to represent a strict EXCLUSIVE-OR relation—if a user is interested in both hiking (trekking) and astronomy (stars), they might be recommended unwanted documents about the *Star Trek* television show. In practice, however, the vocabulary is usually rich enough to ameliorate such effects. For instance, the weight of further words such as `trail`, `mountain`, `starship` or `telescope` would help disambiguate the vector.

Nevertheless, as more and more topics of interest are represented within the same vector, the potential for confusion increases (later experiments will show the corresponding decrease in learning rate). Apart from the difficulty of specifying the relative interestingness of each topic, it is hard to know whether a document which matches one topic strongly should be ranked higher or lower than one which matches many topics weakly.

An alternative scheme is to allow many such profiles per user, one per topic of interest. A *multiple-vector profile* is written $U = \{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$. This prevents some of the problems described, but introduces an additional level of complexity in the user interface, the learning algorithm and the document selection strategy. Except where specified, a *single-vector profile* $\mathbf{m}$ is assumed for each user.

2.2.5 Simulating users' document ranking behavior

So far methods have been discussed for representing user profiles, whose purpose is to enable the recommendation of appropriate documents. Another kind of user model serves to simulate user responses to documents. These responses are an essential component of a simulation environment used for some of the experiments to be described.

Rather than attempting to simulate the full gamut of human information seeking behavior, the user models will attempt to capture solely topical relevance, which as already explained has been shown to be one of the most important factors influencing users' judgments.

When running simulations, a corpus is used which is a union of disjoint editorial categories (the externally supplied category labels are used to stand in for real user's topics of interest). For instance, a corpus may contain articles pertaining to wines, health and art, each article belonging to just one of the three topics. Users are assumed to have a preference ranking among the set of topics present in the corpus. If one topic is preferred to another, then all documents from the first will be preferred to all documents from the second (in accordance with the decision to consider only topical relevance). This may seem at first to be unrealistically strict. However, as will be seen, the corpora used for simulations contain topics which are relatively broad and messy; in addition, the constituent documents are often irregular, contain various multimedia or interactive components, and so are hard to represent in the vector space notation. Although the strict nature of the preference appears to make the learning task easier, in reality the unfocused nature of the topics balances out this effect.

In the experiments users are modeled by a particular class of preference structures. Given two topics T_1, T_2, write $T_1 \succ T_2$ as a shorthand for $\mathbf{d} \succ \mathbf{d}'$ for all $\mathbf{d} \in T_1, \mathbf{d}' \in T_2$. Define an ***n*-pref user** as having a preference relation $\succ$ with the following form for topics $T_1, T_2, \ldots, T_m$:

$$T_1 \succ T_2 \succ \ldots \succ T_n \succ \{T_{n+1}, T_{n+2}, \ldots, T_m\} \tag{2.9}$$

Thus the preference relation for an n-pref user has $n + 1$ equivalence classes; a 1-pref user has a single topic which is preferred to all others, a 2-pref user has two topics they prefer to all others, and in addition has a preference among those two, and so on.

This n-pref construction relies on four key assumptions:

1. Topical relevance is the only factor influencing users' judgments of documents that is considered;
2. Topical relevance is expressed in terms of editorial categories, or topics;
3. A user's interests are represented by a preference ranking among these topics;
4. Such a preference ranking is of the form of an n-pref structure, as defined above.

The first assumption has already been discussed in section 2.2.3 and above, but since preference rankings over editorial categories are not a user model representation commonly seen in the literature, it is worthwhile to examine the reasoning behind each of the remaining three assumptions in turn.

The choice of editorial categories as indicators of user interest is partially a pragmatic one—a corpus can automatically be gathered, and simulated user interests can be generated without human involvement. Nevertheless, empirical evidence exists associating such categories with long-term, ongoing information needs. In studies of newspaper readers, the distribution of editorial categories among articles selected for reading has been found to be predictable and stable over periods of several weeks [Allen, 1990; Graber, 1988, Chapter 5], despite the difficulty of predicting choice among individual articles. Although newspaper articles can be thought of as only a sub-domain of the Web, we anticipate that this behavior would still be exhibited.

It is common in studies such as this to use binary relevant/irrelevant classifications when modeling users' judgments of topical relevance. There is significant evidence, however, which suggests that these judgments are more faithfully modeled by a preference ranking. Rorvig [1988] cites experiments which show that human relevance judgments can be expressed on a transitive, interval scale (by relating them to judgments of "prothetic" stimuli such as loudness, weight, etc.); the consistency of relative relevance judgments compared to absolute ones is shown in several empirical studies [Lesk and Salton, 1971; Saracevic, 1975]. Furthermore, absolute relevance judgments can themselves be expressed using appropriately chosen preference rankings. For instance, a 1-pref user induces a binary classification, allowing comparison with techniques which make the prevalent assumption of binary-valued relevance judgments.

In order to actually run a simulation, it is necessary to pick particular preference rankings to represent user interests. The n-pref structure is a simple example which allows variation in the complexity of preferences. Interestingly, in early tests we observed that replacing a topic with the union of several topics had little effect on the results. For instance, we simulated a 1-pref user where topic T_1 actually corresponded to the union of two editorial categories, rather than just a single category; results were very similar to regular 1-pref user results. This robustness to variation in the

	Relevant	Irrelevant
Retrieved	a	b
Not Retrieved	c	d

Precision $= \frac{a}{a+b}$ (the proportion of retrieved documents that are relevant)
Recall $= \frac{a}{a+c}$ (the proportion of relevant documents that are retrieved)
Fallout $= \frac{b}{b+d}$ (the proportion of irrelevant documents that are retrieved)

Table 2.1: Measures assuming binary-valued relevance, where $a+b+c+d = |D|$, the total number of documents in the corpus.

composition of topics suggests that different structures using the same $\succ$ relation would not shed any new light on the specific issues investigated by our experiments.

2.3 Evaluations

This section explains and justifies the rather non-standard metric which has been used for evaluation.

2.3.1 Relevance-based evaluation

A binary-valued relevance concept is the major principle behind the evaluation of IR systems, and underlies major comparisons such as at the TREC conferences. The main assumption is that given some query and some collection of documents, it is possible to split the collection into relevant and irrelevant piles with respect to the query. The most popular IR measures, precision and recall, depend on such relevance judgments.

These measures are summarized in Table 2.1, which categorizes documents into four types after a retrieval resulting from a query. In practice most systems return a ranked list of documents rather than just categorizing them into a "retrieved" set (shown to the user) and a "not retrieved" set (discarded). We can think of this as a vertical extension of the 2×2 table. Conversely, as explained in the previous section, most users will be able to rank the returned documents into more than just two categories, leading to a horizontal extension of the table.

Rocchio [1971a] defined normalized recall and precision measures to deal with a vertical extension, and ranking-based precision and recall have been proposed to deal with a horizontal extension [Frei and Schäuble, 1991; Yao, 1995]. The following section shows how the table can be extended in both dimensions.

2.3.2 Evaluation using distance between rankings

The *ndpm* measure of normalized distance-based performance [Yao, 1995] is defined with reference to pairs of documents in rankings. The idea is to measure the distance between the ranking predicted by a recommender system (or an ad hoc retrieval system in Yao's original article), and the actual ranking of the user.

If $\succ_p$ is the ranking predicted by the recommender system, and $\succ_d$ is the user's desired ranking, then we define the distance between two documents $\mathbf{d}, \mathbf{d}' \in D$ with respect to these rankings as follows:

$$\delta_{\succ_p,\succ_d}(\mathbf{d},\mathbf{d}') = \begin{cases} 2 & \text{if } (\mathbf{d} \succ_p \mathbf{d}' \text{ and } \mathbf{d}' \succ_d \mathbf{d}) \text{ or } (\mathbf{d} \succ_d \mathbf{d}' \text{ and } \mathbf{d}' \succ_p \mathbf{d}) \\ 1 & \text{if } (\mathbf{d} \succ_d \mathbf{d}' \text{ or } \mathbf{d}' \succ_d \mathbf{d}) \text{ and } \mathbf{d} \sim_p \mathbf{d}' \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

Now the full measure can be defined by summing δ over all unordered pairs of documents in D, and normalizing by dividing by twice the number of pairs in the desired ranking $\succ_d$:

$$ndpm(\succ_p, \succ_d) \;=\; \frac{\sum\limits_{\mathbf{d},\mathbf{d}' \in D} \delta_{\succ_p,\succ_d}(\mathbf{d}, \mathbf{d}')}{2|\succ_d|} \tag{2.11}$$

This definition ensures the *ndpm* value is in the range $[0, 1]$, with a value of 0.5 representing "baseline" performance—if the predicted ranking consists of a single equivalence class, then *ndpm* will always evaluate to 0.5.

Since this is a distance metric between the user and system rankings for a set of documents, it deals with both horizontal and vertical extensions of the table. Its derivation is detailed in [Yao, 1995], where it is shown to follow from a set of axioms defined by Kemeny and Snell [1962], which they claim a reasonable distance measure between rankings would exhibit. These axioms are both necessary and sufficient for the existence of a unique distance measure. Yao argues that these axioms also make sense in an IR setting, and so goes on to define the *ndpm* measure which satisfies them.

In the ideal case the recommender system is expected to exactly predict the user ranking, so $\succ_p = \succ_d$. This is called the *perfect ranking* criterion. An alternative is the *acceptable ranking* criterion [Wong *et al.*, 1988]: if the user is indifferent as to the ranking of a pair of documents, then the recommender system is permitted to rank them in any order. This criterion is commonly used in IR evaluations, and is adopted here. In fact, it is the cause of the asymmetry of equation 2.10: it does not matter how the recommender system ranks documents which the desired ranking has in the same equivalence class.

This new measure is related to the other IR measures discussed (see [Yao, 1995] for full derivations). In the degenerate case where the rankings are just "relevant" and "irrelevant", *ndpm* can be expressed as a single-valued composite measure of precision and recall. Additionally, it can be shown to be an inverse of Rocchio's normalized recall and a composite measure of ranking-based precision and recall. Its advantages over these earlier measures include the fact that it is a single measure, and the clearly defined assumptions on which it is based. Tests of the Fab system reported in Chapter 5 were apparently the first time the *ndpm* measure had been put into practice; since that time other researchers have also adopted this evaluation methodology (e.g., [Asnicar and Tasso, 1997]).

2.3.3 Experimental methodology

Although each individual experiment to come will be introduced with its own particular methodology, this introductory section will consider some of the common problems in applying the *ndpm* measure, and in interpreting the resulting data. We distinguish between a *live* experiment, where there are one or more human users, and a *simulated* experiment, where there are merely simulations of human users.

Measuring user profile accuracy

Given a set of documents, the desired ranking $\succ_d$ can be elicited: In a live experiment, $\succ_d$ must be derived from the user's feedback—methods for which will be discussed in the next section (section 2.4.1); in a simulated setting, $\succ_d$ needs to be generated according to the simulated user interests. To calculate the predicted ranking $\succ_p$ given a user profile $\mathbf{m}$ and a set of documents D, the documents are ranked according to the value of $\textsc{sim}(\mathbf{m}, \mathbf{d}_i)$.

Given the resulting two rankings $\succ_p$ and $\succ_d$, equation 2.11 provides the value of *ndpm*—the distance between the rankings. When measuring this at regular intervals, a decrease indicates that the system is learning an increasingly accurate model of the user. Thus, the profile accuracy test consists of measuring the *ndpm* value at successive points in time. Learning curves such as this are especially appropriate for recommender systems, since speed of adaptation could form a deciding factor in users' decisions as to whether to use a system.

Yao's original formulation [1995] assumes that the entire document collection is ranked by both the user and the system. This is clearly impractical. An alternative is to measure predictions and actual rankings just for the recommendation sets provided to users. However, these are typically carefully selected documents chosen to satisfy the user, and so may be close to each other in terms of user preference. Furthermore, when using a recommender system users do not take very long to discover that the system is adapting based on their feedback. Their feedback soon stops reflecting their true interests and becomes just an artifact of their attempts to control the system. For instance, a user who no longer wishes to receive pages about Bordeaux will rate all pages about Bordeaux with the lowest possible score, despite the fact that these pages are still more appealing than, say, pages about antelopes. This kind of behavior has been consistently observed throughout all of the user tests conducted as part of this thesis.

Therefore we have used variants of the following scheme. At regular intervals the users are given a special list of randomly selected documents, and asked to rate them according to their own interests. The ratings are not used for adaptation, so there is no ulterior motive for the user. The users' ratings are translated to a ranking $\succ_d$, and the system's predictions form the ranking $\succ_p$.

In a simulated setting, a slightly different scheme is used. The corpus D is split into a training set and a test set. Four training iterations are followed by a test iteration; test iterations are numbered 0, 5, 10, 15, etc.

Training Iteration A training iteration is a regular iteration of the simulation system. On each such iteration, for each user, a set of six documents is selected from the unseen portion of the training set, using the user's current profile. The user's ranking of these documents is simulated according to their known topic preferences. The learning algorithm uses this ranking and the document representations to update the user profile.

Test Iteration On each test iteration, for each user, the **entire** test set is ranked according to the user profile, forming the predicted ranking $\succ_p$. The desired ranking of the test set $\succ_d$ is known from the user's topic preferences (recall that users are simulated by the n-pref structure defined earlier). The *ndpm* difference between these rankings is calculated and recorded.

For both live and simulation experiments, *ndpm* values are averaged over all users at the same iteration. Simulated runs generally used 500 users.

Comparison with other sources

Having determined that the system is learning an increasingly accurate model of the user says nothing about the resulting quality of the recommendations. It is possible for the system to learn a perfect model of the user without it ever recommending any pages the user wants to see (e.g., it could always correctly predict that the user would hate every recommendation).

Unfortunately, recommendation quality is harder to measure within the system, and as a result several of the experiments to be described introduce external comparison points. For instance, if the pages recommended by the system are combined with pages from other sources, then the resulting ranking by the user can be used to determine the relative performance of the sources.

If $R = \{\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_n\}$ is the set of documents recommended for a particular iteration, and documents $\{\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_\alpha\}$ come from source A (where $\alpha \leq n$), then the *ideal ranking* for source A is:

$$\succ_A = \{(\mathbf{r}_i, \mathbf{r}_j) \mid i \leq \alpha, j > \alpha\} \tag{2.12}$$

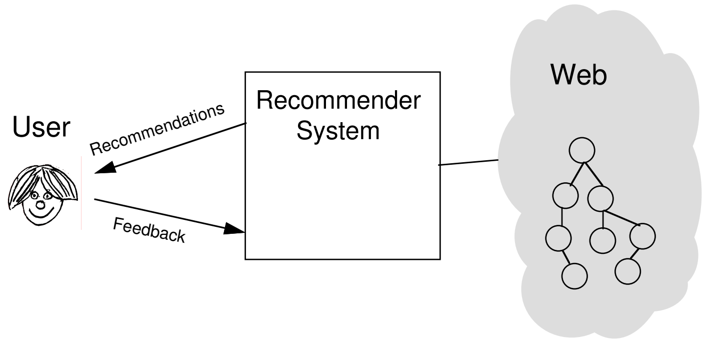

Figure 2.1: A simple model for a recommender system.

In other words, the ranking where the user prefers any document from A to any document not from A, but is indifferent to how documents from A are ranked relative to one other, and how documents not from A are ranked relative to one other. By comparing the user's actual ranking $\succ_d$ to $\succ_A$ (using the *ndpm* measure as before), a score for each source can be calculated, relative to the other sources in the experiment.

In a simulated setting, a user's topic preferences are known. Therefore, a more direct measure is provided by simply recording the percentage of documents delivered from each topic.

2.4 Interactions

Figure 2.1 shows a simple model for a recommender system, and even more generically many IR systems could be characterized in this way. When using a recommender system, each interaction loop would proceed in the following two steps:

1. The recommender system selects a set of documents to present to the user, based upon the user model learned so far;

2. User feedback on these documents is obtained.

The purpose of this final part of the chapter is to view a recommender system in terms of this simple model of interaction—in terms of its inputs and outputs. The frameworks defined for thinking about types of input and strategies for selecting output will prove useful when describing particular implementations and especially when user interface designs are considered.

2.4.1 Inputs: user feedback

Taxonomy of feedback types

The original purpose of relevance feedback on individual documents was to improve queries to retrieval systems [Rocchio, 1971b]. Its subsequent adoption by information filtering and recommender systems has led to a muddling of two notions: indicating that a document is relevant to a topic or user profile, and indicating that more similar documents should be recommended in the next iteration. Clearly, these do not always coincide. As in the example of Chapter 1, one can enjoy an article about Bordeaux wines without desiring the following day's newspaper to be solely concerned with wine (presumably, though, in the reverse case the two notions **do** usually coincide—if the Bordeaux article was disliked, then similar articles are not desired in the future). For a recommender system which must deliver small sets of recommendations, separating "I like this" from "more like this" is crucial; for a retrieval system focussed on a specific query, the distinction can safely be ignored.

Studies of information seeking with human intermediaries [Spink, 1993] have shown that magnitude feedback—where the user wishes to increase or decrease the size of a result set—occurs even more frequently than relevance feedback. A similar notion can be applied to recommender systems: the user may want to control the proportions between different interests represented in their recommendation set (e.g., the proportion of Bordeaux and wine articles compared to health and flu articles). This is equivalent to "more like this"/"less like this" feedback, but applied to sets rather than individual documents.

Table 2.2 lists some user behaviors which can provide information about individual documents. As explained in Chapter 1, explicit feedback encompasses activities that take place in the context of the recommender system—the user must explicitly take action in order to provide it. In contrast, activities that take place in the context of the user's regular work lead to implicit feedback (although note that these boundaries become fuzzier if the user interacts with the recommender system in the course of regular work). Not surprisingly, the obtrusiveness of the feedback gathering mechanism correlates with its cognitive load upon the user. The implicit ways of gathering feedback are entirely unobtrusive, since they merely monitor activities the user is performing anyway. Note that it is not always preferable to have the most direct form of feedback. For instance, as already stated, absolute-valued human relevance judgments are less consistent than relative relevance rankings [Rorvig, 1988]; experiments showing explicitly supplied feedback can be less reliable than implicitly gathered feedback are cited in [Rich, 1979]; reading time has been shown to correlate highly with interestingness of articles [Morita and Shinoda, 1996], in some cases more highly than explicit ratings (although this latter technique is difficult to deploy outside of a controlled environment, since it is hard to know where a user's attention is directed).

The interactivity of any specific recommender system will suggest further additions to the somewhat generic list of Table 2.2. For instance, a system embedded within a word processor can monitor all of the user's menu selections, help queries, operation

Activity		Feedback type	Cognitive load
Reading (being timed)	How long spent reading an article?	Implicit	None
Clicking	Which article title was clicked on?	Implicit	None
Grouping	Which articles were grouped together in a folder?	Implicit	None
Ranking	Which article is this one preferable to?	Explicit	Low
Rating	What is the score for this article?	Explicit	Low
Specifying query	What is a query to retrieve relevant articles?	Explicit	High

Table 2.2: User activities which provide feedback.

```
Excellent
Very Good
Good
Neutral
Poor
Very Bad
Terrible
```

Table 2.3: Ordinal scale on which users rank documents.

initiations and cancellations, and so on.

Eliciting explicit ratings

For many of the systems described in this thesis, users are required to directly enter explicit ratings for recommended items. An ordinal scale is used to obtain user ratings (an ordinal scale is an ordered list of values: the order is significant, the actual values are not). The user places each document into one of seven categories as shown in Table 2.3 (Cox [1980] recommends 5, 7 or 9-point scales; the adjectives are selected from [Mittelstaedt, 1971]). This gives seven equivalence classes in the relation $\succ_d$, and it is guaranteed to be a weak order.[6]

Later experiments will show how implicit feedback can also successfully be used.

2.4.2 Outputs: recommendation sets

A useful way to characterize the output of a recommender system is describe it in terms of a *document selection strategy.* This is an algorithm which picks the documents to show to the user in a recommendation set (denoted $R_{\mathbf{m}}$, where $\mathbf{m}$ is the user profile), given as input some collection of documents ranked according to various measures.

$$\{\succ_1, \succ_2, \succ_3, \ldots\} \longrightarrow \boxed{\text{Document Selection Strategy}} \longrightarrow R_{\mathbf{m}} \tag{2.13}$$

An example of a strategy is to output those documents with the highest topical relevance. The only required input to this strategy is a ranking according to topical relevance, relative to a user profile. The strategy then simply picks off the top few documents. This is the typical strategy for an ad hoc retrieval system.

For a recommender system there are other justifiable strategies. For instance, a ranking according to novelty would enable a document selection strategy which somehow traded off novelty with topical relevance (novelty was introduced in section 2.2.3).

The terms *instance-based* and *set-based* apply equally well to document selection strategies as they did to factors influencing human judgments of documents. Thus a strategy of picking a recommendation set purely according to topical relevance is instance-based, since each document is evaluated independently. In contrast, a

[6]Note that in a very early experiment an 11-point scale with numeric values ranging from +5 to -5 was used.

novelty-based strategy of picking documents such that each is maximally different from the other recommendations is clearly set-based.

Individual strategies will be defined in later chapters; the purpose of this section has been just to introduce this form of description for recommender system outputs. In addition, it illustrates a decomposition which allows an arbitrary IR "engine," producing a ranking according to topical relevance, to be connected to a document selection strategy to form a text recommender system.

2.5 Summary of notation

The following symbols will be used consistently throughout this thesis.

- $\succ_d$ - ranking desired by the user.
- $\succ_p$ - ranking predicted by the system.
- D - the set of all documents, the corpus.
- $\mathbf{d}$ - a document vector.
- $\mathbf{m}$ - a user profile vector.
- $R_{\mathbf{m}}$ - a recommendation set relative to user profile vector $\mathbf{m}$.
- SIM - dot product similarity metric, measures both document-document, document-profile and profile-profile similarity.
- T - a topic: a set of document vectors forming a cluster of topical relevance as judged by a human.
- t - a term (a word).
- U - a set of user profile vectors.

Chapter 3

Single Agent Content-Based Recommendation

Up to now we have yet to discuss how to put into practice any of the ideas discussed. This chapter is a guided tour of a simple architecture for a text recommender system, dealing with just a single user (for multiple users, multiple instances of this system are necessary). Various design decisions and parameter settings are justified with reference to specific investigations carried out in a simulated setting, which is also introduced in this chapter.

3.1 Design

As illustrated in Figure 3.1, the process of recommendation can be partitioned into two stages: *collection* of items to form a manageable database or index, and subsequently *selection* of items from this database for a particular user. The *learning* phase completes the feedback loop. In some instances the collection stage is trivial or performed by a third party, but in the case of the Web it is a real problem faced by the system designer.

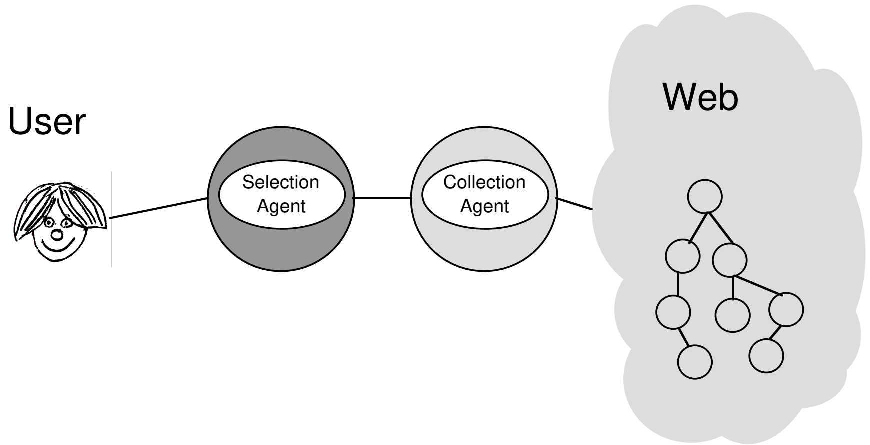

Figure 3.1: Basic system with a single user and a single agent (shown split into a collection and a selection phase).

The processes of Figure 3.1 are referred to as the *collection agent* and *selection agent*, respectively. By agent, we mean just a process with long-term state; an "agent architecture" is a description of roles of agents and communications between them. "Distributed object system" would be equally apt, but would not so clearly reveal the artificial intelligence (AI) origins of this research.

A useful analogy is to think of the collection agent as a "journalist," discovering articles of interest (although of course it does not create new articles as a human journalist would). The selection agent is then an "editor," composing the personalized newspaper given the collection agent's output. The collection/selection decomposition is not really necessary for the simple architecture described here. However, it is introduced as it will prove useful for describing the more complex systems to come.

A simple "learning information retrieval agent" (known by its acronym LIRA) conforming to this architecture was implemented in 1994 [Balabanović and Shoham, 1995]; section 3.4.2 details experiments conducted with this prototype. Its method of operation was as follows:

1. Search the Web, using some user profile as a search heuristic, taking a bounded amount of time—this is the collection phase.

2. Select the best 10 pages to present to the user, using some document selection strategy (which may be just based on the user profile)—this is the selection phase.

3. Receive explicit feedback from the user for each page presented.

4. Update the user profile according to this feedback.

Users viewed their recommendations a maximum of once per day; the collection agent performing Web search did so every night.

3.1.1 Collection

Although various companies and organizations attempt to exhaustively search (and index) the entire Web, this is both unnecessary for recommendation purposes and expensive in terms of CPU and disk resources. A less resource-hungry alternative is to create algorithms which actively search for documents relating to known interest areas.

There is a clear mapping between the problem of searching the Web and the classic AI search paradigm. Each page of the Web is a node, and the hypertext links to other pages are the edges of the graph to be searched. In typical AI domains a good search heuristic will rate nodes higher as we progress towards some goal node. In our Web domain, the search heuristic is a user profile, and models how interesting a page is to a user. There is not necessarily any correlation with the heuristic values of nearby pages. However, the "one good page leads to another" assumption posits that there is often a correlation between the subject matter of hyperlinked pages. Experiments to be described show that this is indeed the case for certain domains.

A standard algorithm for simple best-first search maintains a list of "the best pages to visit next." Unfortunately, the very high branching factor of typical Web pages, together with a size which can almost be regarded as infinite (given the dynamic addition and change of pages), mean that this list will expand rapidly. A more practical algorithm, which allows a system administrator to effectively plan resources, is *beam search*, which limits the size of the search fringe by allowing pages with the lowest heuristic value to "fall off" the bottom. Most collection agents implemented had a search fringe with at most 500 URLs. One negative consequence is the necessity of maintaining a separate, compact representation of all pages previously seen, to prevent loops (document signatures, as defined in section 2.1.1, are used as the compact representation). Further information on AI search algorithms can be found in most AI textbooks, e.g., [Rich and Knight, 1991].

A possible elaboration of this method, not used in the systems described here, is to attempt to categorize the links available from a page. For instance, Spertus [1997] categorizes links into upward, downward, crosswise and outward (based on an analysis of URL strings), and posits heuristics about the results of following different link types.

3.1.2 Selection

The search process has a record, at any time, of the best documents found so far, ranked according to SIM as defined in section 2.4.2. These can easily be extracted upon termination. The document selection strategy applies two other simple rules, both suggested by early user tests:

- Do not show users any documents already recommended in the past 30 days. This is implemented using a "history" hash table indexed according to document signatures. It is a simple application of the novelty factor.
- In a single recommendation set, include at most one page from any site (where a "site" is defined simply as all pages originating from a single IP address). If a page scores very highly, other pages at the same site will often also score highly. However, showing users pages which are children or siblings of each other in the graph both wastes their time and narrows the feedback available to the system. Note that this technique can, on occasion, be overly restrictive—some IP addresses are host to many distinct "sites" as perceived by a user (e.g., organizations that provide free Web page hosting to individuals have essentially no intra-site consistency).

The end result is a recommendation set consisting of 10 pages.

3.1.3 Learning

This system requires users to give explicit feedback, according to a scale as defined in section 2.4.1 or the 11-point scale used for the LIRA system to be described. This feedback is transformed into an integer, with 0 representing the midpoint of the scale.

Relevance feedback

Each recommended page $\mathbf{d}_i$ will be viewed by the user and receive an integer evaluation e_i. Given this information, the relevance feedback method [Rocchio, 1971b] updates the weights of user profile $\mathbf{m}$ by a simple addition. If $\mathbf{m}_k$ is the profile after the k^{th} step, then:

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \sum_{i=1}^{10} e_i \mathbf{d}_i \tag{3.1}$$

In the ad hoc retrieval setting, relevance feedback is used to update query weights given feedback on returned documents. The update rule of equation 3.1 is equivalent to the "Ide regular" rule [Salton and Buckley, 1990] with additional weighting from the users' evaluations. In fact Salton and Buckley found that the "Ide dec-hi" rule generally gave better performance, where only one negative-scoring document is included. However the absence of an initial user-provided search query in our system increases the importance of negative feedback, and so the present scheme was adopted. Variants of relevance feedback have been studied in the context of the routing task developed for the TREC conferences, e.g., [Buckley *et al.*, 1994; Allan, 1995], and in an incremental setting, where user judgments are available a few at a time [Allan, 1996].

In machine learning terms this is a very simple variant of an exemplar-based scheme [Kibler and Aha, 1987], where we incrementally move a single prototype point and classify instances according to their distance from this point. There have also been numerous comparisons between non-incremental ML techniques and variants of relevance feedback, e.g., [Schütze *et al.*, 1995; Lang, 1995; Pazzani *et al.*, 1996]. One of the disadvantages of such techniques is the large number of examples that are required before the learning algorithm can be applied. In contrast, relevance feedback assumes an on-line model where the user gradually sees better and better pages, and does not assume a fixed document collection—users' feedback influences which pages are collected and shown to them.

Although *relevance feedback* has been used in the literature to refer to a variety of different techniques, from now on we will use the phrase to refer to precisely the update rule defined above.

Gradient descent

An alternative method is *gradient descent* [Wong and Yao, 1990], which is a variant of the perceptron learning rule [Rosenblatt, 1960].

On each iteration the user's desired ranking for a set of documents can be inferred from the ratings they give. In addition, the recommender system can use the profile learned thus far to predict a ranking over the same set. An error vector can then be computed for each pair of documents that the recommender system ranked incorrectly. If the user is indifferent as to the ranking of a pair of documents (i.e., if they received the same score), then the recommender system is permitted to rank them in any order. This is just the the acceptable ranking criterion again. An error vector is the difference between two incorrectly ranked document vectors. These error vectors are added to the existing profile vector to update it for the next step:

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \sum_{\mathbf{b} \in \Gamma(\mathbf{m}_k)} \mathbf{b} \tag{3.2}$$

where $\Gamma(\mathbf{m}_k) = \{\mathbf{b} = \mathbf{d} - \mathbf{d}' \mid \mathbf{d} \succ_d \mathbf{d}' \text{ and } (\mathbf{d}' \succ_p \mathbf{d} \text{ or } \mathbf{d}' \sim_p \mathbf{d})\}$

This is known as the *adaptive linear model* of IR. A single step of this procedure is very similar to relevance feedback, generating Rocchio's *optimal query* if started with an empty vector. If the user's document preferences can be represented as a weakly linear preference function (equation 2.8), then the procedure above is guaranteed to converge after a finite number of steps, regardless of the value of the starting vector $\mathbf{m}_0$ [Wong and Yao, 1990] (in other words, there will be a step k where $\Gamma(\mathbf{m}_k) = \emptyset$).

Two different implementations can be imagined:

Single Step In the simplest scheme, given a previously computed user profile and a set of recommendations with associated preference information, the resulting errors can be added to the existing profile just once, i.e., taking one step of gradient descent. Therefore after any set of recommendations, the learning system has effectively performed a single step of gradient descent over all of the documents recommended thus far. The advantage of this method is that previously recommended documents and preferences among them need not be stored. Storage is only required for a user profile per user. The disadvantage is that only a single step of gradient descent can be performed.

Multiple Steps In a more complete scheme, preferences and document representations for all recommendations ever provided are stored. User profiles can be computed afresh at any time, over all the documents recommended thus far. Any number of gradient descent steps can be performed. In a modification of this scheme, a previously computed user profile can be used as a starting point for the gradient descent procedure. Note that by storing preferences for only a fixed amount of time, or by decreasing the influence of older preferences, it is easy to implement a bias for newer judgments using this scheme (for the single step scheme, it would be necessary to decay weights in the learned profile).

The relative performance of these two schemes is investigated in the simulation experiments to follow.

Gradient descent schemes similar to the one described have been successfully applied to text categorization [Lewis *et al.*, 1996; Ng *et al.*, 1997]. Sumner and Shaw [1996] have shown that the adaptive linear model compares favorably to a probabilistic relevance feedback method on a two-stage retrieval task.

The advantage of gradient descent over relevance feedback is that absolute ratings are not required, relative rankings are sufficient. The disadvantage is that, even if only the single step scheme is followed, more computation is required.

Alternative methods

Many competitors to relevance feedback and gradient descent exist. For the retrieval and routing tasks, relevance feedback has been empirically shown to be comparable to other approaches. Furthermore, experiments have shown that queries automatically generated through relevance feedback can outperform human searchers [Foltz and Dumais, 1992; Harman, 1994]. However, more recent work on the filtering and supervised classification tasks suggest that an instance-based algorithm might be better. In the k-nearest neighbors (k-NN) algorithm, a new instance is classified by comparing it to previously seen instances. The label for the new instance is predicted by looking at the known labels of k "nearest neighbor" previously seen instances—those which are closest in terms of features (usually using some distance metric such as Euclidean distance for numeric features or Manhattan distance for binary features). To apply this algorithm to a text recommendation task, all previously recommended documents must be retained. The score of a new document is the average of its k

nearest neighbors in the vector space defined for documents, where the score for each neighbor is further weighted by its distance to the new document. Yang [1994] has shown that this algorithm outperforms relevance feedback for the supervised classification task. Its disadvantage is the large amount of storage required for past recommendations. Furthermore, direct comparisons on an incremental task such as text recommendation have not yet been performed. Nevertheless, an interesting extension of the research reported in this chapter would be to implement and compare different learning algorithms.

More algorithms which have been tried for filtering or classification tasks include: incremental construction of neural networks [Jennings and Higuchi, 1993], using LSI to define dimensions for user profiles [Foltz and Dumais, 1992], genetic algorithms [Sheth and Maes, 1993] and using the minimum description length principle to find good models [Lang, 1995].

3.2 Implementation

The "LIRA" system was implemented during the second half of 1994[1] with the help of Yeogirl Yun. The collection agent was written in C++, and ran nightly. Wrapper code for presentation, selection and learning was written in Python. All Web access was routed through an HTTP proxy server, which allowed the system to uniformly use the HTTP protocol when accessing a variety of different kinds of service, including Gopher, Usenet News and WAIS.

The interface to the system used HTML forms—Figure 3.2 shows a screen shot. This meant that access was available from any Web browser.

Unlike in later systems, LIRA used an 11-point scale for explicit feedback. The

[1] For historical perspective: in June 1994, the Web had approximately 2,700 servers. By December 1994, when the experiments of section 3.4.2 were conducted, the were around 10,000 servers, and by January 1997 about 650,000 [Gray, 1997]

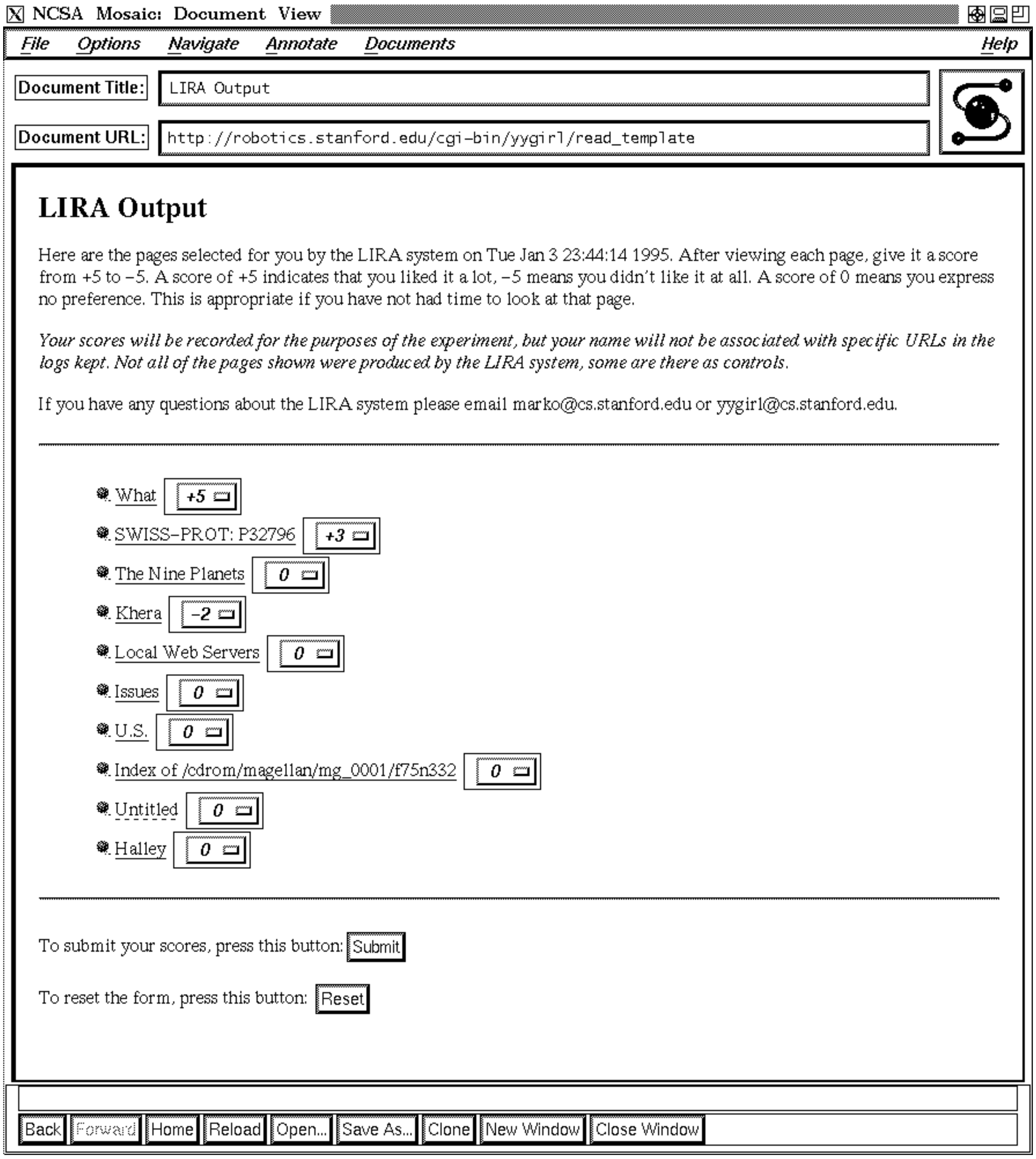

Figure 3.2: The LIRA interface

points on the scale had numeric labels: the integers $-5 \cdots +5$. The relevance feedback learning method was used, and only the 10 highest-weighted words were used for each document vector $\mathbf{d}$.

3.3 Simulation

As well as live tests of LIRA, this chapter reports on some investigations in a simulated setting. That setting is introduced here, and we will return to it in later chapters as further tests are carried out. Of course in an ideal world all empirical work would occur in the live setting. However, studies with human subjects are expensive and time-consuming to perform, and so are often carried out on a relatively small scale, e.g., [Sheth and Maes, 1993; Resnick *et al.*, 1994; Lang, 1995; Pazzani *et al.*, 1996]. A simulation provides a complementary arena for research: populations of hundreds of users over periods of several virtual months are easier to study, and underlying phenomena can be isolated for more thorough analysis.

As explained in section 2.2.5, the n-pref structure is used to represent user interests for the purposes of simulating their responses.

The corpus used here is a collection of Web pages gathered by following links from the Yahoo! topic hierarchy. This is a collection of Web pages organized by human editors into a hierarchy according to topical relevance, and so fits the notion of editorial categories.

Ten topics were gathered (Table 3.1) during May and August 1997, a total of 7840 pages. For each topic, a 1-ply search was conducted following all hyperlinks from the starting URL, so in essence each gathered topic includes two levels of the hierarchy. Links which pointed "up" the hierarchy or to generic Yahoo! help information were ignored. In addition, pages which occurred in more than one topic were removed (a total of 23 pages, or about 0.3% of the total).

Name of topic	URL (http://www.yahoo.com prefix omitted)	Number of pages
Music	/Entertainment/Music/	788
Travel	/Recreation/Travel/	1463
Religion	/Society_and_Culture/Religion/	790
Finance and Investment	/Business_and_Economy/Finance_and_Investment/	518
Multimedia	/Computers_and_Internet/Multimedia/	512
Outdoors	/Recreation/Outdoors/	655
Auto Racing	/Recreation/Sports/Auto_Racing/	490
Literature	/Arts/Humanities/Literature/	1470
Food and Eating	/Entertainment/Food_and_Eating/	457
Law	/Government/Law/	697
	Total	7840

Table 3.1: Topics used for simulation corpus.

Each gathered topic was randomly split, with 75% of the documents going to a training set and the remaining 25% to a separate test set (in total the training set had 5884 pages and the test set 1956). In order to maximize realism and retain the noisiness characteristic of Web data, the corpus was left as near as possible to its raw state. As a result, many documents contained few or no words (e.g., 676 documents, about 9% of the total, had vectors with fewer than 10 words, and 45 documents had empty vectors).

The categories chosen are clearly only a small sample of the many possible clusters of topical relevance users might find interesting. However, since each category encompasses a diverse set of Web pages judged by a human editor to be topically similar, we hope that it is also provides an appropriate example with which to conduct tests, and that the results would apply equally to further editorial categories.

3.4 Experiments

Three very different sets of experiments relating to the single agent pure content-based system are described. Although all logically belong together, they were performed at different times; some in live and some in simulated settings. The first presents anecdotal results from early attempts at creating a recommender system. The second shows results from two live tests of the LIRA system, and the third presents a variety of simulation studies comparing different learning methods, parameters within those methods and different text representations.

3.4.1 CS227 Class Projects

Before discussing the experimental results from LIRA and the simulation, it is worthwhile noting the origin of this design and indeed the starting point for this thesis. The Stanford University class CS227, "AI Programming in Prolog," was taught, in the Spring of 1994, by Yoav Shoham with myself as a teaching assistant. A simplified text recommender system was set as the class project, in order to give students a chance to experiment with search and machine learning algorithms in a fun setting. Here is an excerpt from the class handout describing the project (at the time called "The Automatic Hotlist"):

> "You walk in to your office in the morning and sit down at your computer. It has been patiently whirring away the whole night, scanning information sources all over the world. "Good morning," it says, "Here are some articles I thought you'd want to see." "A-ha!" you say, "These two are great. But this one about Prolog is really boring. Don't show me stuff like that anymore." "OK," says the computer, adjusting its parameters. Tomorrow its recommendations will be slightly better tuned..."
>
> from CS227 Handout #9, 10 March 1994

```
1.  http://www.service.com/PAW/thisweek/sports/1994-Jun-3.new-and-recommended.html
2.  http://www.commerce.digital.com/palo-alto/chamber-of-commerce/events/wCup/home.html
3.  http://www.service.com/PAW/thisweek/sports/1994-Jun-3.sports-shorts.html
4.  http://www.atm.ch.cam.ac.uk/sports/wc94.html
5.  http://www.cedar.buffalo.edu/khoub-s/WC94.html
⋮
```

Figure 3.3: First five entries from a sample top-ten list produced by a student's program.

In order to provide a controlled and safe test-bed for the students' agents, the first task was to cooperatively construct a self-contained "mini-web" of interlinked pages. Although the mini-web contained only 240 nodes and 850 links, it was surprisingly easy to experience the well known "lost in hyperspace" feeling.

Early assignments included building a map of the entire mini-web and inventing heuristics for best-first search, which were tested by searching for pages containing specific keywords. Eventually learning components were also incorporated, creating complete agents similar to the description in section 3.1.

After testing the agents on the mini-web, the final for the class involved sending them out into the real Web to see what they could find. A trial over twenty simulated days was run for each agent (each day actually being ten minutes of real time), with "daily" feedback from a user. The system presented 5 new pages each day, and at the end of the twenty days supplied an ordered "top-ten" list of what it considered the most interesting pages found.

Search: Most of the projects based their search on a standard best-first algorithm, with bounds imposed on the memory and time used for each search.

Features: Features ranged from TFIDF as described, to schemes where words inside HTML tags were rated higher, to using structural features such as the length of a document or the number of pictures it contained.

Learning: Learning methods included a neural network approach, variants on nearest neighbor schemes, and various simpler weight updating strategies.

Given the short time-frame, the systems produced were remarkably successful. However, due to the informal nature of the experiments, only anecdotal results can be reported. In one experiment, a user gave high scores to pages which related to soccer and the 1994 World Cup. After the twenty days of feedback, the system presented a top-ten list where the top five pages were World Cup related (Figure 3.3). In another experiment, a preference for Japan-related pages led to a top-ten list containing only such pages. In fact most of the testers reported similar successes.

3.4.2 Tests of LIRA

The LIRA system was a direct outgrowth of the class project described above.

Learning a single-topic preference

Two very different experiments were performed to test LIRA. In the first one we attempted to verify that the algorithm could converge on a user's interests if given a well-defined single-topic preference, equivalent to a 1-pref user in the simulated setting. We chose the broad topic "music." The scoring strategy was as follows:

+5 for pages relating to music.

+2 for pages which although not directly related to music, looked as if they might lead to a music-related page.

-5 for pages unrelated to music.

The experiment was run for 16 iterations ("days") of the algorithm, with 20 minutes of CPU time on a Sun Sparc 10 allowed for each iteration. This corresponds to

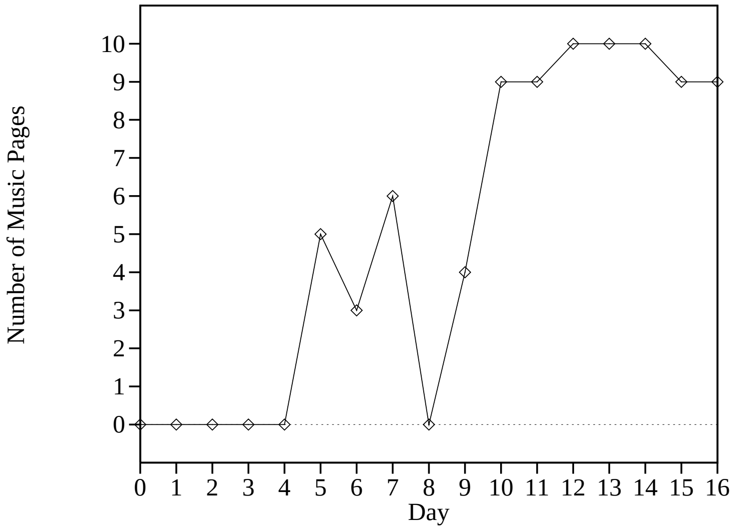

Figure 3.4: Results of an experiment where only music-related pages where rated highly.

several hundred URLs accessed per iteration, depending on the network load. The results (Figure 3.4) show that the system was able to successfully learn the concept of music-related pages, with 9 or 10 out of the 10 pages shown every day relating to music after the 10^{th} day.

Table 3.2 shows the highest-weighted words from the resulting user profile. The first ten are clearly music related (`Page` and `Young` refer to the musicians Jimmy Page and Neil Young). The total vector contains 1110 words.

One interpretation of these results is that the "one good page leads to another" hypothesis holds true for the example of music-related pages.

Comparison with "cool" and random pages

The second experiment was a comparison of sources, following the methodology explained in section 2.3.3. In order to establish the feasibility of the overall approach, the system was matched up against a popular page on the Web where a single hand-picked "cool site" is featured every day. In this way the system could prove its value

music	19.72	mail	1.91
album	8.33	show	1.88
song	6.09	sampl	1.76
record	5.89	indi	1.71
band	5.56	homepag	1.68
page	5.29	databas	1.56
young	4.91	number	1.46
regga	4.13	indielist	1.43
artist	3.66	sound	1.40
ska	3.47	hors	1.38
link	2.85	collect	1.38
list	2.31	email	1.29
search	2.31	perform	1.27
art	2.21	vibe	1.25
tour	2.05	radio	1.24

Table 3.2: The highest-weighted words and their weights from the user profile after the end of the experiment to find music-related pages. These words have been stemmed, e.g. `regga` was originally `reggae`.

to users in comparison with an existing widely used resource. As a simple control, a further source provided randomly picked Web pages.

Each day a separate process took six pages produced by LIRA, three random pages and one human-selected page. These were presented to the user in a random order. In this way neither the user nor the experimenter knew which page came from which source. A log was maintained of the evaluations received for each of the three sources.

The human-selected page described was the "Cool Site of the Day" link [Davis, 1994]. Note that this was not tuned to a particular user. The random pages were selected from a static list of several hundred thousand URLs[2], checked for accessibility. The system did **not** use the evaluations of these control pages to update its weights. The system was allocated 20 minutes of CPU time on a Sun Sparc 10 per user per day.

Figure 3.5 shows the *ndpm* distances to their ideal ranking for each source, from ratings collected during the first 24 days of operation of the system. Since *ndpm* is the distance between desired and actual rankings, a smaller value indicates a more accurate user profile. Not every user used the system every day, especially as the period in question included the Christmas and New Year holidays in 1994. Hence only a few users performed a large number of iterations.

Unfortunately an analysis of profile accuracy is not available for this early experiment (in fact, it predated the publication of the *ndpm* paper [Yao, 1995]—Figure 3.5 is a recent reinterpretation of the original results. However, predictions of user ratings were not made at the time, and there is insufficient data to do so now).

Given the short duration of the experiment and the limitations of the implementation, the results were very encouraging. LIRA's performance is consistently better than the randomly-selected pages. It beats the human-selected page more than half of

[2]Provided from the Lycos database, courtesy of Michael Mauldin.

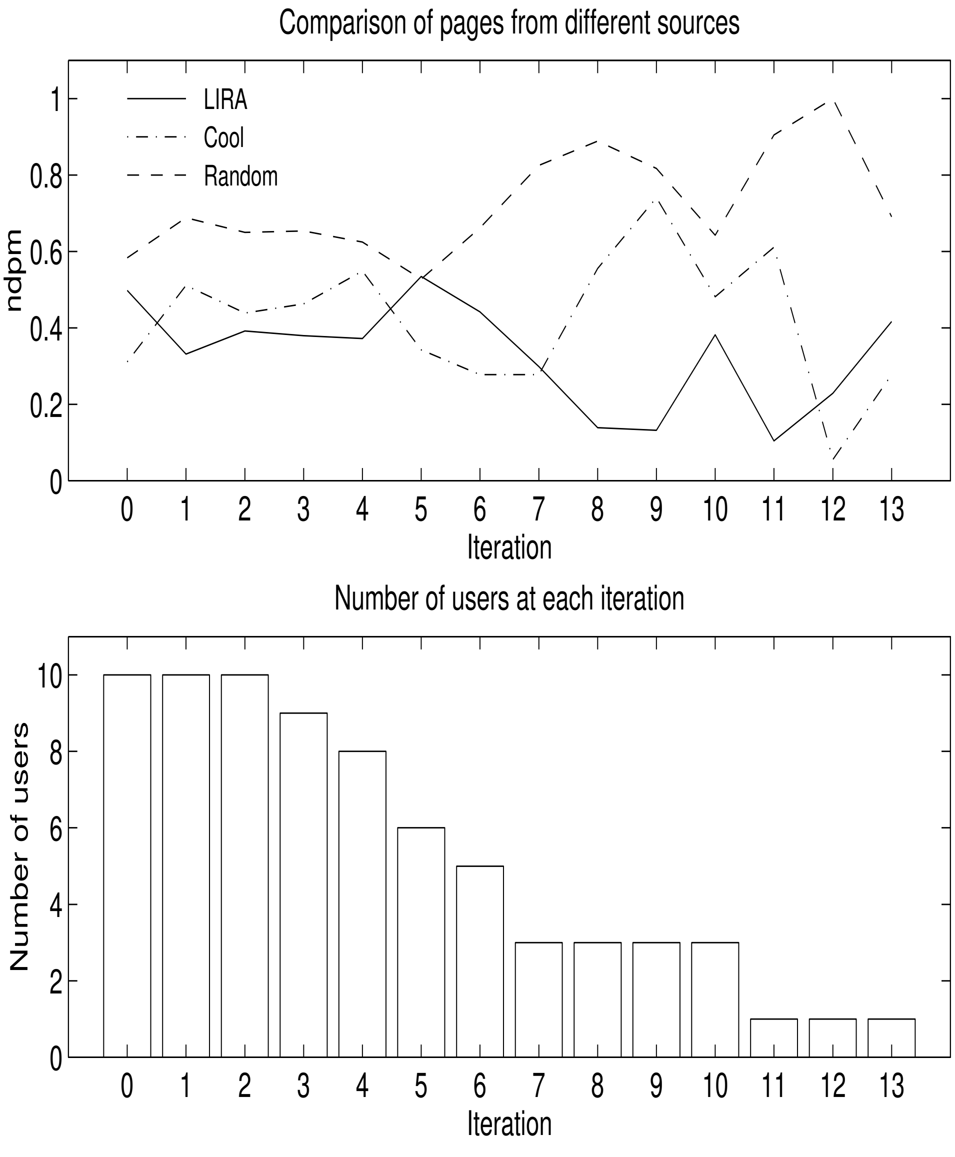

Figure 3.5: Comparison of the LIRA system against random and human-selected ("cool") pages.

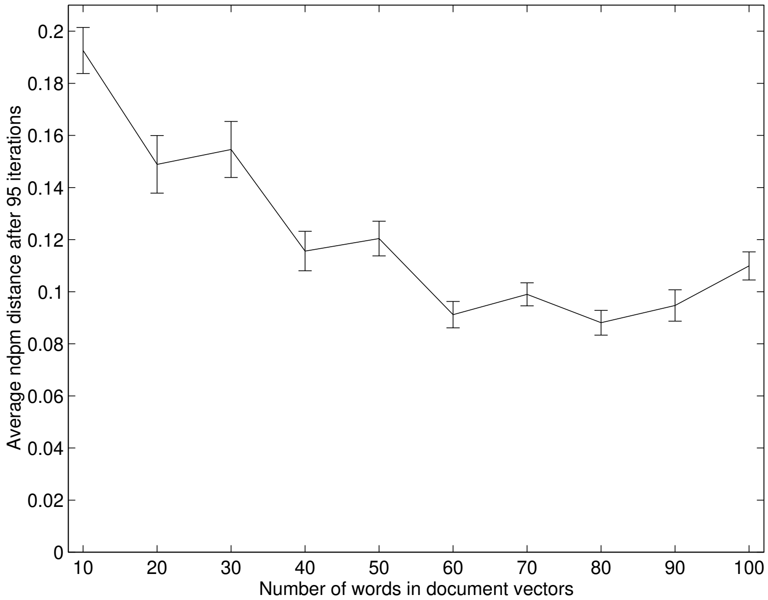

Figure 3.6: Variation of *ndpm* value after 95 iterations of the recommender system with differing numbers of words used from each document, for 1-pref users. Error bars show 95% confidence intervals.

the time. The greater variability in the score for the human-selected page is probably due to the fact that only a single such page was available.

These results provided an initial feasibility test, and encouraged the construction of the bigger and better systems to be described in later chapters. LIRA was taken off-line in early 1995.

3.4.3 Simulation results

The results given above used the relevance feedback learning method and only took the 10 highest-weighted words from each document. In this section these two factors will be examined in more detail, in the simulated setting already described.

Optimizing document vector length

Figure 3.6 shows a series of experiments where the maximum number of words used from each document is varied, from 10 to 100 (in some instances the total number of words in a document will be less than this maximum). The single step gradient descent scheme was followed. The *ndpm* scores shown represent profile accuracy after 95 iterations.

The differences in the 40–100 word range are not particularly significant (the greatest difference is between using 50 and 80 words, and even there $p > 0.8$). Further tests showed no improvement when using more than 100 words per document. These results approximately agree with [Pazzani *et al.*, 1996], who found 96 words optimal for a corpus of Web pages, and our own tests with Reuters articles, where the optimum was found to be between 90 and 100 words. The 10-word scheme employed by LIRA, thus, is seen to be suboptimal (these simulation tests were carried out more recently than the LIRA tests).

Upper bound performance

The gradient descent learning rule defined in section 3.1.3, if iterated until convergence, is guaranteed to learn the desired ranking $\succ_d$ exactly if the weak linearity condition (equation 2.8) holds. In this section we examine the extent to which this condition holds given our particular document representation scheme and the Yahoo! simulation corpus. This will also provide a useful comparison point for *ndpm* values achieved in different scenarios to be introduced.

We apply the gradient descent procedure directly on the test set to determine an upper bound on the performance of our system. This is usually referred to as the retrospective test [Robertson and Sparck Jones, 1976]. Furthermore, the rate of convergence to a correct user profile provides insight into the suitability of the

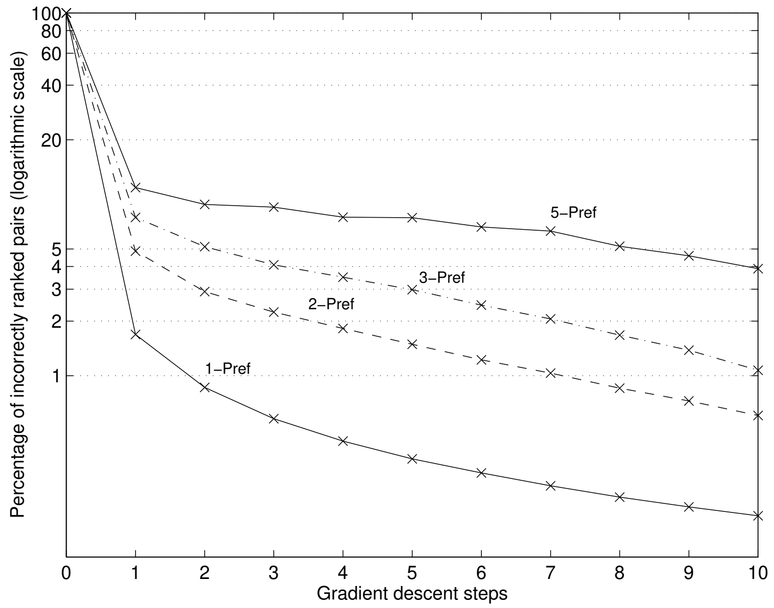

Figure 3.7: Gradient descent on test set (test of optimal learning), in each case averaged among all possible users with the same preference structure, or 500 randomly generated users, whichever is the lesser.

adaptive linear model for this simulation dataset, and Web pages generally (very slow convergence would indicate that perhaps more complex nonlinear functions were necessary).

The gradient descent procedure (equation 3.2) identifies on the k^{th} step the number of pairs of documents ranked incorrectly by the user profile of the $(k-1)^{\text{th}}$ step. The graph of Figure 3.7 shows how the percentage of incorrectly ranked document pairs (plotted here on a $\log_{10}$ scale) decreases with each step of gradient descent, as the procedure converges. Note that pairs about which the desired ranking is indifferent were ignored, so the percentage represents the ratio between the number of incorrectly ranked pairs and the total number of pairs where a preference was required.

It is clear that the majority of the convergence occurs in the first step. The amount of computation involved in a gradient descent step is approximately proportional to the number of incorrectly ranked pairs. As the user profile converges, therefore, the

Number of preferences	Average *ndpm* after 10 steps
1	0.0026
2	0.0099
3	0.0224
5	0.0608

Table 3.3: Comparison of *ndpm* values given different numbers of preferences held by users, after 10 steps of gradient descent directly on the test set.

amount of work for each step becomes correspondingly less. Although this would seem to suggest that it is worthwhile performing extra steps, the significant storage overhead of retaining past recommendations and preferences, required for the multiple steps gradient descent scheme, could outweigh these considerations.

The results of Figure 3.7 suggest that for simpler preference structures the adaptive linear model is well suited to this domain—the feature set and user profile representation are sufficiently rich to capture a good approximation of the preferences in a small number of iterations. Looking at the 2-pref users, after 10 steps, on average less than 1% of pairs are incorrectly ranked. However, full convergence has not occurred for any one user, and this does not happen until the 50$^{\text{th}}$ step. Given the noisy nature of the dataset, it seems wise to ignore these few stubborn pairs as outliers.

As the preference structures increase in complexity, the optimal performance of the linear model worsens, and it would be interesting in these cases to compare the performance of some of the nonlinear models which have been proposed, e.g., by [Fuhr, 1989].

By taking the user profiles generated by the gradient descent procedure over the test set, values of the *ndpm* measure can also be calculated for various cases. Table 3.3 shows the results, which are significantly superior to subsequent runs where the training set is used for learning. These represent the upper bound for the given

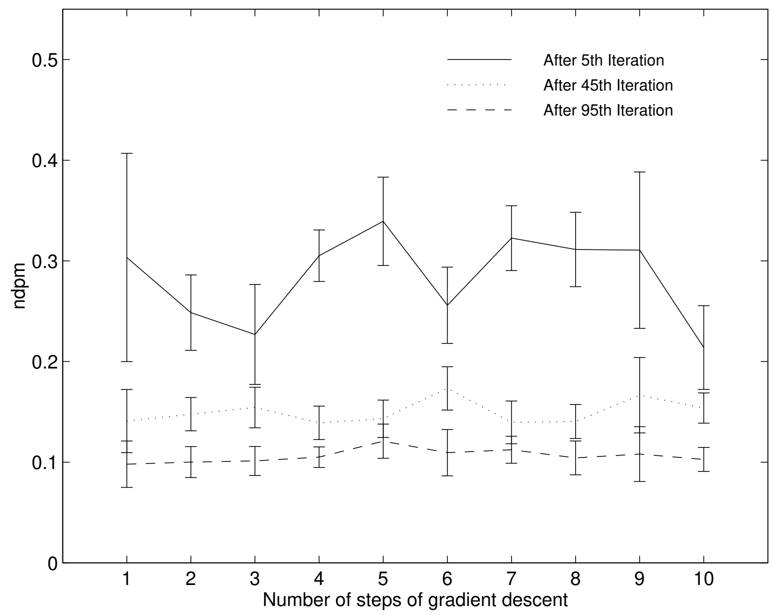

Figure 3.8: Comparison of *ndpm* distances using different numbers of steps of the gradient descent algorithm, after different numbers of iterations of the recommender system, for 1-pref users. 95% confidence intervals shown.

corpus using the single step gradient descent method.

Comparing gradient descent strategies

As opposed to the retrospective tests of the previous section, from here on we follow our stated methodology of alternating four training iterations with a test iteration. Figure 3.8 shows how performance varies against the number of steps of gradient descent done for each simulation iteration, for 1-pref users. There is very little improvement beyond the first step. Therefore, where gradient descent is employed, the single step scheme is used. This has the advantage of requiring less storage space or processing time than the multiple steps scheme.

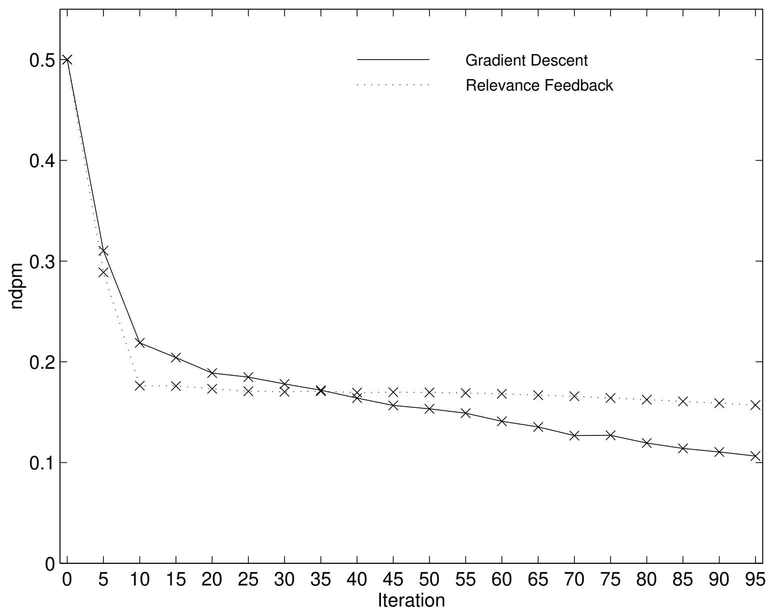

Figure 3.9: Learning curves for 1-pref users comparing a single iteration of gradient descent to relevance feedback.

Comparing gradient descent and relevance feedback

For 1-pref users it is possible to directly compare gradient descent with relevance feedback (as defined in section 3.1.3). Similar to the single-topic experiment of section 3.4.2, it is necessary to simulate particular ratings for the relevance feedback algorithm. This experiment used values of $+2$ when a document was in the preferred topic, and $-\frac{1}{2}$ when it was not. These values were chosen as they have been commonly used in IR relevance feedback evaluations, e.g. [Allan, 1996]. Thus the comparison here is between "generic" relevance feedback and gradient descent schemes—neither have been optimized for the data set.

The results are shown in Figure 3.9. Despite not having the extra information inherent in absolute scores, the gradient descent procedure performs comparably on this dataset. Nevertheless, the relevance feedback method is used in the majority of the systems described in this thesis, due to its appealing speed and simplicity.

3.5 Problems with the pure content-based approach

The experimental results shown have confirmed that a recommender system employing the vector space representation, gradient descent or relevance feedback learning, performing beam search on the Web, can successfully recommend documents of interest to users with simple preferences.

However, a pure content-based system as defined has several shortcomings. In general only a very shallow analysis of only certain kinds of content can be supplied. In some domains the items are not amenable to any useful feature extraction methods with today's technology (e.g., movies, music, restaurants). Even for text documents, the vector space representation captures only topical relevance, as explained in section 2.1.1.

A second problem, which has been studied extensively both in this domain and in others, is that of over-specialization. When the system can only recommend items scoring highly against a user's profile, the user is restricted to seeing items similar to those already rated. Often this is addressed by injecting a note of randomness—for example, in the context of information filtering, the crossover and mutation operations (as part of a genetic algorithm) have been proposed as a solution [Sheth and Maes, 1993].

The LIRA architecture, in common with many information filtering systems, has linear complexity—a new collection and selection agent must be added for every new user. Given that the collection agents, searching the Web, are expensive to run in terms of resources, this property severely limited the number of users that could be served from our research facilities. A system where the collection agent is a simpler and less resource-intensive standing query (connected to a database or newsfeed) faces the same issue, albeit not until greater usage levels.

Finally there is a problem common to most recommendation systems—eliciting user feedback. Rating documents is an onerous task for users, so the fewer ratings are required the better. With the pure content-based approach, a user's own ratings are the only factor influencing future performance, and there seems to be no way to reduce the quantity without also reducing performance. A corollary of this is that every new user to the system must start training an agent from scratch.

In the following section it will be seen how a collaborative approach can overcome many of these problems. However, it causes an additional set of problems of its own. In Chapter 5, a hybrid scheme will be introduced, which eliminates the difficulties caused by using either approach alone.

Chapter 4

Collaborative Recommendation

The content-based system defined in the previous chapter fits within the purview of information filtering [Oard, 1997], and uses techniques originally developed in the context of ad hoc retrieval by the IR community. A contrasting approach is *collaborative* recommendation, a more recent area of research which has variously also been termed collaborative filtering and social filtering. Instead of extracting features from items (such as using constituent words to represent documents), a user model is made up of unprocessed preferences over previously seen items. By computing similarity between users, an item liked by one user can be recommended to similar users.

In this chapter the design of a "pure" collaborative system will be defined in more detail. A brief survey of work in this area will follow, and it will be seen how many of the disadvantages ascribed to content-based systems can be overcome. The chapter will conclude with an analysis of the significant new problems encountered by the collaborative approach.

4.1 Design

Unlike the user profile vectors defined earlier, which exist in a space whose dimensions are all of the words of some document corpus, user profiles in a collaborative system exist in a space whose dimensions are all of the possible recommendable items of some collection. Let $C = \{c_1, c_2, \ldots, c_p\}$ be the set of all these items (so the elements c_i represent movies, music CDs, books, etc., depending on the application domain). Then a user profile $\mathbf{g} = [g_{c_1} g_{c_2} \ldots g_{c_p}]$ is a vector representing the user's ratings over these items. If a user gives a rating of +2 to movie c_i, entry $g_{c_i} = 2$. As for the vectors defined for documents, a missing value $\oslash$ is required to denote items not yet seen or rated. Just as simpler IR systems will retain only the presence or absence of a word rather than a weight, some collaborative systems will not represent ratings but just keep lists of preferred or liked or perhaps purchased items.

For a machine learning supervised classification task, the k-NN algorithm is often used (as described in section 3.1.3). Although it is a fairly easy matter to apply this algorithm to the collaborative filtering task, no such experiments are currently known to have been performed. However, the algorithm to be described here, variants of which are used by many collaborative filtering systems, does bear some resemblances.

For a user for whom recommendations are required, the k nearest neighbor users are found. Instead of a distance metric, a similarity metric is usually employed (most often the Pearson statistical correlation coefficient), with some precomputation to attempt to normalize the different rating habits different users might have. It is common to ignore negative ratings in this similarity calculation. The similarity metric is denoted by CORR. A "recommendation vector" $\mathbf{h}$ is created for a user $\mathbf{g}'$ which is the weighted sum of these nearest neighbor vectors:

$$\mathbf{h} = \sum_{\mathbf{g}_i \in \Phi} \mathbf{g}_i . \textsc{corr}(\mathbf{g}', \mathbf{g}_i) \tag{4.1}$$

where Φ is the set of the k nearest neighbor user profiles to $\mathbf{g}'$ using the same measure CORR.

Potential items to recommend are all those which a user has not seen before, i.e., every c_i such that $g_{c_i} = \oslash$ and $h_{c_i} \neq \oslash$. Usually a small number of the highest scoring items are recommended (the items c_i such that h_{c_i} has a high value compared to other elements of $\mathbf{h}$). The number of items to recommend is either fixed in advance or determined by a threshold.

4.2 A brief survey

Malone *et al.* [1987] proposed three kinds of information filtering activities: cognitive, economic and social. Cognitive filtering corresponds to what we have been calling the content-based approach. Economic filtering is based on estimated search costs and benefits of use, and is not considered in this thesis. The social filtering approach is the subject of this chapter.

The canonical "pure" collaborative method described above is believed to have been originally described in a US patent [Hey, 1989; Hey, 1991], although Microvox Systems, Inc. claim an earlier system—a touch-tone accessible dating service[1]. The academic literature was introduced to the term *collaborative filtering* by the Tapestry system [Goldberg *et al.*, 1992], which allowed users to create standing queries based, in part, on annotations by other users of the system. An envisaged scenario was finding the important documents on the Usenet newsgroup `comp.unix-wizards` by filtering on "documents replied to by Smith, Jones or O'Brien."

Several years later a number of academic systems were introduced using methods

[1]Readers interested in ongoing debates about the origins of collaborative filtering are referred to exchanges on the `collab@sims.berkeley.edu` mailing list, currently archived at http://www.sims.berkeley.edu/resources/mailing-lists/collab/

similar to that described above, and to Hey's patents. Domains included recommendation of music [Shardanand, 1994; Shardanand and Maes, 1995], videos [Hill *et al.*, 1995], movies [Fisk, 1996], Web pages [Wittenburg *et al.*, 1995; Rucker and Polanco, 1997], Usenet news articles [Resnick *et al.*, 1994; Miller *et al.*, 1997] and Web links mentioned within them [Terveen *et al.*, 1997]. The period from 1995 to the present (March 1998) saw the explosive growth of the Web, and the widespread commercialization of collaborative filtering technology. Some valuable insights can be gained by analyzing the domains where this technology has been deployed successfully. However, an unfortunate consequence is that little is known about the underlying algorithms. At the time of writing, companies with technology based on collaborative filtering include Firefly Network Inc., Net Perceptions Inc., Imana Inc., Wisewire Corp. and LikeMinds Inc. An example of a high-profile successful deployment is the Amazon.com, Inc. on-line book store[2], where software from Net Perceptions Inc. (an outgrowth of the GroupLens research project [Resnick *et al.*, 1994]) provides recommendations of books. These are based both on records of books purchased by each user, and, separately, on optionally entered explicit ratings. In addition, recommendations are provided by human editors or book store staff, relative to topics of interest or "moods" rather than individual shoppers.

In a strand of research somewhat separate to these consumer applications, systems have been developed to help with locating colleagues or information within organizations. ReferralWeb [Kautz *et al.*, 1997] creates "social networks" using cooccurrence of individual's names on Web pages as evidence for the existence of relationships, and other systems require explicit communication of recommendations between users, e.g., [Maltz and Ehrlich, 1995].

[2] Currently at http://www.amazon.com

4.3 Comparing pure collaborative and content-based methods

The collaborative method described above in section 4.1 resolves a number of problems found with the content-based methods described earlier in section 3.5:

- Recommendations can successfully be made even for domains where automated content analysis is beyond current technology (such as restaurants, books, movies).
- Users can be recommended items dissimilar to those recommended in the past (assuming a breadth of interests within the user population), thus preventing the problem of over-specialization.
- Rather than requiring sufficient training examples to be able to identify items of interest, the system needs sufficient ratings to locate the user in an appropriate neighborhood of other users. Depending on the composition of the user population, the collaborative method has the potential to start providing satisfactory recommendations more quickly. In contrast, the learning speed of the content-based method depends on the nature of the incoming documents.
- Often the domain of a collaborative filtering system is such that the activity of requesting and reviewing recommendations is entirely separate from the consumption of the recommendations—for instance, when the recommendations are books, movies, restaurants or music CDs. In contrast, a typical text recommender system provides facilities both for requesting and reviewing recommendations, and for reading the recommended documents. In the latter scenario, having to supply ratings is often seen as a hindrance which distracts users from their primary task of viewing and reading documents. In the former scenario, it

is possible that users are more willing to interact with a recommender system as they browse through its recommendations, in much the same way as they might pleasurably browse in a book store (so a collaborative recommender system is directly supporting an existing activity). Part of this distinction stems from the fact that books, movies, music CDs and restaurants are all "experientially larger" items in the sense of requiring time to be "consumed," and being recognizable and memorable just from their names. On the other hand, individual news articles, as conduits for underlying news stories, would not create such strong impressions.

However, the collaborative approach does introduce certain problems of its own:

- If a new item appears in the database there is no way it can be recommended to a user until some more information about it is obtained, through another user either rating it or specifying which other items it is similar to. Thus, if the number of users is small relative to the volume of information in the system (because there is a very large or rapidly changing database) then there is a danger of the coverage of ratings becoming very sparse, thinning the collection of recommendable items.

- For a user whose tastes are unusual compared to the rest of the population there will not be any other users who are particularly similar, leading to poor recommendations.

- The last two problems depend critically on the size and composition of the user population, which also influence a user's group of nearest neighbors. In a situation in which feedback fails to cause this group of nearest neighbors to change, expressing dislike for an item will not necessarily prevent the user from receiving similar items in the future.

- The lack of access to the content of the items prevents similar users from being matched unless they have rated the exact same items. For instance, when Web pages are being recommended, if one user liked the CNN weather page and another liked the MSNBC weather page, the two would not necessarily end up being nearest neighbors.

The following chapter will introduce ways to overcome these problems.

Chapter 5

Multiagent Hybrid Recommendation

In this chapter the "Fab" multiagent architecture will be introduced. The design was motivated by two goals:

- To combine content-based and collaborative recommendation to eliminate the disadvantages of each;
- To take advantage of the overlapping interests of users both to improve recommendations and to gain efficiencies of scale.

Fab was designed and implemented during 1995, and opened up to a small number of users from January 1996 (at http://fab.stanford.edu). The final version, which will be the one described in detail, was unveiled in January 1997, and shortly afterwards opened to "surf-up" users from the general public. The number of regular users peaked at around 100, but by April 1997 the decision was made to restrict access in order to concentrate on more focussed studies.

As in the earlier LIRA system, we assume that users see a small number of recommended Web pages on each iteration, and provide explicit feedback. The 7-point scale

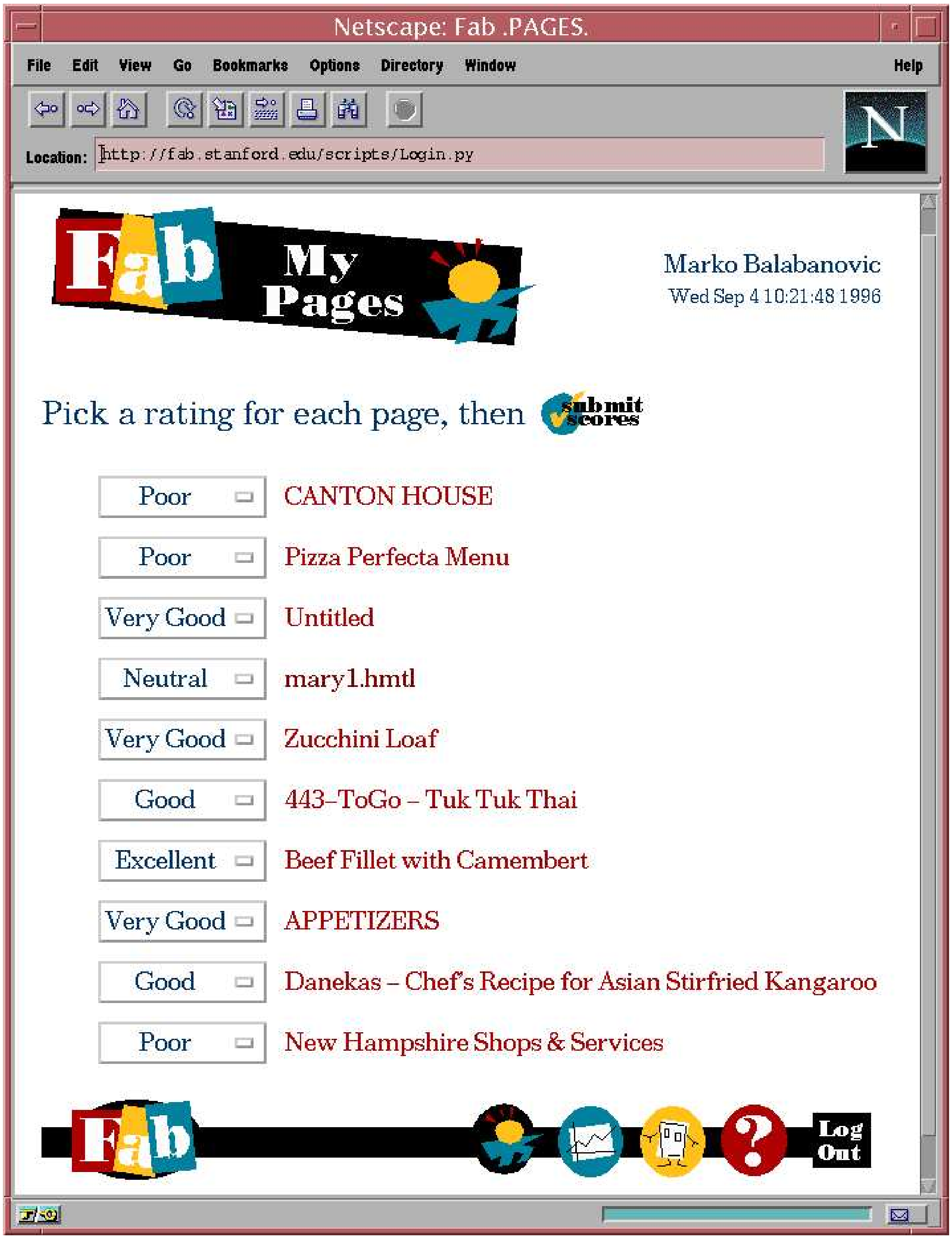

Figure 5.1: The Fab interface

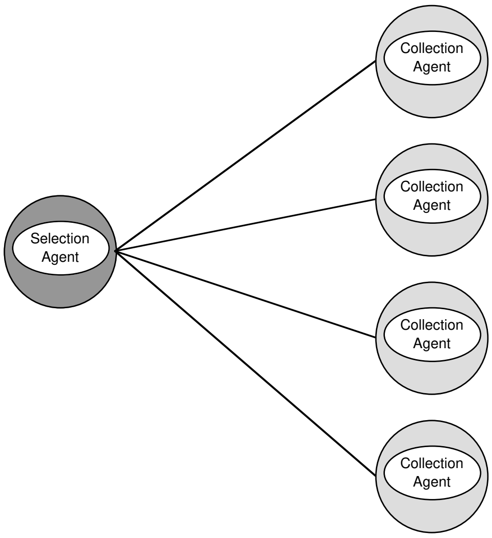

Figure 5.2: Hypothetical system design: single user, many collection agents

of Figure 2.3 is used. Figure 5.1 shows an example screenshot of the Fab interface, as seen in a Web browser, for a user interested in cooking and recipes.

5.1 Introduction: Two "thought systems"

In Chapter 3 an architecture was introduced with a strict decomposition of one agent per user, although the agent was partitioned into selection and collection components. As a gradual introduction to the extensions of this chapter, consider the following two "thought systems:"

Single user / Many agents Figure 5.2 shows a system where there are many collection agents serving a single user, who is represented by a single selection agent (analogous to a situation where several journalists submit stories to a single editor). Each collection agent has a search profile vector representing the topic for which it is

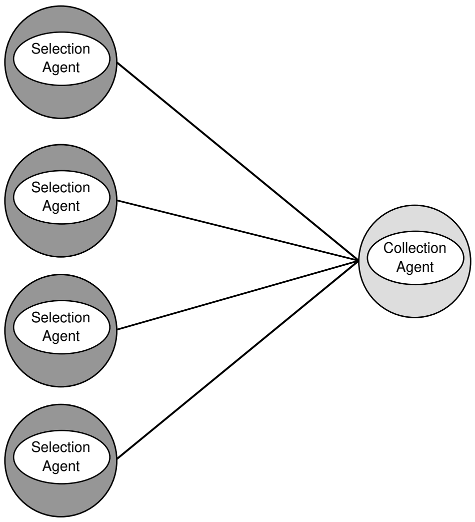

Figure 5.3: Hypothetical system design: many users, single collection agent

searching. Thus the user's separate topics of interest can be served more effectively, and the workload is distributed among several processes. In addition, the selection agent maintains the user's personal profile.

Many users / single agent In contrast, Figure 5.3 shows a scenario where many users (each represented by a selection agent) are served by a single collection agent. This collection agent will, over time, adapt to serve the needs of some average of all of the users. Thus problems of scaling are overcome, since only one search process is required. However, the pooling of feedback from all of the users means that, although faster, the personalization is less personal. The situation is analogous to a single journalist submitting stories to several editors (one per user). Both the collection and selection agents have separate profiles, representing the pages being sought and the user's interests, respectively.

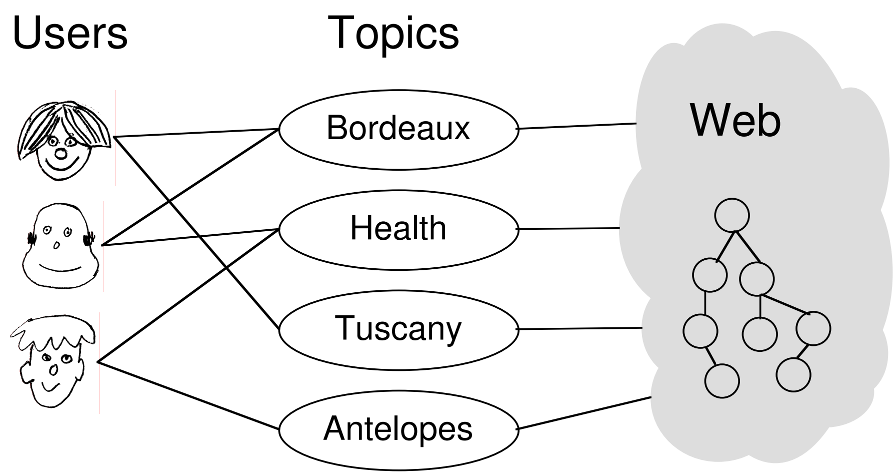

Figure 5.4: Model underlying system design: many-to-many mapping between users and topics

5.2 Design

The actual multiagent design combines aspects of both of these hypothetical scenarios. The underlying model is shown in Figure 5.4—the collection stage gathers pages relevant to a small number of topics, computer-generated clusters of interests which track the changing tastes of the user population. These pages are then delivered to a larger number of users via the selection stage. One topic can be of interest to many users, and one user can be interested in many topics. The equivalent agent architecture is shown in Figure 5.5. Each user is served by many collection agents, and each collection agent serves many users. Or, to complete the earlier analogy, there are many freelance journalists and many editors. Each story a journalist submits will be used by the editors who find it most interesting. As before, both collection and selection agents maintain their own profiles. A collection agent's profile represents its

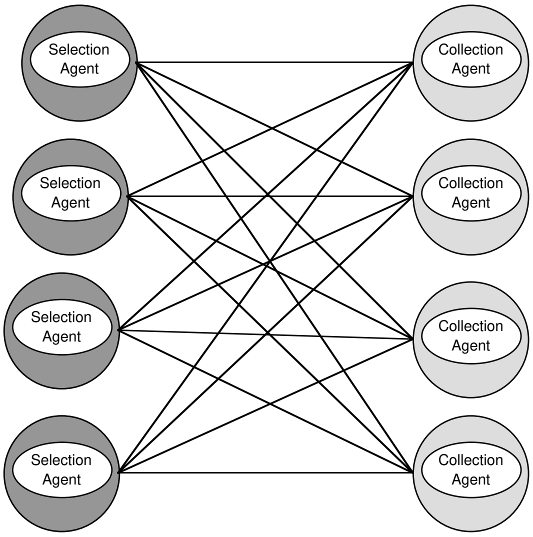

Figure 5.5: Selection and collection agents

current topic, whereas a selection agent's profile represents a single user's interests.

The system further generates an *amalgamated profile*—an average of all of the selection agent profiles, representing the interests of the population of users as a whole. This allows an "average user" to be automatically generated and maintained, which proves useful both for comparisons and for various services to be outlined below.

The scheme of Figure 5.5 would in fact lead to needlessly complicated communications, and in practice is simplified with the addition of a single *central router* as shown in Figure 5.6. The architecture is best explained by considering the path of a document through the system. Let's say the document is found by agent A, one of the collection agents. At regular intervals, collection agents submit to the central router the pages they have gathered that best match their search profiles. Those pages that best match each user's personal profile are then forwarded by the router to that user's selection agent. The user then rates these pages. The ratings are used

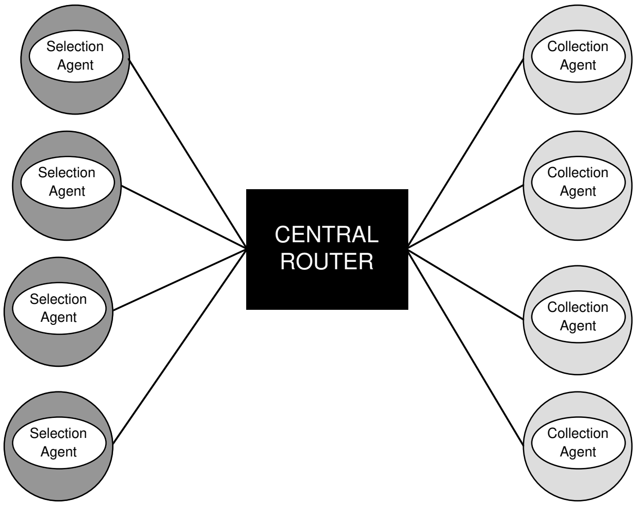

Figure 5.6: Selection agents and collection agents linked via central router

as feedback for the agents to learn from: the selection agent uses the feedback to update the user's personal profile. It also forwards the feedback, via the central router, to the originating agent A, which will update its search profile in the same way. This adaptation process will now be described in more detail.

5.2.1 Selection agent adaptation

The selection agent receives a stream of articles from the central router—its job is to present small sets of recommendations to the user, on demand. In order to do so, it maintains a copy of the user's personal profile, as well as a history of pages previously recommended (restricted to the past 30 days in the Fab implementation). The resulting document selection strategy is to pick the most topically relevant documents, relative to the user profile, with the following two provisos as in the LIRA system:

1. There is at most one page from any Web site (or more precisely, from any Web server running at the same IP address).

2. Pages previously recommended are not shown again. The matching is done based on vector signatures (section 2.1) rather than URLs, to better take into account replicated pages or those accessible from multiple Web servers.

The user's individual profile is updated given his or her ratings, using the relevance feedback update rule (equation 3.1). The user's feedback represents a significant investment in time and effort. By storing it in their own private selection agent's profile, we insure that it can never be "drowned out" by other users' feedback, and in fact it is easily exportable for use in other applications.

In addition to the relevance feedback, there is a further update applied to the profile every day. A user's interests change over time, and it is assumed that this happens in real time, regardless of the number of recommendations requested per day. This process is modelled using a simple decay function—each night all the weights in the profiles are multiplied by 0.97 (so after 23 days, the weight of an unreinforced word would have halved).

5.2.2 Collection agent adaptation

One of the main aims of the architecture is to encourage specialization among the collection agents, to the areas of overlapping interest in the user community. There are two separate learning processes involved, analogous to individual learning and evolution in genetic algorithms (although the Lamarckian form of evolution to be described is not used very often compared to Darwinian evolution).

Individual agent learning

Each collection agent receives feedback only on those pages it originally discovered, and then only if they are actually seen and rated by a user. This feedback is incorporated into its search profile using the relevance feedback update rule. A good analogy is with a shopkeeper, who receives feedback only from customers who choose to buy things in the shop. Dissatisfied customers are unlikely to return to make further purchases, whereas satisfied customers will return regularly. Over time the goods sold will become increasingly tailored to the desires of the shoppers who continue to visit. Nevertheless, there is no permanent linkage between shops and shoppers. In the same way, users who give negative feedback for a particular topic being served by a collection agent are less likely to be shown such pages in the future, and so less likely to influence the direction of that collection agent significantly.

Evolution of agent population

The specialization of individual agents as described will not necessarily lead to the best allocation of resources, as the topics served might not be those of maximal interest to the community. There is also a need for a longer-term evolution of the agents.

Collection agent performance is monitored in two ways. The appropriate measure of central tendency for ordinal variables is the median. The *success* of an agent, therefore, is the median of all scores received. The scale is the same as the scale used by users to rank documents, namely the scale shown in Figure 2.3. In addition the composition of pages shown to users is monitored. The *popularity* of an agent is the average percentage of its pages in users' recommendation sets.

For a particular agent to be clearly superior to another agent, it must have a median score that is consistently higher, and it must supply a higher percentage of the users' pages. An agent with a low popularity but a high success score is a *niche* agent, successfully specialized to a narrow topic. This is analogous to a high precision/low recall situation in a regular IR system. The case of high popularity but a low success score should be transient, as it indicates that the user profiles do not match the users interests well. Therefore it is sufficient to decide upon agent performance using just the success measure.

At regular intervals (weekly, for the Fab implementation), the following procedure is followed:

1. Find best and worst agents over last week, according to their success rating.

2. If neither of them received a positive success score, restart the worst agent from scratch (i.e., with an empty profile).

3. If the worst agent received a negative success score, but the best agent received a positive one, duplicate or split the best agent and kill the worst.

The aim is to allocate the most resources to the topics of most interest to the user population. Thus if there are insufficient resources, a consequence of this scaling-up scheme is that those users with unusual interests will not be served well.

Note that two options are listed for dealing with the best agent: duplication or splitting. If an agent is duplicated, its two clones are guaranteed to receive different feedback (since each user sees a page only once, and feedback only goes to the originating agent). Therefore the clones will diverge (to the extent that user interests diverge). An alternative is to split the agent along semantic lines by applying a 2-way clustering method to the user profile. This will be discussed in more detail in the following section.

More explicitly biological processes such as crossover and mutation are normally used in genetic algorithms, where the population of individuals (corresponding to our collection agents) would typically be much larger. In comparison with our system, where user feedback determines fitness, there is not usually such a high cost associated with evaluating an individual. An additional difference is that the Darwinian model of evolution dominates, although recent research has illustrated advantages of the Lamarckian model [Grefenstette, 1991; Ackley and Littman, 1992].

Two behaviors are hypothesized for this design for a collection agent adaptation scheme:

Specialization By only giving feedback to the collection agent that originally found a page, we are attempting to encourage specialization. If an agent begins to specialize to a topic, users not interested in that topic should no longer be shown pages from that agent, which should narrow its specialization further. Eventually we expect the agent will settle into a niche where there is a group of users interested in the topic that its profile represents. Thus the overlaps between users' interests can lead to an economy of scale, whereby several users interested in one topic can be served by a single agent, and conversely users interested in several topics can be served by several agents.

Serendipity If **any** collection agent finds a page that matches a user's profile, the user should see it. This includes agents that have specialized to finding pages for other groups of users, agents that monitor various existing Web page recommendation services, agents that merely produce random pages, etc. This property can be thought of as allowing for serendipitous recommendation, whereby users are shown pages from outside of their regular sources which happen to match their interests.

One of the aims of the experiments described below is to find out whether these behaviors are indeed exhibited.

5.2.3 Profile splitting

In the previous section, a method for implementing collection agent evolution by splitting the best agent profile was mentioned. That method is described in more detail in this section.

The usefulness of spectral methods in IR has been well known for some time. Singular value decompositions of word/document matrices for LSI have proven valuable both for improving retrieval performance and for reducing dimensionality to allow machine learning methods to be applied [Deerwester *et al.*, 1990]. Decompositions of word/word cooccurrence matrices were studied in the early days of IR research, in attempts to reproduce human-generated classification schemes [Borko, 1962].

An unusual aspect of the problem of splitting a profile is that documents are not involved, only words. Furthermore, since a 2-way split is required, it is not necessary to undertake the difficult task of estimating how many clusters are best.

As well as updating their profiles given feedback, collection agents maintain a word cooccurrence matrix Q. Each entry q_{ij} is incremented by 1 every time words t_i and t_j cooccur in a document for which the agent received feedback (more precisely, the cooccurrence is not over all the words in the Web pages, but only those represented in the document vectors). Initially, $q_{ij} = 0$ for all i, j.

One option, inspired by LSI, is to cluster the words in the profile by simply finding the largest eigenvector of Q, which represents a dimension along which the words would be widely spread. One-dimensional clustering is just a matter of picking an appropriate point along that line.

However, following from the observation that the cooccurrence matrix is equivalent to an adjacency matrix for an undirected, weighted graph, it transpires that readily available *graph partitioning* software can do a reasonable job of splitting the profile in near real time.

The problem of graph partitioning is to partition the vertices of a graph so that each partition is of approximately the same size, while minimizing the number of edges connecting the partitions. In the weighted case, we insure that the partitions are of approximately equal weight (by summing the weights of the vertices) and minimize the total weight of the edges connecting partitions.

Since this is an NP-complete problem, algorithms have been proposed that employ a number of heuristics to arrive at an approximate solution in linear time. The most frequently quoted application is balancing computational load over parallel processors for complex simulations. Graph partitioning algorithms have not previously been applied to IR-related tasks, to our knowledge.

The CHACO 2.0 graph partitioning software package [Hendrickson and Leland, 1994; Hendrickson and Leland, 1995] has been made available by Sandia National Laboratories for academic use[1]. It works using a multilevel algorithm. First, a series of increasingly coarse approximations to the graph are generated. Second, a spectral method is used to partition the coarsest graph. Finally, this partition is projected back through successively finer graphs. At each stage local refinement is performed, moving selected vertices using a variant of the Fiduccia and Mattheyses improved version of the Kenighan and Lin algorithm [Fiduccia and Mattheyses, 1982; Kernighan and Lin, 1970].

When applied to a cooccurrence matrix Q, a graph partitioning algorithm, suitably parameterized for a 2-way clustering, will produce two sets of nodes (which correspond to words), such that the sets are roughly of equal size and the edges between them are minimized (i.e., there is little cooccurrence of words in separate clusters). Given a profile $\mathbf{m}$ and the cooccurrence matrix Q representing the documents used to build $\mathbf{m}$, a splitting method to create $\mathbf{m}', \mathbf{m}''$ is:

1. Run a 2-way graph partitioning algorithm on the graph induced by interpreting

[1]Licensing details currently at http://www.cs.sandia.gov/CRF/chac.html

matrix Q as an adjacency matrix, creating two sets of words C_1, C_2

2. For each word t_i in C_1, set $m'_{t_i} = m_{t_i}$, and $m''_{t_i} = \oslash$

3. For each word t_i in C_2, set $m''_{t_i} = m_{t_i}$, and $m'_{t_i} = \oslash$

The above provides an example implementation for splitting collection agent profiles. In the public versions of Fab, and in the experiments to be described, the simpler duplication method was used.

5.2.4 Collection agent types

So far only collection agents performing a beam search of the Web have been described (section 3.1.1); in actuality the architecture described makes it easy to experiment with many kinds of agents. The following types have been created:

Search Agents As already described, these agents perform a beam search of the Web. For each agent the heuristic function used to evaluate a Web page $\mathbf{d}$ is $\textsc{sim}(\mathbf{m}, \mathbf{d})$, where $\mathbf{m}$ is the agent's search profile.

Index Agents Rather than actually searching the Web, these agents attempt to construct queries for existing Web indexes in an attempt to avoid duplicating work. The indexes used were Alta Vista, Inktomi (now called HotBot) and Excite[2]. Since little information about the IR models employed by these commercial systems is forthcoming, it is hard to design optimal queries. Furthermore the indexes are tailored mainly to short queries (justifiably, given that, for example, the average length of a query to the Alta Vista index is under 2 words [Monier, 1996]). Entering more than 20 search terms often results in zero recall. Thus the current implementation is fairly

[2] Current URLs are http://www.altavista.digital.com, http://www.hotbot.com and http://www.excite.com, respectively.

rudimentary: submit as a disjunctive query the top q words from the agent profile, where q has been optimized by hand for different indexes, and varies from 10 to 20.

Non-Adaptive Agents For purposes of comparison some simpler agents, which do not maintain their own profile, have also been implemented:

NO-MEMORY INDEX AGENTS These agents work exactly like regular index agents, but they draw their words from the amalgamated profile. Thus these agents are not trying to specialize but are serving the "mass market."

RANDOM AGENTS These agents retrieve pages from various sources of random pages available on the Web (i.e., they are not truly randomly picked pages, but rather randomly picked from various preselected collections). Pages were drawn from Alta Vista, Yahoo, URouLette[3].

COOL AGENTS These agents retrieve human-recommended pages from nine "cool page of the day" sites around the Web. These pages have been selected for their interest to the general community, not any particular user, and are often newly released sites.

NEW-YAHOO AGENTS These agents retrieve all of the new pages added to the Yahoo hierarchy on any given day, providing a somewhat edited supply of newly created sites.

5.2.5 Including explicit collaboration

A further refinement of the architecture is to include a phase of explicit collaboration among the selection agents, reflected by the added connections of Figure 5.7. Whenever a user rates a page above a certain threshold score (which could depend on the

[3]At the time of writing, the Alta Vista service appears to no longer be available, and the others are at http://random.yahoo.com/bin/ryl and http://www.uroulette.com.

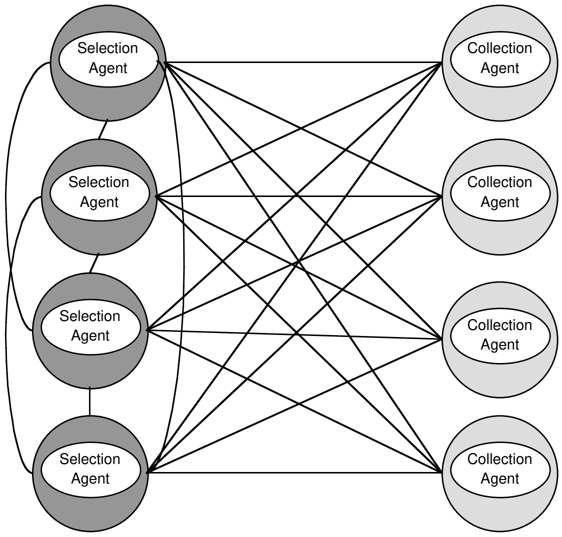

Figure 5.7: Selection agents and collection agents showing explicit collaboration links

user's previous scoring habits, but in the Fab implementation was fixed as being any positive rating), the page is directly routed to that user's nearest neighbors. The similarity between users is calculated using the SIM measure, except that negative portions of the profiles are ignored. As is common with collaborative filtering systems, deciding on user similarity based on shared dislikes is avoided as being overly broad. The number of nearest neighbors considered was fixed—to be a neighbor, a user had to be among the closest 10 users.

For example, a user interested in wine might be recommended a document with a map of a vineyard if another user with a similar profile rated the document highly. If the document contained little or no parseable text, it is unlikely to have been recommended by a regular content-based approach. Note that the ability of users to provide new examples to the system, for instance by rating particularly interesting pages found while browsing, significantly increases the chances of such a scenario occurring.

5.2.6 Advantages of the architecture

The architecture described exhibits certain advantages over pure content-based or collaborative methods due to its hybrid content-based/collaborative nature:

- By making collaborative recommendations, other users' experience can be used as a basis rather than the incomplete and imprecise content analysis methods available.
- By making content-based recommendations as well, items unseen by others can also be recommended.
- The profile built from the content of items is used to make good recommendations to users even if there are no other users similar to them. Furthermore, items similar to those disliked in the past are filtered out.
- Collaborative recommendations are made even between users who have not rated any of the same items (as long as they have rated *similar* items), extending the reach of collaborative systems to include databases that change quickly or are very large with respect to the number of users.

Additionally, the adaptation of the collection agents, particular to the Fab architecture described, enables some features impossible with either the pure collaborative or content-based approaches alone:

- A smaller number of collection agents than there are users can be instantiated, perhaps even a fixed number. This allows the system to scale gracefully as the numbers of users and documents rise. The exact number of collection agents required is determined by several factors, including the extent of the overlaps between users' interests and the tradeoff between the available computing resources and the quality of recommendations required.

- The collection agents automatically identify emergent communities of interest, allowing the support of social interactions between like-minded people and the automatic provision of group as well as individual recommendations. Effectively, like-minded users are pooling their resources, as each collection agent will be receiving feedback from all users interested in a topic.

- A brand new user to the system is shown a random selection of pages, from those arriving at the central router. However, the router receives pages that various agents believe will best match the current user population. Thus the new user is already starting from a much higher level than would be expected from an empty profile, especially if the system is deployed in an organization or special interest group where there will be significant overlap between users' interests.

It can be seen that by allowing recommendations to be made as a result of either the content-based or collaborative methods, the particular disadvantages of using either approach alone (described in sections 3.5 and 4.3) are overcome.

Furthermore, it is possible to cheaply maintain a population of "parasitic" users. These users have selection agents maintaining their personal profiles, but their feedback does not influence the collection agents. Taking even less resources, it is possible to support users who do not supply feedback at all. These "parasites" view pages preselected by an established group of users—in other words, pages best matching an amalgamated profile.

5.2.7 Related work

Two research efforts share many of our own goals. The Amalthaea architecture [Moukas and Zacharia, 1997], publicly described at about the same time as Fab, has many similarities. It also has a population of collection agents, although they adapt to serve a single user. An earlier system also from the MIT Agents group [Sheth and Maes, 1993] used techniques from artificial life to evolve a population of agents similar to our collection agents, although again each user was served independently. The InfoSpiders system [Menczer, 1998] concentrates more on the discovery of documents in a hypertext structure, given particular queries, by using a population of agents controlled with artificial life techniques.

Systems based explicitly on economic principles have also been proposed (e.g., [Mullen and Wellman, 1995; Karakoulas and Ferguson, 1995]), although there the research focus tends to be understanding properties of complex dynamic systems.

An architecture similar to ours has been described in [Höök *et al.*, 1997]—however, there the job of the collection agents is performed by human experts.

Researchers from the collaborative filtering community have attempted to overcome some of the problems of the pure approach as described in section 4.3. For instance, the Ringo music recommendation system [Shardanand and Maes, 1995] makes use of a limited number of features (such as the name of the performing artist). Systems that recommend documents of various kinds have been designed in order to ensure greater overlaps in items seen by users. So GroupLens [Resnick *et al.*, 1994] performs matching in the context of regular readers of Usenet newsgroups, Siteseer and GAB do so by comparing users' bookmark folders [Rucker and Polanco, 1997; Wittenburg *et al.*, 1995].

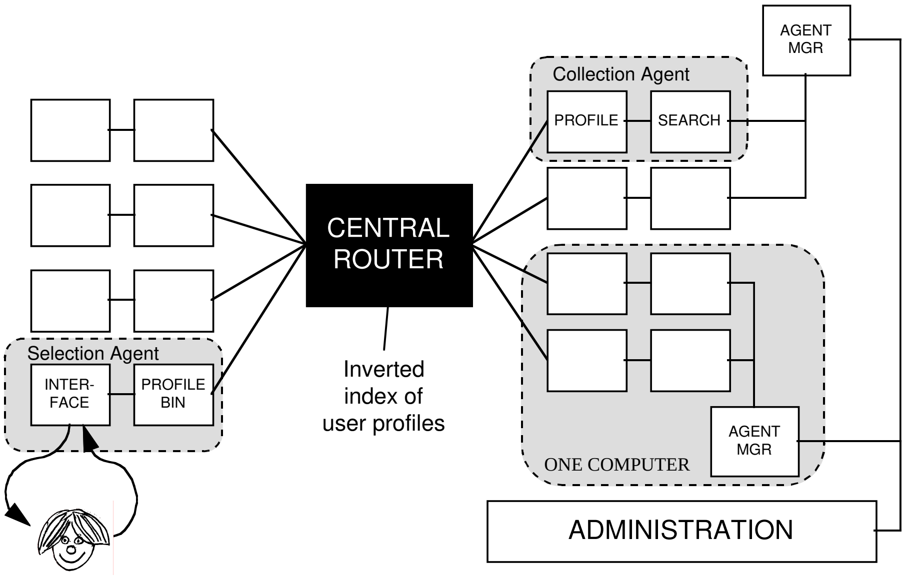

Figure 5.8: Components of implemented Fab architecture

5.3 Implementation

A number of design/test iterations revealed user requirements that had to be met by the implementation, the most important of which was fast access to a set of recommendations, and preferably to several subsequent sets. Therefore tradeoffs were decided in favor of response time rather than storage space or processing time.

Figure 5.8 shows an implementation-level view of the architecture. It consisted of

URL *(canonicalized)*
Site *(IP address extracted from URL)*
Title
Vector $\mathbf{d}$
Signature $\hat{\mathbf{d}}$

Table 5.1: Document token data structure

a number of Python processes, distributed over several Sun workstations and communicating over the HTTP protocol with basic session authentication (more precisely, the CGI protocol [Robinson, 1996] was used for requests, and responses were sent as Python marshalled files). The *document token* data structure, shown in Table 5.1, was used for transmission of document information. The following sections describe the functioning of the different kinds of agents and the central router in more detail.

5.3.1 Collection agents

Each participating computer has an `AgentManager` process, which runs hourly and communicates with a central administration server to find out which agents needed to be run on that computer at that time. A Web interface to the administration server allows a human administrator to allocate agents to computers—for instance, by specifying that a search agent should run from 1am until 9am on a particular machine. From an administration standpoint, agents are either atomic or continuous. Atomic agents execute a fixed set of instructions and then terminate. Continuous agents continuously perform their task (for instance, a Web search) until terminated by an external signal. The hourly `AgentManager` deals with both kinds of agents, restarting continuous agents in the event of problems (e.g., a machine re-boot). Certain problems are considered sufficiently severe to merit sending email to the human administrator, for instance a disk partition becoming too full.

Collection agents are implemented in Python and C, and use the mSQL relational database package [Hughes, 1993] for reliable long-term storage. A central database (also maintained with mSQL) caches names of Web servers which disallow automatic "spidering" via `robots.txt` files [Koster, 1994]. No coordination among search agents occurs, so it is theoretically possible for several agents to request pages from the same Web server at the same time. In practice the small number of agents makes this unlikely, but the issue would have to be tackled for a larger implementation.

The collection agent interface consists of two main facilities:

- Submitting a list of document tokens to the central router, tagged with a `ReturnAddress` URL identifying the agent;
- Receiving a list of ⟨token, feedback⟩ pairs back from the router, at a program running at the given `ReturnAddress`. This feedback is used to update the collection agent's profile, independent of any process that could concurrently be conducting a search. Long-running searches, e.g., the Web beam search processes, will check for updates to their search profile at regular intervals, and re-order their data structures accordingly. The reason for the asynchronous nature of this update is to allow batching of feedback from several users, if it arrives nearly contemporaneously.

5.3.2 Central router

The central router accepts submissions from collection agents via a well-known URL. The submissions are lists of document tokens tagged with a return address as specified above.

The main function of the central router is to act as a communications hub. It maintains an inverted index of user profiles (i.e., a mapping from words to the user profiles in which they occur, along with associated weights). Thus incoming document tokens can rapidly be dispatched to those selection agents where the match is above a threshold (usually the threshold was set to a very conservative value of 0, but higher settings could be used to decrease network traffic). The advantages of this mechanism (rather than the more common inverted index of documents, against which user profiles are matched) were described in [Yan and Garcia-Molina, 1995], where its usage for the SIFT information filtering system was detailed. With the large user profiles built up over time by Fab, these advantages are even more telling.

The central router further accepts feedback from selection agents, in the form of $\langle$token, feedback, `ReturnAddress`$\rangle$ triples. These are simply forwarded to the given `ReturnAddress`, representing the originating collection agent.

The nearest neighbors for each user are computed using the router's inverted index by a continuously running process. The resulting lookup table of nearest neighbors is cached at the router, so that incoming feedback, in addition to being returned to the originating agents, can also be forwarded to other similar users. The tradeoff here is for speed over accuracy—the cached nearest neighbors table could be several hours out of date (the update process is given relatively low priority as it does not interact directly with the user, and so can take on the order of several hours to recompute the nearest neighbors table).

Additional functions of the central router are to maintain a cache of submitted Web pages (pages with coarse versions of associated graphics (created with lossy JPEG compression) were kept for 4 days, thus usually allowing users a more consistent response time—accessing a cached page after it has been purged causes a redirection to the original URL), to keep records for the purposes of compiling experimental results, and to generate the "front page" displayed at the public Fab site —this is a selection of articles chosen so as to best match the amalgamated profile. Thus, in a similar fashion to "Page One" of the Fishwrap personalized news system [Chesnais *et al.*, 1995], users see a selection of articles of interest to the whole community on their way to logging in to see their personal selections. In addition, curious non-users can get a sense of what is of interest to the community of users (the occasional offensive site appearing on the front page is manually removed if spotted!).

5.3.3 Selection agents

The final aspect of the implementation is the selection agents. For each user a "bin" is maintained to keep all of the pages arriving from the router. As a page arrives, its signature is compared to the user's history list (actually stored as a hash table on disk using the Berkeley "DB" package [Seltzer and Yigit, 1991], interfaced to Python's "shelf" data structure). Previously seen pages are immediately discarded. Otherwise, the score of a page relative to the profile is computed, and the page is entered into the bin, which is implemented as a fixed-length heap (containing 100 elements). Pages with insufficiently high scores to place in the top 100 are discarded. A regular process takes the contents of the bin and generates sets of 10 recommendations, checking that no two pages from the same Web site are included in any one set.

The only step remaining when the user requests their pages is to run a set of precomputed recommendations through a process which combines them with HTML templates to produce a finished Web page (using Python in conjunction with Lex-generated C code). Again, the tradeoff is for speed over accuracy—it is possible that the bin and the profile have been updated since the sets of recommendations were last generated. However, in informal questioning and from looking at complaints, response time was consistently seen to be the most important factor for users.

The page provided for the user is actually an HTML form, the submission of which directs feedback back to the selection agent. The document tokens are stored within the HTML as hidden fields (i.e., they are not seen in the user interface, but are communicated to the selection agent). Given this feedback, the selection agent immediately returns control to the user (before any processing is done), ensuring a consistently fast response time. The user can go on to request the next set of recommendations right away. Thus some fairly complex locking is required in order that the selection agent can simultaneously update the user profile and history list, forward the feedback back to the central router, and potentially deliver further sets of recommendations. Were the user to request more than 10 sets of recommendations in quick succession, the bin would be exhausted—at this point the selection agents asks the user to try back in a few hours. In practice this rarely happens.

Additional facilities allowed users to:

- Provide feedback on arbitrary Web pages, rather than just those recommended by Fab (this facility is not shown in the earlier user interface illustrated in Figure 5.1);
- View the top terms in their profile;
- View a chronological record of past recommendations.

Although inspection of the collection agents was also possible, this facility was never made publicly available.

Currently all selection agent functions are executed on a central server. However, there is nothing in the architecture precluding selection agents from running on the "client side," on users' personal machines. Such decentralization would improve response time for the user, and in addition would allow stricter privacy controls over user profiles. Note that the central inverted index could either be entirely bypassed, if users had sufficient processing power to deal with all incoming documents, or it could be used as a purely negative filter, representing a users' dislikes but not their likes. Presumably the negative component of a user profile is both less personal and less valuable to organizations distributing advertising or marketing. More complex schemes can be imagined to insure varying amounts of privacy. For instance, small groups of mutually trusting users could band together and present a locally amalgamated profile to the central router, restricting access to personal profiles to a local router controlled by the group (similar schemes can be imagined to anonymize mail order purchases, for instance). Further, feedback from imaginary group members could also be simulated locally, to further confuse the central router. Investigation of privacy issues for collaborative filtering systems is an interesting area for future research, but unfortunately one that is not addressed in this thesis.

The full implementation described consisted of roughly 10000 lines of Python code (not counting comments!) plus ancillary components in C, Lex as well as various HTML templates. Processes were typically distributed over 4 computers, ranging from a Sun SPARC-5 to a Sun Ultra-2. Regular users and administrators could access and control the system, respectively, from any Web browser. The decision to use HTTP protocols rather than a distributed object framework such as CORBA meant that it was very easy to add new computers to the system (at the most, each would have to run an off-the-shelf Web server), and furthermore that many different languages could be integrated due to the wide availability of HTTP libraries. In addition, in joint work with Tom Schirmer, Scott Hassan and Steve Cousins, a module was built interfacing Fab to the CORBA-based Stanford Digital Library testbed [Stanford digital library group, 1995], allowing users of the DLITE interface [Cousins *et al.*, 1997] to receive Fab recommendations and provide feedback. To facilitate interoperation, Fab user IDs were optionally associated with digital library *E-Person* user IDs.

5.4 Experimental methodology

For the experiments to be described, document vectors contained the top 100 weighted words from a document.

5.4.1 Profile accuracy

A profile accuracy test followed the methodology introduced in section 2.3.3. At regular intervals (every 5 iterations, with approximately one iteration occurring per day) the user was presented with a special list of documents randomly selected from those submitted to the router. The following text was also shown: "Your ratings of these special evaluation pages are used to assess how well the system is doing, and will not affect your profile. You may have seen some of these pages before."

5.4.2 Comparison of sources

A second test is a comparison of sources, following the methodology of section 2.3.3. System-recommended *personal* pages are compared to pages from three other sources: *random* pages, *cool* pages and *public* pages. Random and cool pages are found by random and cool agents respectively; public pages are those which best match the amalgamated profile (which, in terms of the architecture, acts as a special extra user to which pages are sent by the central router).

There has been criticism of collaborative recommendation services [Mitchell, 1995] that they have never been shown to do better than the "majority vote" rule—they would do as well just recommending the most popular items. This is not directly applicable in our case: as there is a constant stream of new items, no one item will necessarily get rated by many users. However the comparison between personal and public pages gives us a measure of how much difference the personalization makes.

5.4.3 Declared topics

Our most scarce resource is not computer speed or storage, but the attention and quantity of users willing to test the system. Thus we decided to try to combine the two experiments described into one.

The results which follow describe a combined experiment. Users were recruited using mailing list and newsgroup advertisements. Each user was asked to declare a topic of interest in advance, and rate pages according to that topic (to allow easier interpretation of the profiles learned). The topics chosen were: computer graphics and game programming, library cataloging and classification, post-industrial music, sports information and gaming, Native American culture, cookery, 60's music, hiking, evolution, Web development.

Every five rounds of evaluation the users were shown a selection of 16 pages, four each from the four sources as described. The relative rankings of the sources **and** the difference between user and system rankings were recorded.

5.5 Experiment results

For this series of experiments we ran nine best-first search collection agents for eight hours per day each, during which time they would inspect between 1,000 and 5,000 Web pages (depending on computer and network load). We also ran two of each type of index agent (once every 8 hours), two random agents (also once every 8 hours) and one cool agent (once a day). As well as the users participating in the experiment we supported around 30 parasitic users, whose feedback did not affect the collection agents. The results are shown from 25 evaluations, corresponding roughly to one month of usage. Nine users participated in the experiment, although only seven made it to the 15^{th} evaluation, only five to the 20^{th} and only two to the 25^{th}. All the profiles were started empty. Users were not given the opportunity to provide arbitrary examples to Fab, but could only rate the supplied pages. In addition, the explicit collaboration step of section 5.2.5 was deactivated.

Two sets of statistical results are presented, followed by anecdotal results, observations of user behavior and users' comments.

5.5.1 Profile accuracy

Figure 5.9 shows the average *ndpm* at each evaluation point. The gradual downward progression indicates that the selection agents are successfully learning increasingly accurate profiles of the users over time, as the distance between the predicted and actual rankings decreases. Indeed by the 25^{th} evaluation it is equivalent to a difference

tablespoon	2.95	sprinkl	1.95	tomato	1.58
teaspoon	2.44	saut	1.92	cup	1.51
onion	2.16	chop	1.92	stir	1.44
flour	2.13	parslei	1.92	preheat	1.37
minc	2.09	saucepan	1.79	pepper	1.34
garlic	2.06	sauc	1.71	parmesan	1.33
clove	2.00	butter	1.59		

Table 5.2: Top twenty words and associated weights from the profile of a collection agent specializing in cooking. Some of the word endings have been removed (e.g., "mince","minced" and "mincing" all become "minc") or altered (e.g., "parsley" becomes "parslei") as part of the stemming process.

between 16-item predicted and actual rankings of just a single item misplaced by two positions.

The difficulty of the topics chosen by users varies enormously. A very rough idea of this can be gained from a simple keyword search of the Infoseek Web index: at the time of the user tests (June 1996), 20558 documents matched "cooking" but only 158 matched "library cataloging classification". One of the advantages of the vector space representation is that we can inspect the learned profiles: 90% of the top 400 terms in the user profile learned for cooking are obviously cooking related (see Table 5.2 for a sample), compared to only 2% for library cataloging and classification.

5.5.2 Comparison of sources

Figure 5.10 shows how pages from the different sources performed relative to one another. As expected the personal pages improved as adaptation occur ed, and did significantly better than pages from the other sources. Given the small number of predeclared topics it is not surprising that the the cool pages were not much better than the random pages. The public pages would provide a more interesting comparison if additional users and arbitrary topics were allowed.

There was considerable variation in collection agent performance for individual

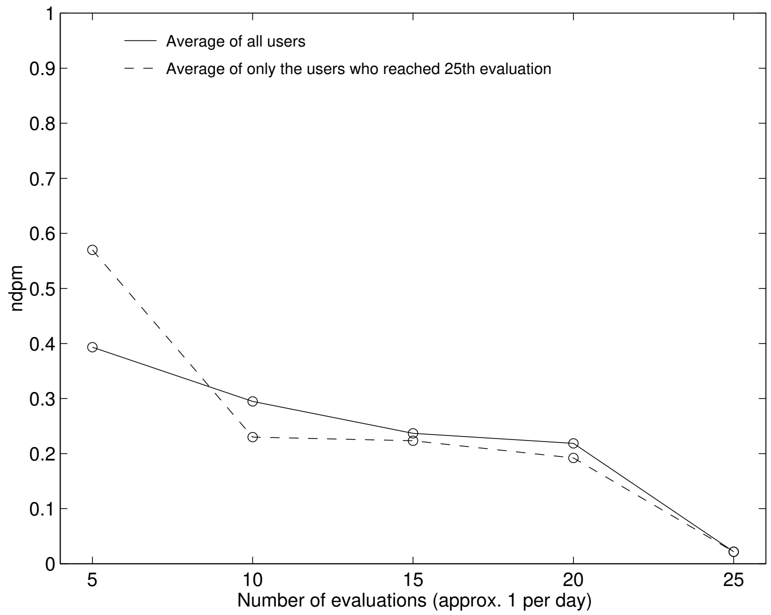

Figure 5.9: Distance between actual and predicted rankings, averaged at each evaluation point.

users. For instance the user interested in cooking received between six and ten pertinent documents per day after only four days, whereas the user interested in library cataloging, despite receiving many documents about libraries over the whole month, never received one on the exact topic.

A comparison of the success and popularity scores for search and index agents yields no discernible trends, with both classes of collection agent performing similarly. The index agents use negligible resources in comparison to the search agents, but we expect over a longer period that the finer adaptability of the search agents would be a dominant factor.

5.5.3 Specialization

In some cases it is very easy to see the specialization that has gone on among the collection agents—for instance an inspection of all the profiles reveals that one agent

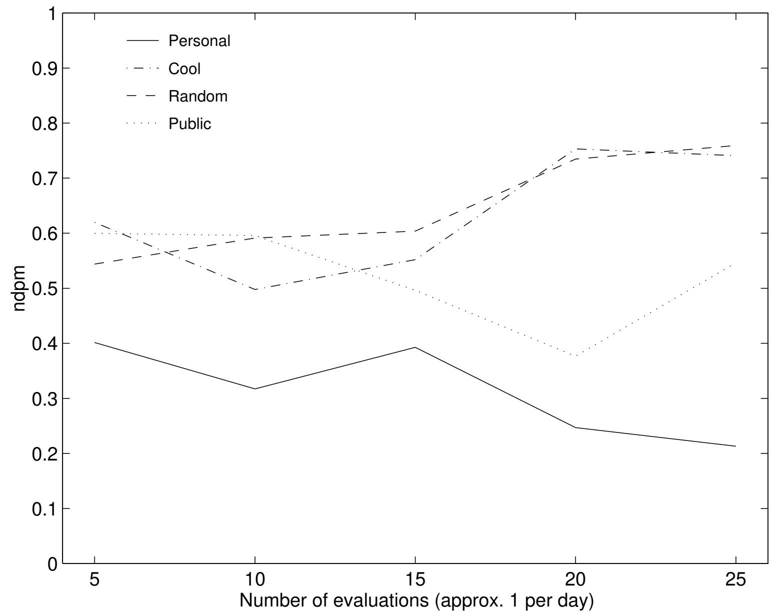

Figure 5.10: For each source, distance between user rankings and its ideal ranking, averaged over all users at each evaluation point.

is a "cooking expert," with 77% of the top 400 terms obviously cooking related, one other has a tangential interest and the rest have no apparent interest. Indeed the cooking user customarily receives between 50% and 90% of the recommendations from the "cooking expert" agent.

On the other hand "music" is a common theme for two of the topics, and this is reflected in the less strict specialization of the agents. Three agents have an approximately equal number of music-related terms in their profiles, and the two users interested in music-related topics typically receive their music-related pages from a mix of these agents.

5.5.4 Serendipity

An example of the serendipitous recommendations resulting from the interplay between selection and collection agents comes from a collection agent specializing in pages about India (mistakenly, as the intended topic was Native American cultures). Apart from attempting to serve the user interested in Native American cultures, at various times this agent provided pages on biodiversity in India to a user interested in evolution and recipes for Indian food to the user interested in cooking. Similarly, the users interested in Web development and computer graphics received pages on computing textbooks, relevant to both their topics.

These examples show that over time the agents can specialize to specific topics, and automatically converge to areas of overlap between the users.

5.5.5 User observations

One problem we encountered during this experiment was users being extremely strict, giving a high score only to pages exactly on their topic and a very low score otherwise. This leads to a lack of positive examples to learn from and subsequent fairly random behavior. The system works best when users "guide" it, by starting with a broad topic and slowly narrowing it down. Later systems avoid this kind of problem by using the already described mechanism allowing users to provide their own example pages.

An additional observation was that the real-time decay function turned out to be overly strict—users who logged in infrequently would find that their carefully trained agents would have forgotten all of their feedback! An alternative is to refrain from decaying a profile on days where no recommendations are requested, and furthermore to calculate the decay based on recency of word occurrence rather than using a simple multiplicative function. No experiments have been conducted to test these alternatives.

As well as the controlled experiment described, the Fab system was available for public use for a time. As with many implemented recommender systems, both commercial and academic, a common problem is that users are not willing to spend time to rate pages (e.g., see [Miller *et al.*, 1997]), and so after an initial flurry, usage levels drop. The next chapter will discuss methods for resolving this problem by using implicit feedback.

5.6 Summary: Exploiting overlaps between users interests

This chapter concludes investigation of the first research issue motivating the thesis: How to exploit overlaps between users' interests. Through introducing a series of increasingly sophisticated working prototypes, the advantages of hybrid content-based/collaborative systems have been revealed. In addition, it has been shown how an adaptive multiagent system can hone in on the overlapping interests between users, providing appealing scale-up properties.

Chapter 6

Composition of the Recommendation Set

The second major research issue motivating this thesis is how to decide upon the composition of recommendation sets, that is, how to select articles for a single edition of a personalized newspaper. This issue is largely beyond the scope of the well-studied core IR tasks, but is a key consideration for users of recommender systems.

The aim of this chapter's research has been to give the user more control over the document selection strategy and the tradeoffs involved in composing the recommendation set. The original motivations for this came from the Fab system described in the previous chapter. Each user's selection agent receives a stream of documents from both content-based and collaborative strategies. Each of these documents belongs to one of three categories:

- The selection agent is confident that the user will like it (i.e., it is similar to documents previously given a high rating).
- The selection agent is confident that the user will not like it (i.e., it is similar to documents previously given a low rating).

- The selection agent is not confident about its prediction (which could be either positive or negative).

Documents in the middle category can safely be discarded. Deciding between the first and third categories is the central issue, and it corresponds to the exploration/exploitation tradeoff. In the field of machine learning, this tradeoff is most commonly discussed in the context of reinforcement learning. The "k-armed bandit" provides the canonical example: an agent is in a room with k one-armed bandits (gambling machines where there is a probability of winning a prize each time their "arms" are pulled). On each turn, the agent can pull any arm, with no cost but missed opportunity. If the agent believes one arm has a high payoff, should it choose that arm all of the time (exploitation), or choose another for which it has less information (exploration)? Equating user satisfaction with payoff reveals this same tradeoff in the text recommendation task: should the system recommend pages which it believes the user likes, or recommend pages about which it has little information? This tradeoff does not occur for the usual IR tasks, since they do not require systems to select their own training data. In contrast, a recommender system performs active learning.

The first half of this chapter describes an implementation of an exploration/exploitation parameter, and a series of simulated experiments designed to test its properties. As well as being exposed in the public Fab interface from early 1996, the parameter is a building block for the system described in the second half of this chapter. The "Slider" interface allows users to specify multiple topics of interest, and control the proportions between them, and in addition includes an alternative view of the exploration/exploitation parameter. A parallel goal for Slider was to explore the use of implicit rather than explicit feedback. As a major interaction design overhaul, Slider incorporates many improvements suggested by Fab users over time, and has received a commensurably more enthusiastic response from test users in a short, informal usability study.

6.1 Exploration vs. exploitation control

As stated above, the goal of this section is to define an exploration/exploitation parameter, and to investigate its properties through a series of simulation experiments. These properties relate to both user satisfaction and system performance. Imagine a scenario where a user is interested in regularly receiving both documents about music and documents about literature. By random chance a recommender system might deliver a document about music in its first set of recommendations. If the user ranks this document highly, the system might then be so successful in learning this single topic preference that all documents subsequently delivered are on the topic of music. This over-specialization problem was introduced in section 3.5. Such a system is pursuing a purely exploitative document selection strategy, and by restricting its own training data is not learning about other interests the user might have. One reason for this is that a recommender system has a goal that is often in conflict with user satisfaction: that of learning a user model. By performing less exploration, always conservatively recommending documents that are similar to those for which the user has already expressed a preference, the system could be prolonging the time required to learn an accurate user model. The experiments in this section investigated the effects of an exploration/exploitation parameter on the learning rate of a recommender system and the composition of sets of recommended documents.

6.1.1 Design

Two different document selection strategies are investigated. The exploitative strategy is implemented using the SIM measure, as in earlier systems.

The exploratory strategy is formalized using a simple measure of *overlap* between a profile $\mathbf{m}$ and a document $\mathbf{d}$:

$$\textsc{overlap}(\mathbf{m}, \mathbf{d}) = |\{t_i \mid (d_{t_i} \neq \emptyset) \wedge (m_{t_i} \neq \emptyset)\}|$$

where $|.|$ denotes the cardinality of a set. Thus the overlap is the number of terms in common between those the user has seen and those contained in the document. It can be thought of as a measure of the certainty of the value of the similarity score. For instance, if the SIM value is derived from an overlap of a single word, it is probably less reliable than one derived from an overlap of 60 words. The exploratory strategy chooses those documents with the smallest overlap.

It is further possible to blend both strategies. A "mix" strategy includes both the most similar and least overlapping documents in a recommendation set, relative to the user profile. For instance, for a 6-document recommendation set, a 50%/50% mix strategy supplies 3 exploratory choices and 3 exploitative choices. The inputs and outputs of all of these strategies can therefore be represented as follows:

$$\{\succ_{\textsc{sim}}, \succ_{\textsc{overlap}}\} \longrightarrow \boxed{\text{Document Selection Strategy}} \longrightarrow R_{\mathbf{m}} \tag{6.1}$$

The exploration/exploitation parameter s is defined as the percentage of exploratory choices in the recommendation set. It is this parameter over which the user has control. In the Fab interface, it is presented to the user as a 3-position control (Figure 6.1), where the settings are labeled "broad" ($s = 100$), "balanced" ($s = 50$) and "narrow" ($s = 0$).

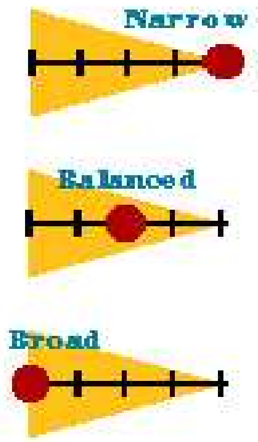

Figure 6.1: The three positions of the exploration/exploitation selector, as seen in the user interface.

6.1.2 Implementation

In this section the implementation of the exploration/exploitation parameter is explained the context of the Fab system. The simulated setting, used for the experimental results to follow, permits a much simpler implementation as there are no response-time requirements.

Whereas in the description of section 5.3.3 there is a single "bin" for each selection agent to hold incoming documents, the new design requires two bins. One is sorted according to SIM, the other according to OVERLAP (both relative to the user profile $\mathbf{m}$). Instead of a single set, the selection agent precomputes three sets of recommendations for each iteration, corresponding to broad, balanced and narrow settings specified by the user (although since the balanced set is a mixture of the other two, it is only really necessary to generate two sets per iteration).

The job of the central router is also slightly more complicated. As each new document token arrives, the inverted index of user profiles is used to calculate a SIM score against each user profile. Since this involves iterating over all of the words in the document vector, it is an easy matter to simultaneously calculate an OVERLAP score. However, since a **minimal** overlap is required, care must be taken to also include those users where the overlap is 0 (who for that reason would not normally feature in the SIM calculation). Document tokens can now be forwarded on the basis of overlap as well as topical relevance.

Note that the collection agents are unchanged from the designs presented in Chapter 5. However, the presence of RANDOM and NEW-YAHOO agents, for instance, guarantees a steady supply of exploratory choices.

The final aspect of the implementation is the interface. As shown in Figure 6.1, the user was presented with a 3-position control. Two different implementations were provided. For users of older browsers, the control consists of 3 separate bitmaps, each hyperlinked back to the selection agent. By clicking on different parts of the control, the user sends HTTP requests back to the selection agent to view the results of a different document selection strategy. Given that all three sets of recommendations have been precomputed, this communication scheme causes unnecessary delays. Therefore, for users of newer browsers (e.g., Netscape Navigator version 3.0 and above), a JavaScript version is provided. All three sets of recommendations are downloaded to the user's browser at once, although the user sees only one at a time. By clicking on different parts of the control, JavaScript functions are triggered which cause a different set of recommendations to be displayed, without requiring any HTTP communication.

6.1.3 Experiments

The simulated setting allows for the exploration/exploitation parameter to be isolated for analysis, in a way that would be difficult to replicate as a live experiment. The following experiments use the Yahoo corpus (Table 3.1) and the simulation methodology defined in section 2.3.3: four training iterations followed by a test iteration.

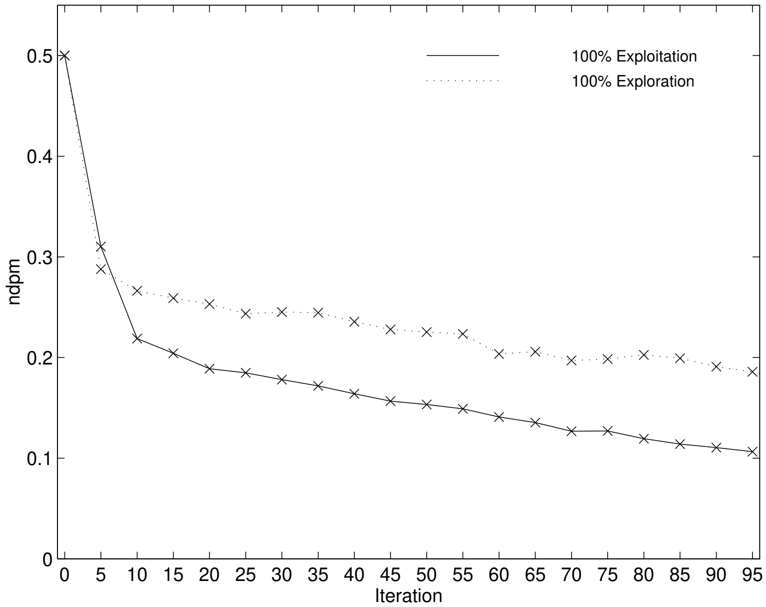

Figure 6.2: Exploitation vs. exploration document selection strategies for 1-pref users. Graph shows *ndpm* values averaged over all 10 possible users, measured at test iterations (every fifth iteration).

Users are simulated with n-pref structures as defined in section 2.2.5. The top 60 weighted words from each document form the document vector; the gradient descent update rule is used.

The particular hypothesis under investigation is that where a user has more than a single topic of interest, a more exploratory document selection strategy will lead to a faster learning rate for the user profile. However, a *purely* exploratory strategy will never recommend documents to the user similar to those preferred in the past (which seems hardly likely to please the average user).

Comparing document selection strategies

Single preference, or 1-pref, users provide the simplest test case for comparing document selection strategies. Figure 6.2 shows the learning curves over time for the pure exploitation and pure exploration strategies (this is a profile accuracy test, as

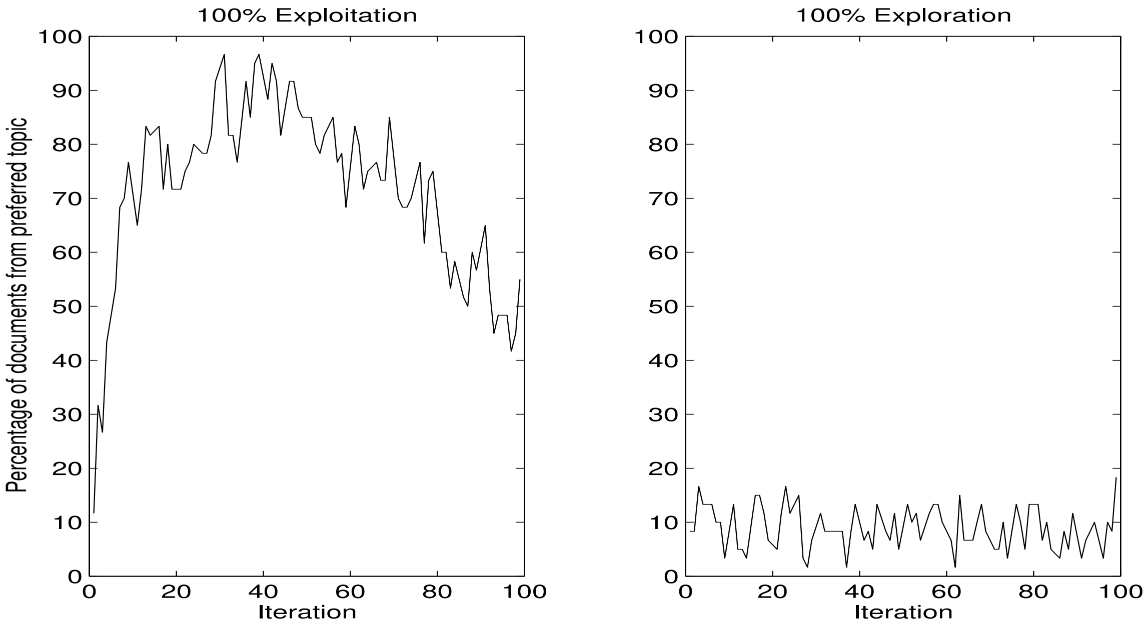

Figure 6.3: Composition of documents in recommended sets for 1-pref users: 100% exploitation and 100% exploration strategies. Recorded for every training step, and averaged over all 10 possible users.

explained in section 2.3.3). As expected, there is no benefit to exploration in this case, since over-specialization to one topic is exactly what is required. Figure 6.3 shows the composition of recommended documents resulting from the two strategies. As can be seen, a much greater proportion of documents come from the single preferred topic in the exploitation case, and so in all likelihood this would be the optimal strategy if a user really had such a simple preference structure. Note that even for the exploitation strategy, in later iterations the percentage of documents from the preferred topic drops—it becomes more difficult to retrieve relevant documents from the corpus as it is depleted of those in the preferred topic.

Learning rates for a population of 3-pref users are shown in Figure 6.4. As predicted, in this case an exploratory strategy leads to faster user profile acquisition, as a purely exploitative strategy is likely to deliver the majority of documents from a single topic. Indeed, Figure 6.5 shows that the composition of documents from a

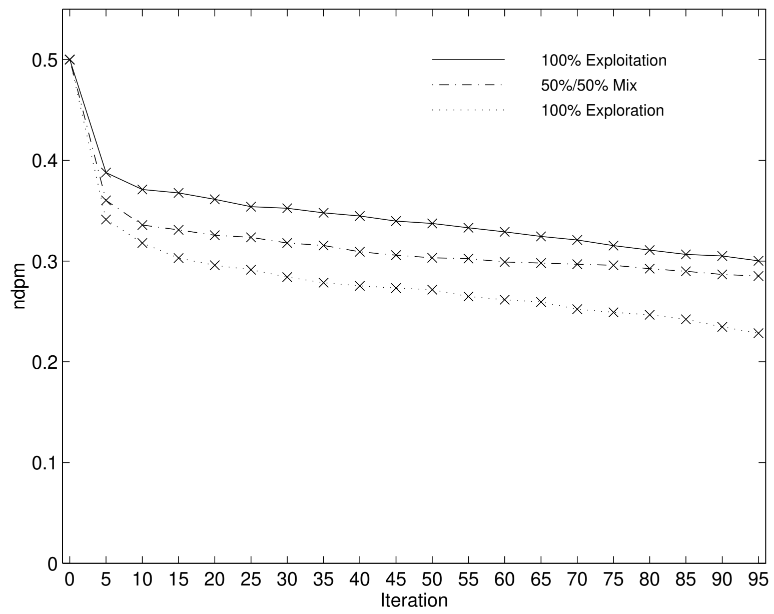

Figure 6.4: Exploration vs. exploitation document selection strategies for 3-pref users. Graph shows *ndpm* values averaged over 500 randomly selected users, measured at test steps every fifth iteration.

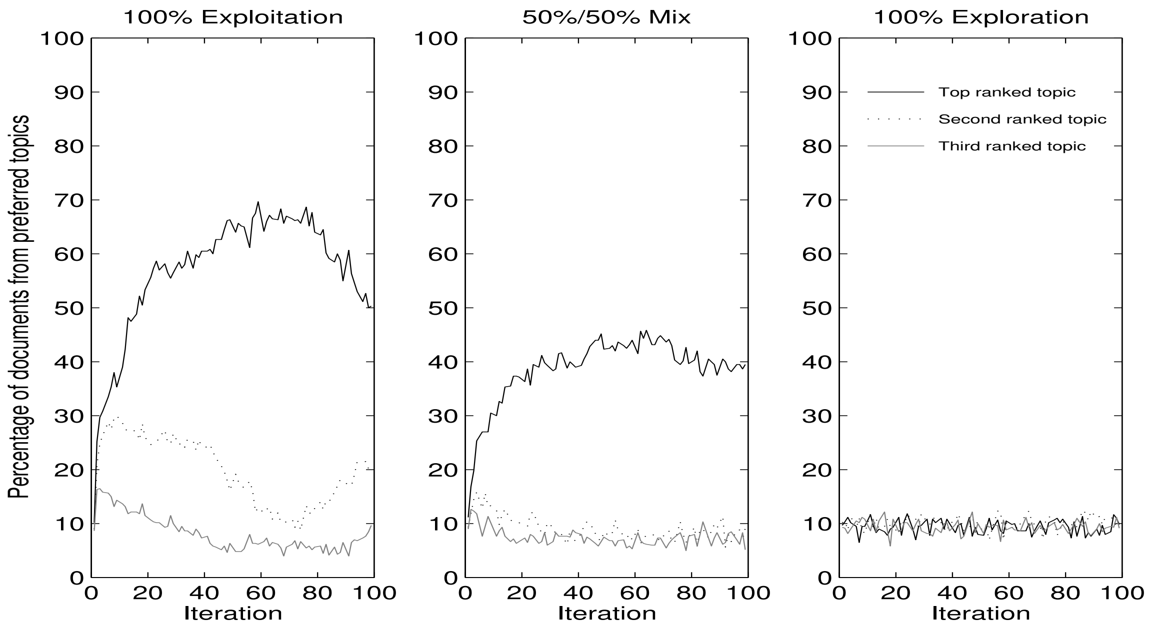

Figure 6.5: Composition of documents in recommended sets for 3-pref users: 100% exploitation, 50%/50% mix and 100% exploration strategies. Recorded for every training step, and averaged over 500 randomly selected users.

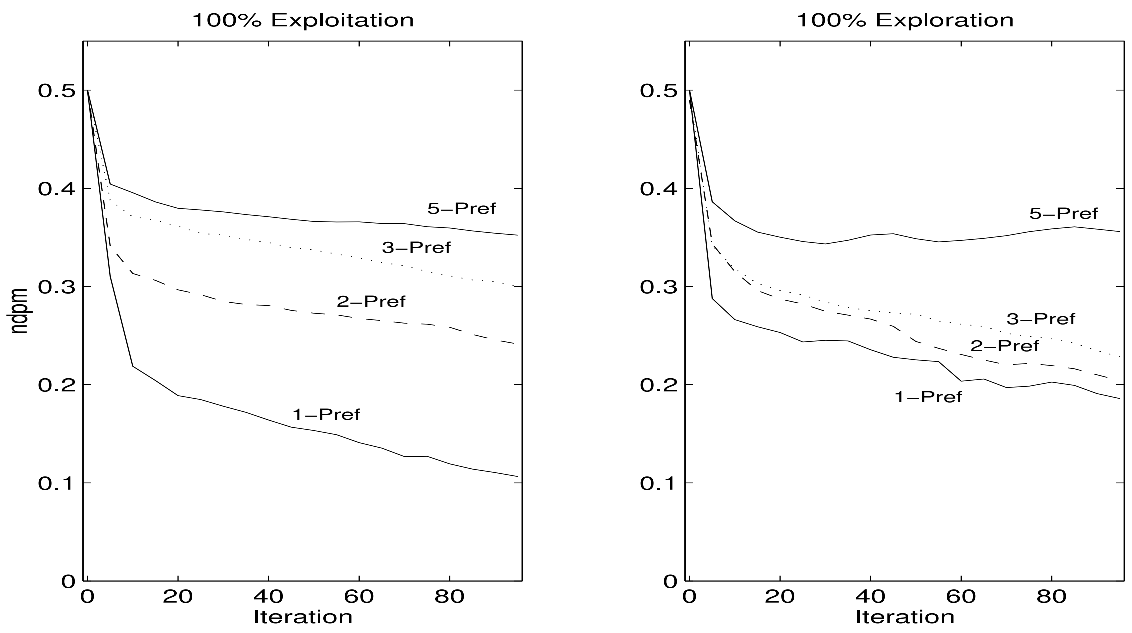

Figure 6.6: Learning curves for users with increasingly complex preferences.

purely exploratory strategy is less dominated by documents from the most preferred topic. This provides an excellent illustration of the exploration/exploitation tradeoff: faster learning versus more documents delivered from preferred topics. It can be seen that a strategy of mixing 50% exploitative choices and 50% exploratory choices leads to an intermediary learning rate and an intermediary document composition. Further intermediate mixes that were tried fell between the results shown as would be expected—for clarity they have not been shown in Figures 6.4 or 6.5. These results are encouraging—they suggest that varying the composition of recommendation sets by changing the mix of exploratory and exploitative choices provides the user with a predictable and well-behaved control.

In general as the number of preferences is increased, learning becomes more difficult, and the simple linear techniques employed become less viable. Figure 6.6 shows how learning is less successful with more complex preference structures.

Changing user profiles

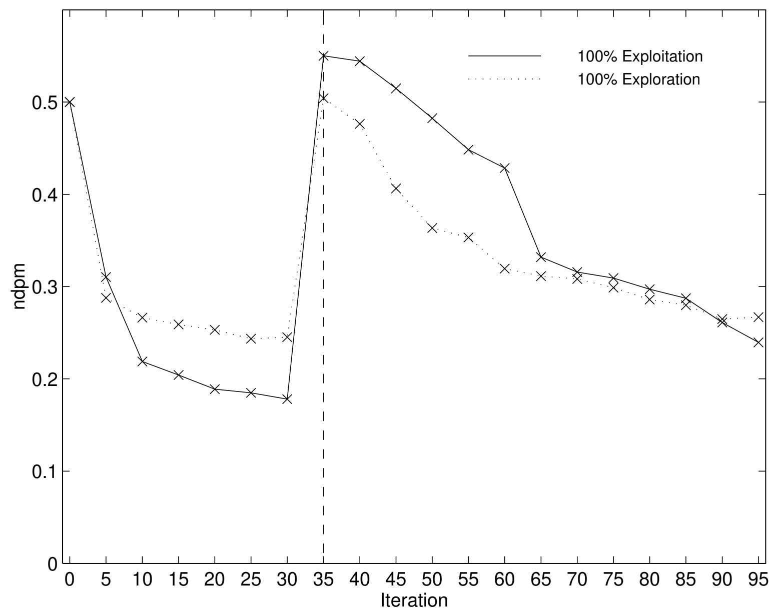

Figure 6.7: Exploitation vs. exploration document selection strategies for 1-pref users where preferences change before iteration 35. Graph shows *ndpm* values averaged over all 10 possible users.

So far, only static user preferences have been considered. However, in a realistic setting for a recommender system, users' interests will be in constant flux. Many different simulations of user preference changes can be imagined, even within the limitations of the n-pref user preference structures defined earlier. In the following experiment, a very simple, drastic change is modeled, which provides a significant challenge to the learning system. A topic from the least preferred equivalence class is promoted to be the most preferred topic. Thus a topic the user previously found among the least interesting becomes the most interesting. An example of such a change in the preference structure for a 1-pref user is:

$$T_1 \succ \{T_2, \ldots, T_{m-1}, T_m\} \longrightarrow T_m \succ \{T_1, T_2, \ldots, T_{m-1}\} \tag{6.2}$$

Since there is no interaction among users in this simulation, all of their preferences are changed synchronously, so that the effects will be maximally observable. Figure 6.7 compares learning curves for the pure exploitation and pure exploration strategies, for 1-pref users (compare to Figure 6.2, which is the same experiment but without the preference changes). It can be seen that the "lag" is much greater for the exploitative strategy, since it is unlikely to deliver documents from the newly preferred topic.

Normally, user interest changes are handled by information filtering systems with some kind of function lessening the influence of older preference judgments, e.g., [Allan, 1996], or the decay function described in section 5.2.1. In other words, there would usually be a change to the learning scheme (the rule used to update a user profile). In contrast, we show here that a change to the document selection strategy can also increase responsiveness to user changes, without necessitating assumptions about the longevity of user interests. It may be more efficient still to alter the document selection strategy when there is evidence that user interest changes are occurring, for instance if there is an increase in the disparity between predicted and actual rankings. Alternatively, deploying an exploration/exploitation control (as described earlier) allows users to manually broaden the selection of documents when their interests change. Thus, another interpretation of the experiment above is that exposing such a control to the user is an appropriate measure to improve responsiveness to interest changes.

6.1.4 Related work

At the intersection of machine learning and IR lies the related problem of text categorization. Lewis *et al.* have applied active learning to this problem, and in particular have shown how actively selecting training data for classifiers can improve performance considerably [Lewis and Gale, 1994; Lewis and Catlett, 1994]. Their technique of *uncertainty sampling* resembles our exploration strategy, and is related to machine learning algorithms such as boosting and query-by-committee (summarized in [Freund, 1994]).

Bookstein [1983] has presented decision theoretic models that explicitly consider the utility of sets of documents retrieved by an IR system, rather than evaluating each document separately.

Collaborative techniques, as introduced in Chapter 4, can be thought of as utilizing the variability of the user population in place of explicit exploratory strategies. Some systems have gone further by explicitly factoring in overall popularity of news articles as well as personal relevance. For instance, the "Krakatoa Chronicle" [Kamba *et al.*, 1996] allows users to vary the proportion of articles matching a user profile to those with popular appeal. In a similar vein, some commercial personalized news services supplant automatic recommendations with occasional overrides, where human editors manually distribute articles of particular importance [Harper, 1997].

Maximal marginal relevance

A new measure that attempts to combine the notions of topical relevance and novelty with respect to the set of recommended documents has been proposed [Tiamiyu and Ajiferuke, 1988; Frederking *et al.*, 1997]. The *marginal relevance* of a document $\mathbf{d}$ with respect to a user profile $\mathbf{m}$ is defined as:

$$\textsc{mr}(\mathbf{m}, \mathbf{d}) = \lambda.\textsc{sim}(\mathbf{m}, \mathbf{d}) \; - \; (1 - \lambda). \max_{\mathbf{d}_i \in R}(\textsc{sim}(\mathbf{d}, \mathbf{d}_i)) \tag{6.3}$$

where R is the set of documents already chosen to be recommended (so initially $R = \emptyset$), and λ is a tunable parameter (this is the definition from [Frederking *et al.*, 1997]).

Note that the setting of the λ parameter does not equate to the proportions of pure exploitation and pure exploration documents as we have defined them. The MR measure takes no account of documents seen *before* the current set of recommendations, whereas the pure exploration strategy takes no account of other documents *within* the current set of recommendations. So rather than delivering a series of documents believed to be new to the user (or at least different from those presented previously), a set is delivered that represents a mix of different topics.

It is unclear which of these strategies a user would prefer, and indeed the motivation for the MR measure was as a way to organize large search results from retrieval operations, rather than as a document selection strategy for a recommendation system.

6.2 The "Slider" interface

The research described in this section is, on the one hand, a new user interface design which explicitly addresses the tradeoff among multiple topics of interest and at the same time gathers implicit feedback about recommended documents. On the other hand, as the latest work chronologically, this is a complete system, which builds on all of the experience with previously described implementations, representing a point in the design space that best exemplifies the goals of this thesis.

Rather than recommending discovered Web pages of all different kinds, as the Fab system does, Slider recommends only news articles. There are two reasons for this specialization:

- The style of the articles and even some of the content is familiar to the users, and they already have a good conceptual model for the domain. Thus deciding on the interestingness of an article is a fairly rapid process, compared to looking over a Web page, which may be part of a larger site and even include interactive features. This enables the kind of short usability tests (where users are observed struggling with the interface) which were desired for testing aspects of the interaction design.

- The exponential growth of the Web during the period between the first experiments described in this thesis (tests of LIRA in 1994) and tests of Slider (early 1998) has necessitated narrowing the scope of the collection phase. It is no longer meaningful to be informed about "what is new on the Web," and the task of following threads of interest can often be better accomplished by restricting the pages considered to those from edited collections. Of course this is dependent on the user's particular interests, and the nature of the relevant Web content. Playing it safe by considering only news articles from well-known sources makes it easier to test the interaction design of Slider by eliminating a source of variation.

6.2.1 Design

The basis of the "Slider" user interface is the extension of user profiles to the multiple-vector case. Whereas previously each user has been represented by a single vector $\mathbf{m}$, in this section each user is represented by a set of such vectors $U = \{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$, which will sometimes referred to as *topic profiles.* Correspondingly, instead of a single recommendation set $R_{\mathbf{m}}$, the system's output is a set $R_U = \{R_{\mathbf{m}_1}, R_{\mathbf{m}_2}, \ldots\}$. The following observations result from this extension:

- Since separate topics can now be explicitly represented in the interface, it is feasible to allow the user to directly control the proportions between them. This is a finer-grained control over the composition of the overall recommendation set than was previously possible.

- If the interface allows users to group recommended items into topics of their own choosing, then this grouping can be used as implicit feedback (Table 2.2), requiring less effort on behalf of the user.

- If the interface allows the user to easily create or delete topics, then transient, short-term information needs can be served in parallel with ongoing, long-term information needs.

The main requirements driving the design of Slider were therefore:

- Affordances to allow users to easily create and discard topics;

- Affordances to allow users to group recommended articles into topics;

- Maintenance and tracking of user profile vectors, one for each user-defined topic.

In addition, an effort was made to create a "lightweight" interface, where screen decoration is spartan and the number of mouse clicks per operation is minimized. The aim is to encourage long-term use and prevent the interface from distracting from the main content (the recommendations themselves), thus minimizing what Cooper would call the "tax" on users [1995].

The interface, based on the principles of direct manipulation [Hutchins *et al.*, 1986], consists of a number of overlapped sliding panels (hence "Slider"), which the user can drag up or down (revealing the panel below). Each panel represents a topic; users define topics by grouping articles together. Each topic receives recommendations based on topical relevance (relative to its topic profile, which is built up from the example articles provided by the user's grouping). Previously recommended documents are excluded: a minimal application of novelty over topical relevance. In addition, documents chosen with an exploratory strategy are recommended in the context of a special "Other News" topic. Positioning, optional naming, creation and deletion of topics are all under user control—users can select their own level of "formality" [Shipman and Marshall, 1994].

Studies of newspaper readers show that very few articles are read in their entirety. Reading just the beginning or scanning through quickly are common behaviors [Graber, 1988]. Therefore summaries are provided in the interface, to allow users to anticipate the content of an article without having to view the full text.

Rather than describing the resulting design textually, it is presented by a series of annotated screen shots (Figures 6.8–6.13). These represent the first three iterations of using the system for a new user; the corpus of documents is a selection of news articles over the period 2–16 January 1998 (to be described more fully in section 6.2.3).

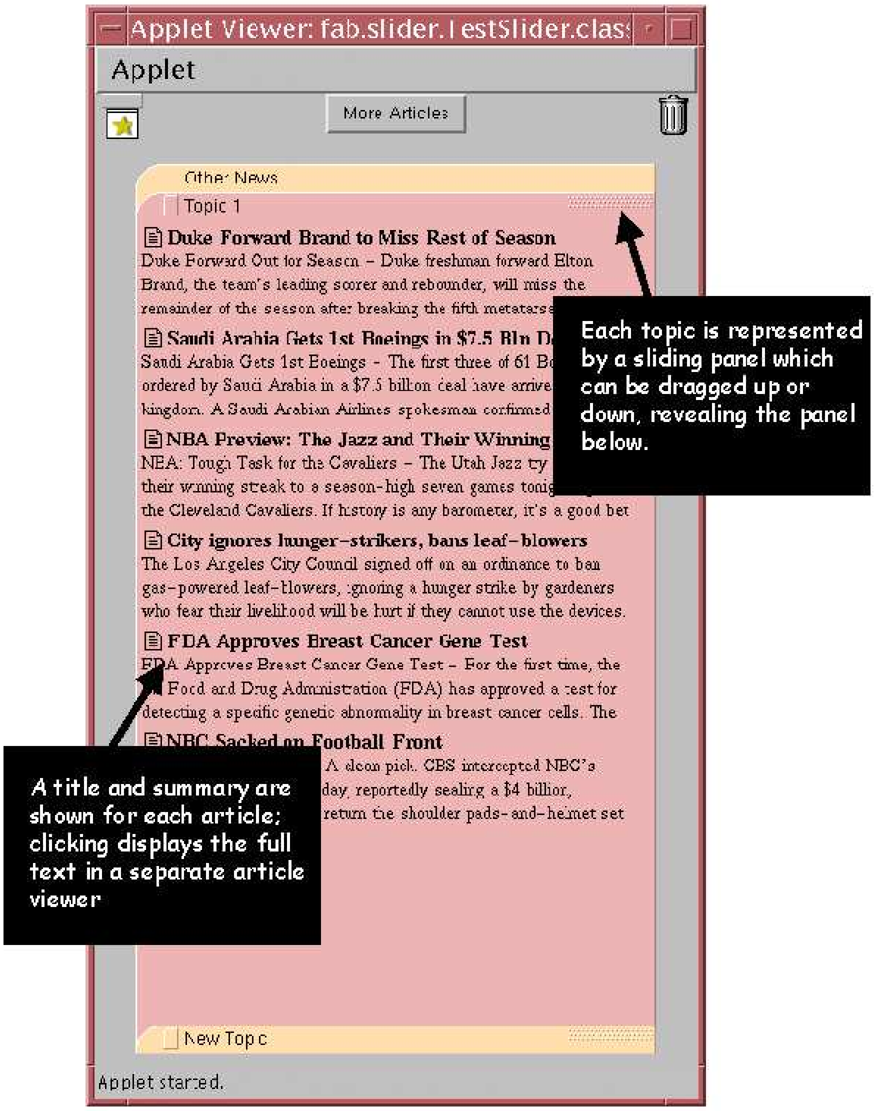

Figure 6.8: Slider interface, 1 of 6

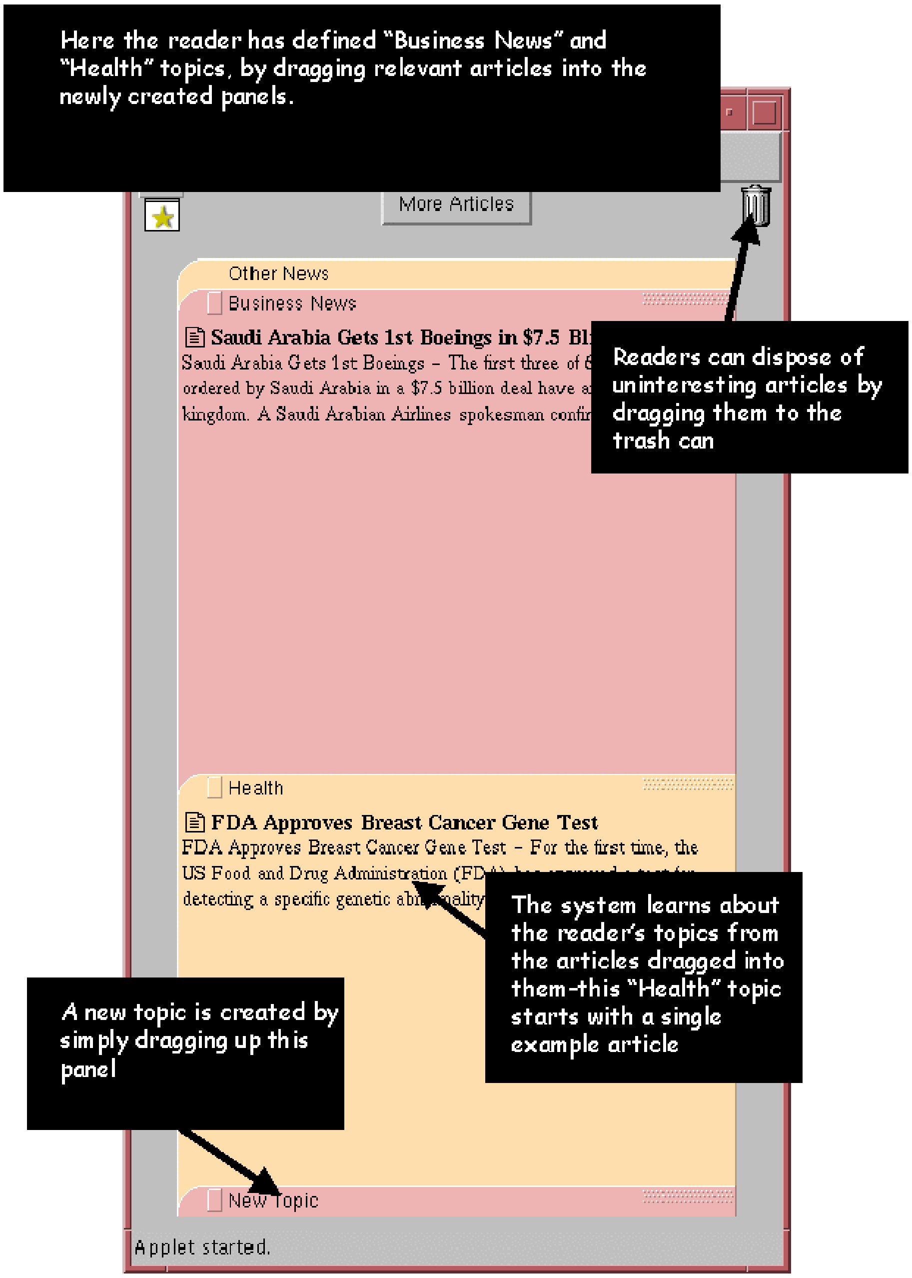

Figure 6.9: Slider interface, 2 of 6

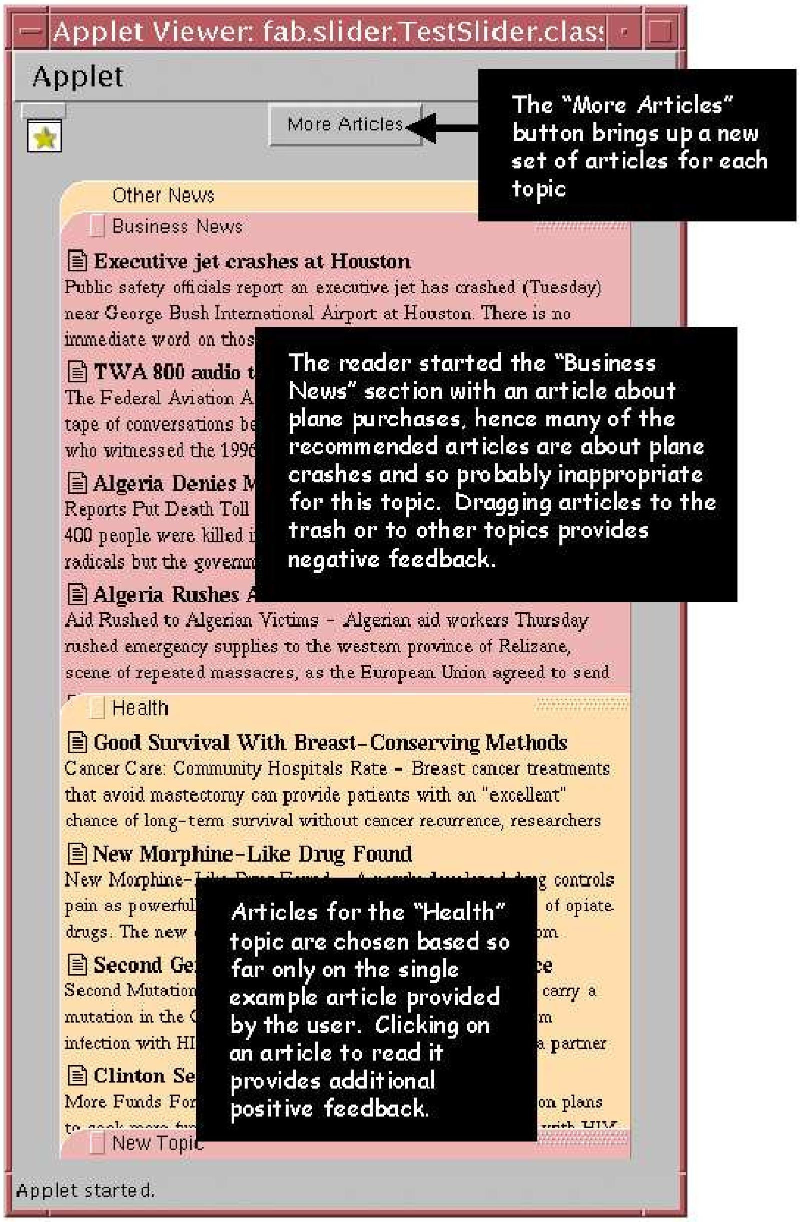

Figure 6.10: Slider interface, 3 of 6

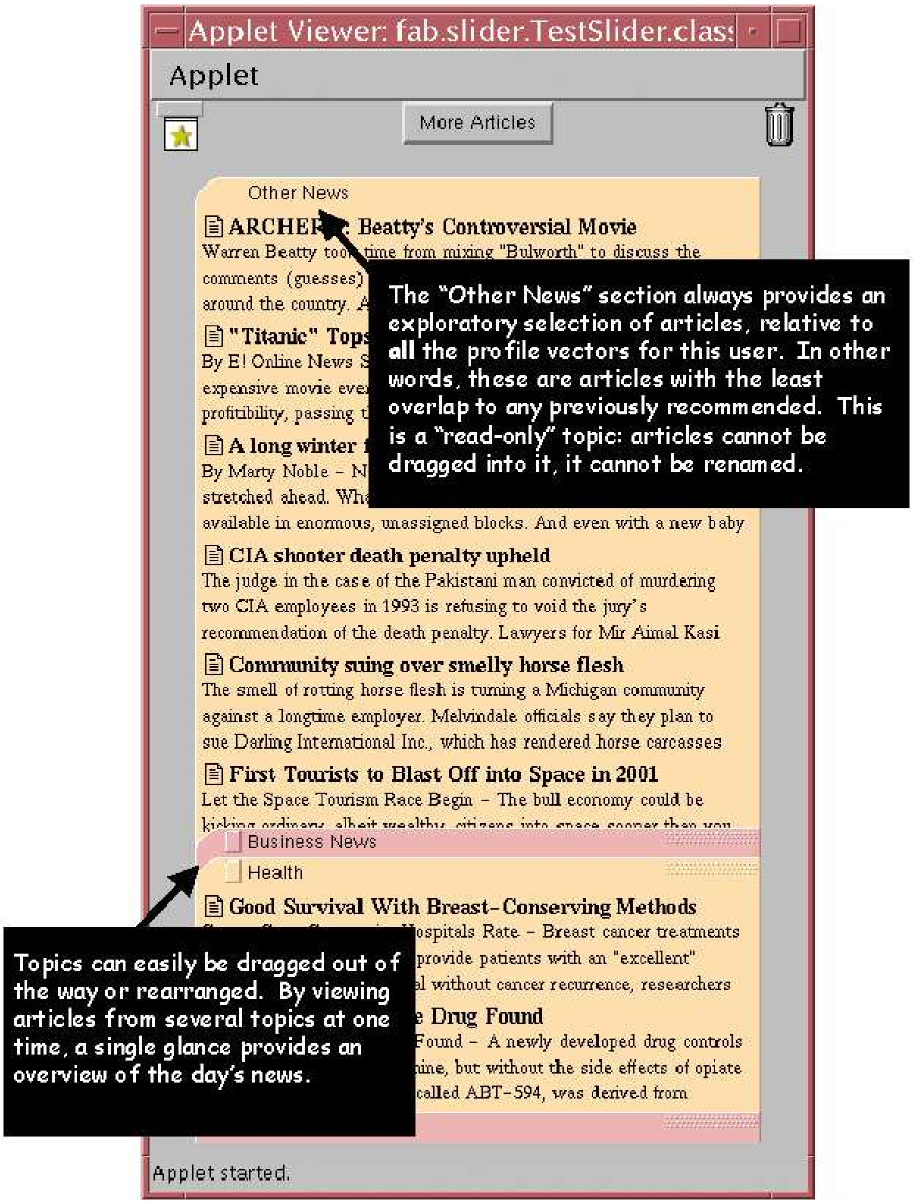

Figure 6.11: Slider interface, 4 of 6

Figure 6.12: Slider interface, 5 of 6

Figure 6.13: Slider interface, 6 of 6

Interpreting input

The particular interface illustrated provides rich opportunities for making sense of the user's actions. In the following description of how different interactions are interpreted as relevance feedback, $T_1, T_2 \ldots$ denote topics as defined by the user in the interface; the system correspondingly maintains a multiple-vector user profile $U = \{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$. Numeric parameters are denoted by λ_1–λ_5.

Create new topic T_1: No particular feedback is implied (until the user moves items into this new topic).

Move item d from topic T_1 to topic T_2: Subtract $\lambda_1\mathbf{d}$ from profile $\mathbf{m}_1$ (it was erroneously recommended for topic T_1), and add $\lambda_1\mathbf{d}$ to profile $\mathbf{m}_2$ (such items should be recommended in the context of topic T_2).

Delete item d: Subtract $\lambda_2\mathbf{d}$ from every profile; items like $\mathbf{d}$ should not be recommended at all.

Click on item d in topic T_1 to see full text: Add $\lambda_3\mathbf{d}$ to the profile $\mathbf{m}_1$; it is a good example of an appropriate item for topic T_1.

Drag a gold star to item d in topic T_1: Add $\lambda_4\mathbf{d}$ to profile $\mathbf{m}_1$.

Delete entire topic T_1: Profile $\mathbf{m}_1$ is no longer of interest; remove $\mathbf{m}_1$ from set U.

Take no action for item d recommended for topic T_1: Add $\lambda_5\mathbf{d}$ to the profile $\mathbf{m}_1$; if an item was not moved or deleted mildly positive feedback is inferred.

For the experiments to be described, the parameters were set as in Table 6.1. Note that multiple profile additions or subtractions can result from the actions above: if a user both read an article and dragged a gold star to it, the total feedback would be $0.25 + 0.5 + 3 = +3.75$.

Parameter	Description	Value
λ_1	Weight for moved article	3
λ_2	Weight for deleted article	3
λ_3	Weight for read article	0.5
λ_4	Weight for article annotated with gold star	3
λ_5	Weight for article where no action taken	0.25

Table 6.1: Parameters used to adjust profiles given user actions.

The implicit feedback scheme above is consistent with the findings of Sakagami and Kamba [1997] (although there feedback was on a per-article basis, so the grouping behaviors were not supported). Their "enlarge window" action, similar to our "clicking" behavior, was found to correlate well with users' subsequent relevancy ratings of articles. In addition, they found that users preferred to have the option of explicit feedback for indicating particularly strong interests—this option is provided by the gold star in the scheme above.

Document selection

For each user-defined topic T_i, the standard exploitative, content-based strategy is followed. Six unseen documents are picked, which have the highest similarity to the topic profile $\mathbf{m}_i$ (using the regular SIM measure).

One of the interesting features of this interface is the approach to the exploration/exploitation tradeoff. The same mechanisms as described in section 6.1 are applied, but exposed to the user in an entirely different way. Exploratory articles are presented in the context of a separate topic called "Other News." More precisely, documents $\mathbf{d}$ are picked for "Other News" such that $\textsc{overlap}((\sum_{\mathbf{m} \in U} \mathbf{m}), \mathbf{d})$ is minimized (excepting again all previously recommended documents). "Other News" is a topic into which articles cannot be moved, and which can be neither renamed nor moved (it is a "read-only" topic). Thus "Other News" represents articles dissimilar to any recommended in the past, for any of the user-defined topics.

The problem of overspecialization still can and does occur in this setting. For instance, the user creates a "Technology News" topic and drags into it some example technology articles, including one mentioning Microsoft Corp. Over time it can easily happen that the topic becomes overwhelmed with articles about Microsoft, an overly narrow selection. In a sense the overspecialization problem has been moved one level down—now it occurs per topic instead of for the whole recommendation set. This is preferable from the user's perspective, since it is still possible to maintain disparate topics of interest, or to create new ones. Furthermore, additional articles from the "Other News" topic can be dragged into a user-defined topic like "Technology News," thus broadening it by introducing feedback over new subject matter.

6.2.2 Implementation

The interface illustrated in Figures 6.8–6.13 is implemented as a Java applet. Extensive use is made of the SubArctic user interface toolkit [Hudson and Smith, 1996], which provides a set of components easily customizable through subclassing.

The applet communicates to a Fab selection agent through an HTTP interface, similar to those described in Chapter 5. Since Fab agents are implemented in Python, some translation work is required when passing name/value pairs over the HTTP connection. Luckily the existing CGI protocol is easy to use when making a request from a server. In the reverse direction, plain text formatted according to the Java Properties class (`java.util.Properties`) is used to encode name/value pairs in an easily parseable manner.

The layout shown is viable in a wide range of sizes, and special care is taken to ensure that, where possible, proportions between sizes of the different topic panels are maintained as the applet window is resized.

As alluded to in Figure 6.8, a separate article viewer was implemented—a simple scrollable text window. The full text of each article is retrieved via a cache on a local Web server—consistent response time is assured by not requiring the Yahoo! Web server to be accessed in real time. Furthermore, the simple viewing window prevents any distractions which may potentially result from using a full-fledged Web browser.

6.2.3 Experiments

This section describes usability tests, resembling the informal, qualitative type of testing espoused in [Gomoll, 1990; Nielsen, 1994]. The aim was to observe usage of the interface, and determine:

- Whether the behaviors of creating new topics and grouping articles accordingly would be adopted and lead to results users consider interesting or useful;
- How the resulting topics compare to the category labels assigned to articles by the originating news services;
- How users would perceive or exploit the "Other News" topic.

Corpus

The corpus consisted of news articles covering the period 2–16 January 1998, as shown in Table 6.2. The usability tests were conducted starting 16 January 1998—it was felt that a recent collection of general news would hold users' interest better than the commonly used Reuters-22173 or Reuters-21578 collections, which contain Reuters financial stories from 1987, or the TREC-AP collection, which contains Associated Press stories from 1988–90[1].

[1]Information on these collections is available from David D. Lewis' professional home page, currently at http://www.research.att.com/~lewis.

Name of topic	URL (http://www.yahoo.com/headlines prefix omitted)	Number of articles
Business	/business/summary.html	165
Technology	/tech/summary.html	115
International	/international/summary.html	129
Sports	/sports/summary.html	190
Entertainment	/entertainment/	141
Politics	/politics/summary.html	151
Health	/health/summary.html	98
Odd	/odd/	141
Crime	/crime/	210
E! Online	/eonline/	58
The Sporting News	/sportingnews/	59
Wired News	/wired/	88
ZDNet News	/zdnews/	79
	Total	1624

Table 6.2: Topics used for news article collection.

Articles were collected from the publicly accessible Yahoo! Headlines Web site[2]. The majority were newswire articles provided by Reuters, although a variety of other sources were represented including the United Press International (UPI) newswire, entertainment magazines *Variety* and *E! Online*, sports magazine *The Sporting Life* and technology news services *Wired News* and *ZDNet News*. No local news was included, only national (USA) or international.

Summaries for most articles were fetched from the Yahoo! Headlines site. Where no summary was available, a simple one was generated by extracting the first few sentences of the article (using a number of site-specific heuristics to skip past the various headers and navigation buttons). This is effective since news articles are written in an "inverted pyramid" style, in which the first paragraph essentially summarizes

[2]Currently at http://www.yahoo.com/news.

the remainder of the article (reading further provides more detailed information).

Users

There were 9 users (4 women and 5 men), recruited using notices sent to a number of electronic mailing lists. They included 4 students, 2 university researchers, 1 librarian, 1 software engineer and 1 market researcher. 5 of them rated themselves as regular computer users, and 4 as competent computer programmers.

Task description

The users were told that they would be using a personalized newspaper, which would attempt to learn about them by monitoring their actions. The task was described simply as "using the system to see articles that match your topics of interest." They were informed of the range of dates covered by the corpus, and that the articles were the sorts of stories which might appear in a national newspaper (i.e., more specialized areas of interest would not be represented). The users were asked to think aloud during the resulting interactions, which lasted approximately 45 minutes and were videotaped. No assistance was provided unless the user had trouble with some aspect of the interface for at least several minutes.

6.2.4 Observations

There are three classes of observations resulting from the usability studies: successes and problems with detailed design of the user interface components; quality of recommendations given users' intentions and desires for particular topics; and the nature of users' overall expectations and activities as they experienced the interface. The former class of observations will not be discussed here: as expected, a number of improvements were suggested to both the visual appearance and details of the interactivity. The latter two classes, however, will be described here. Given the informal nature of the experiments, results are of an anecdotal rather than statistical nature.

Topics

All users categorized articles into topics they created (between 3–7 topics each, a total of 35 altogether). The majority (30) of these topics turned out to be subsets of the categories as listed in Table 6.2. For instance, one user defined a topic named "Technology," where the desired content was described as "breakthroughs, people, new gadgets, not stuff about the marketplace" (whereas the corpus "Technology" topic contains a lot of financial news). Another example was a topic named "Films," which is not only a subset of the corpus "Entertainment" topic, but was narrowed further still to exclude film-related financial news (e.g., box office takings) and films or actors not of particular interest.

The remaining 5 topics appeared to be outside of the corpus categories, and indeed 4 of these did not lead to successful recommendations due to the paucity of relevant articles in the corpus: "Mountaineering," "Art," "Soviet Central Asia" and "Journalism." The 5^{th} topic, "Social Issues," did lead to recommendations with which the user was satisfied (a mix of "Health News," "Politics," and "World News" articles).

The topic profiles were learned successfully: altogether 28 of the 35 topics, by the final iteration, had returned a set of recommendations that the user felt were mostly or wholly relevant for the topic. The number of iterations per user varied from 4 to 23. Table 6.3 shows a user's recommendations at the 4^{th} iteration.

Most users ended up with a similar understanding of "Other News:" "brain candy," "things that I might find interesting and kind of fun to read," "a way for me to pick extra topics," "the 'Living' section," "everything I haven't categorized yet," "random selection of other news outside of my areas of interest."

Topic name	Article title
Other News	Hacking the Human Operating System, the Brain
	REVIEW/TELEVISION: 'Nightmare' A Refreshing Sci-Fi Film
	Packers/Niners in NFC Title Game Sunday
	Cricket: Kallis Century Saves South Africa
	NFL Playoffs Begin Saturday
	Godfather of Blues Junior Wells Dies
Domestic law enforcement issues	Sniper kills two in shootings
	Judge upholds state smoking law
	Mobster says FBI agent got loan
	School shooting suspect in court
	Tobacco settlement likely on Thursday
	McVeigh atty wants to lengthen appeal
Foreign incidents	Report Says Hundreds More Killed in Algeria
	Algeria Rushes Aid to Massacre Survivors
	U.S. Eyes Boosting Cultural Ties with Iran
	Attackers Slaughter 103 Near Algiers
	Muddled Message from Iran on U.S. Relations
	U.S. Reviewing its Sanctions Policy
Domestic technology	A hard hearing for Microsoft, DOJ
	Microsoft Stock Sinks As Battle Expands
	Siemens Confirms Plans For Chip Venture
	Intel Optimistic On Digital Pact
	FTC Allows Intel Graphic Chip Acquisition
	Microsoft accuses government of contradicting self

Table 6.3: Example set of recommendations.

Overall experiences

A trait of all of the users tested, and indeed users of the earlier LIRA and Fab systems, is that the interestingness or content of an article is a secondary reason for their actions: primarily, they are consciously attempting to control the learning system. For instance, on explaining the decision not to delete an uninteresting article: "I don't want to give it [the computer] the wrong impression.". When asked why an article was dragged to the trash or annotated with a gold star, the answer would most frequently be phrased in terms of controlling the software: "I want it to..." In this case, people learn to control the adaptive software quicker than it can learn to adapt to their needs! Incidentally, this is a reason not to use strictly objective relevance assessments (such as those gathered for TREC corpora) when simulating performance of recommender systems.

A duality was observed in users' categorization behavior. On the one hand, a desired topic was known ahead of time, on the other, topics arose from attempts to categorize the incoming news (initially a randomly picked set of articles). Most users engaged in a mix of these behaviors, possibly due to the mixed message of the phrase "personalized newspaper"—you can choose what you are interested in, but you cannot choose what is "news". Hence there would be a mixture of annoyance ("This was supposed to be finance, and it turned into much more general news... that's terrible!") and fatalism ("Iran and Algeria are dominating the foreign one," referring to a "Foreign Incidents" topic) when faced with seemingly inappropriate or undesired recommendations. It was interesting to note that all of the users engaged in categorization based on subject matter. Sometimes this seemed a natural orderliness, and at other times it was resented as a necessary step towards training the system ("I have to go in and clean up that topic, otherwise it will start to get polluted [with irrelevant articles]"). Integration with longer-term storage and organization mechanisms (e.g., a file system) would lessen the "explicit feedback" flavor of the grouping activity engendered by this experimental setting.

The most frequent problem users had with their recommendations was the increasing narrowness of each topic. Although "Other News" was usually recognized as a broader source of articles, there were many requests for controls to broaden a particular topic, or have a view of the space of articles ("I don't know what I'm missing"). Some of these complaints are really an artifact of the experimental setting, since a live deployment of this system would recommend only very current news articles—the narrowing down is only possible with a larger database of articles which contained, for instance, two weeks' worth of regular stories about ongoing Microsoft Corp./US Department of Justice lawsuits.

6.2.5 Related work

Although document and user representations created for the Fab system have been used for the implementation described, other commercial or academic IR engines providing similar capabilities could be substituted. The intention behind the Slider prototype was more to investigate interactions with a recommender system based on users' grouping behavior. Several information filtering systems are of particular relevance. INFOSCOPE [Fischer and Stevens, 1991] users create "virtual newsgroups" by defining filtering rules to select articles from specified parent Usenet newsgroups. In addition, agents automatically suggest new rules (based on article header information) given users' reading habits. In a similar vein, the "pile" metaphor [Rose *et al.*, 1993] was developed to allow automatic or manual informal groupings of documents, as a possible extension to file system interfaces. An advisory system suggests appropriate piles for incoming documents based on content vectors. The "Fishwrap" personalized newspaper [Chesnais *et al.*, 1995] presents both a customizable selection of topics and "Page One"—articles representing the readership's breadth of interests, somewhat similar in spirit to our "Other News."

Other researchers have investigated the use of implicit feedback. Morita and Shinoda [1996] demonstrate a system that successfully learns user profiles by measuring reading time, and similar results are reported in [Miller *et al.*, 1997]. Kamba *et al.*, in the "Krakatoa Chronicle" mentioned earlier [Kamba *et al.*, 1996], as well as the subsequent Japanese language ANATAGONOMY system [Sakagami and Kamba, 1997], monitor users' window control actions such as resizing, scrolling and maximizing, all of which are taken to indicate interest in an article. Other observable activities which have been used for feedback include bookmarking behavior while using a Web browser [Rucker and Polanco, 1997], postings to Usenet news groups [Terveen *et al.*, 1997], and the action of recommending an item to a colleague [Maltz and Ehrlich, 1995].

6.2.6 Discussion

In an earlier chapter (section 2.2.3) it was pointed out that the goals of a text recommender system are very limited compared to those of a mass-audience newspaper. The user is aided in following particular threads of interest, but there is no promise of completeness of coverage as would be expected from a newspaper. One interpretation for the requests for broadness in the tests of Slider is that this limitation is not justified—that users do indeed require a similar completeness. The fear of missing something important seems ever-present with a software rather than human editor. Perhaps this issue is accentuated by the fact that the domain is news articles as opposed to arbitrary Web pages.

A true deployment of Slider would see it embedded within or closely integrated with other applications the user would use for browsing, reading and organizing news articles or other documents. In this way the feedback used would truly be implicit. However, the version tested, as shown in Figures 6.8–6.13, is a stand-alone application, and so there is a more explicit nature to the feedback. A more realistic test, beyond the scope of most university projects, would be to integrate Slider fully with existing news Web sites or browsing software, and study its usage over the long term. As well as personalized "Other News," the opportunity would then exist to also present top stories as selected by human editors, and allow users to freely add such stories to their own topics.

6.3 Summary: Deciding on the composition of recommendation sets

The second stated research issue motivating this thesis was how to decide upon the composition of recommendation sets. The exploration/exploitation parameter was implemented to allow users to allow users to select "broad" or "narrow" sets of recommendations. The series of experiments in this chapter confirm our hypotheses about this parameter. For users with simple single-topic preferences, an exploitative strategy has turned out to be best; for more complex users such as those with triple-topic preferences, there is a tradeoff between faster learning of user models and delivery of more preferred documents. In addition, exploratory strategies adapt to user preference changes more rapidly than exploitative strategies.

The Slider interface also increases users' control over their recommendation sets, in this case by explicitly representing several interests. The interface provides a lightweight mechanism for users to define multiple topics of interest and control the proportions between them. Observations from initial usability tests with a collection of news articles show that topic profiles are successfully learned using only the implicit feedback of users' clicking and drag-and-drop actions. The exploration/exploitation parameter is employed in a rather different fashion, to produce the "Other News" topic.

Users' comments suggest that a per-topic broad/narrow control would also be useful—a promising direction for future research is therefore to combine the two user interface approaches taken above.

Chapter 7

Conclusions

The work described has spanned a number of academic disciplines (including information retrieval, machine learning, human computer interaction and user modeling) in order to create better text recommender systems. In coming years it is possible that "recommender systems" will deserve an established academic discipline of their own. A 1996 CSCW workshop [Konstan and Bharat, 1996] led to the creation of an on-line community of researchers linked by a mailing list and Web page[1]; a recent special issue of the *Communications of the ACM* journal (introduced with [Resnick and Varian, 1997]) is being followed up by a workshop scheduled for the 1998 AAAI conference [Kautz *et al.*, 1998]. This chapter, therefore, will review some of the research issues that remain to be addressed within this field, and point out some promising directions for future work. The thesis is concluded by way of a summary: the main research contributions are revisited.

[1] Currently at http://www.sims.berkeley.edu/resources/collab/

7.1 Future directions for recommender systems

In this section a few research issues are highlighted which, given all of the experience with users of LIRA, Fab and Slider, are perceived to be key stumbling blocks on the way to wider usage of personalized recommender systems.

- For recommender systems to become prevalent, they must make use of implicit feedback. The Slider system demonstrates that having access to users' grouping, reading and browsing activities provides a wealth of examples from which to learn. Many computer users today keep multiple, separate hierarchies of content, for instance for email, Web pages and personal files and documents (not counting various hierarchies which may not be stored on a computer, such as papers in filing cabinets). An interesting opportunity exists to create interfaces that allow general retrieval and organizational activity for all of these kinds of hierarchies, integrated with a recommender system.

- An ongoing endeavor of research into personalized newspapers is how to balance editorial input with personal recommendations—users want both. The exploration/exploitation parameter can be considered an initial attempt to capture the related dimension of "breadth" versus "narrowness." A useful capability, often requested by users, would be to be able to control these factors more precisely, for instance to be able to request more breadth while remaining within the context of a certain topic.

- One of the advantages of the text recommendation task is that it lends itself to producing output for less interactive, more widespread media. For instance, a daily period spent commuting by car could be spent listening to an audio version of the day's recommended news articles. Alternatively, an "anytime" recommender system could continually be attempting to optimally fill be remaining time between the present location and the destination (estimatable from GPS location information). In situations such as these, knowing desired proportions between a user's topics of interest is of even greater value than in a highly interactive interface such as Slider.

- Two extremes were discussed at the beginning of this thesis. Pure content-based recommendation uses low-level features of a document: its words. Pure collaborative recommendation uses very high-level features: other users' ratings of the document as a whole. A final research issue to ponder is the extraction or inference of mid-level features. For instance, users' categorizations give us not only an implied rating for a document, but also an appropriate label and a number of other documents that are similar from that user's perspective. Tests of the Tapestry system [Goldberg *et al.*, 1992] revealed both "eager" readers, who were the first to read items and make annotations, and "casual" readers who were happy to wait and reap the benefits of other users' experience. The PHOAKS system [Terveen *et al.*, 1997] relies on a similar role decomposition. Mid-level features, then, such as quality, specificity, style of writing or political leaning could all potentially be inferred from the actions of specialized eager readers, if they are provided with suitable interfaces.

7.2 Contributions revisited

A number of research contributions were listed in section 1.3—here a brief summary of each one serves to conclude this thesis.

An architecture linking populations of adaptive software agents, which takes advantage of the overlaps of interests among users of a recommender system to increase efficiency and scalability

The Fab architecture, described in Chapter 5, allows a variable number of users, whose interests may be constantly changing, to be provided Web page recommendations by a variable number of agents, depending on resources available. As user tests show, the agents converge, over time, to areas of interest to multiple users, where positive feedback is most plentiful. The extent to which the population of users has interests in common determines the efficacy of this scaling-up method.

A novel recommendation mechanism combining a content-based and a collaborative method

In addition to the features described above, the Fab architecture provides a way to combine the advantages of two common recommendation schemes: content-based and collaborative. In doing so, it not only eliminates many of the disadvantages of using either approach alone, but it also extends the reach of collaborative filtering systems to dynamic text-based domains.

A new affordance allowing users to control the breadth or narrowness of their set of recommendations

The exploration versus exploitation control, described in detail in section 6.1, is designed to address the problem of deciding on the composition of sets of recommendations, a major research theme of this thesis. It allows users to view documents chosen from either an exploitative strategy, as in a regular information filtering system, or from an exploratory strategy, using a novel implementation. In a simulated setting, experiments show that the exploratory strategy increases the rate of learning of user profiles, and the rate of adaptation to changing user interests, at the expense of showing users documents that are less often to do with their preferred topics.

A new interaction design allowing users to vary the proportions between topics of interest, where only implicit feedback is required

The Slider user interface, described in section 6.2, introduces a lightweight interface mechanism allowing the user to specify multiple topics of interest, and to view articles from several topics, in varying proportions, at a single glance. In addition, the interface does not require explicit feedback from the user, but, as shown in informal usability tests, successfully learns topic profiles given the user's drag-and-drop and clicking actions.

Bibliography

[Ackley and Littman, 1992] D. H. Ackley and M. L. Littman. A case for Lamarckian evolution. In Christopher G. Langton, editor, *Artificial Life III*. Addison Wesley, 1992.

[Adams and Lloyd, 1984] Douglas Adams and John Lloyd. *The Meaning of Liff*. Crown Pub., 1984.

[Allan, 1995] James Allan. Relevance feedback with too much data. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 1995.

[Allan, 1996] James Allan. Incremental relevance feedback for information filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 270–278, August 1996.

[Allen, 1990] Robert B. Allen. User models: theory, method and practice. *International Journal of Man-Machine Studies*, 32:511–543, 1990.

[Armstrong *et al.*, 1995] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. WebWatcher: A learning apprentice for the World-Wide Web. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Resources*, March 1995.

[Asnicar and Tasso, 1997] Fabio A. Asnicar and Carlo Tasso. ifWeb: a prototype of a user model-based intelligent agent for filtering and navigation in the world-wide web. In *UM97 Workshop on Adaptive Systems and User Modeling on the World Wide Web, 6th International Conference on User Modeling*, June 1997.

[Balabanović and Shoham, 1995] Marko Balabanović and Yoav Shoham. Learning information retrieval agents: Experiments with automated web browsing. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Resources*, March 1995.

[Balabanović and Shoham, 1997] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.

[Balabanović *et al.*, 1997] Marko Balabanović, Yoav Shoham, and Yeogirl Yun. An adaptive agent for automated web browsing. Technical Report CS-TN-97-52, Stanford University Department of Computer Science, February 1997.

[Balabanović, 1997] Marko Balabanović. An adaptive web page recommendation service. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 378–385, February 1997.

[Balabanović, 1998a] Marko Balabanović. Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction (to appear)*, 8(1), 1998.

[Balabanović, 1998b] Marko Balabanović. An interface for learning multi-topic user profiles from implicit feedback. Working Paper SIDL-WP-1998-0089, Stanford University Digital Libraries Project, 1998.

[Balabanović, 1998c] Marko Balabanović. The "slider" interface. *IBM interVisions*, 11, February 1998.

[Baldonado *et al.*, 1997] Michelle Baldonado, Chen-Chuan K. Chang, Luis Gravano, and Andreas Paepcke. Metadata for digital libraries: Architecture and design rationale. In *Proceedings of the 2nd ACM International Conference on Digital Libraries*, July 1997.

[Belkin and Croft, 1987] Nicholas J. Belkin and W. Bruce Croft. Retrieval techniques. In Martha Williams, editor, *Annual Review of Information Science and Technology (ARIST)*, volume 22, pages 109–145. Elsevier Science, 1987.

[Belkin and Croft, 1992] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, December 1992.

[Bender *et al.*, 1991] Walter Bender, H. Lie, Jon Orwant, Laura Teodosio, and N. Abramson. Newspace: Mass media and personal computing. In *Proceedings of the Summer 1991 USENIX Conference*, pages 329–349, 1991.

[Berners-Lee *et al.*, 1992] T. Berners-Lee, R. Cailliau, J-F. Groff, and B. Pollermann. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 1(2):52–58, Spring 1992.

[Berry and Fristedt, 1985] D. A. Berry and B. Fristedt. *Bandit problems: Sequential allocation of experiments.* Chapman and Hall, London, UK, 1985.

[Bollman and Wong, 1987] P. Bollman and S.K.M. Wong. Adaptive linear information retrieval models. In *Proceedings of the 10th International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 157–163, June 1987.

[Bookstein, 1983] Abraham Bookstein. Information retrieval: A sequential learning process. *Journal of the American Society for Information Science*, 34(5):331–342, 1983.

[Borko, 1962] Harold Borko. The construction of an empirically based mathematically derived classification system. In *AFIPS Proceedings, Spring Joint Computer Conference*, pages 279–289, 1962.

[Brajnik *et al.*, 1987] Giorgio Brajnik, Giovanni Guida, and Carlo Tasso. User modeling in intelligent information retrieval. *Information Processing & Management*, 23(4):305–320, 1987.

[Buckley *et al.*, 1994] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART: TREC-3. In *Proceedings of the 3^{rd} Text REtrieval Conference*, November 1994.

[Carbonell, 1983] Jaime Carbonell. The role of user modeling in natural language interface design. Technical Report CMU-CS-83-115, Carnegie-Mellon University, Department of Computer Science, 1983.

[Chesnais *et al.*, 1995] Pascal R. Chesnais, Matthew J. Mucklo, and Jonathan A. Sheena. The Fishwrap personalized news system. In *Proceedings of the 2^{nd} IEEE International Workshop on Community Networking*, pages 275–282, June 1995.

[Cooper, 1995] Alan Cooper. *About Face: The Essentials of User Interface Design.* IDG Books, 1995.

[Cousins *et al.*, 1997] Steve B. Cousins, Andreas Paepcke, Terry Winograd, Eric A. Bier, and Ken Pier. The digital library integrated task environment (DLITE). In *Proceedings of the 2^{nd} ACM International Conference on Digital Libraries*, July 1997.

[Cox, 1980] Eli P. Cox, III. The optimal number of response alternatives for a scale: A review. *Journal of Marketing Research*, 17:407–422, November 1980.

[Davis, 1994] Glenn Davis. Cool site of the day, 1994. `http://www.infi.net/cool.html` *(no longer available)*.

[Deerwester *et al.*, 1990] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[Dozier and Rice, 1984] D. Dozier and R. Rice. Rival theories of electronic newsreading. In R. Rice, editor, *The New Media*, pages 103–128. Sage Publications, 1984.

[Fiduccia and Mattheyses, 1982] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *Proceedings of the* 19^{th} *IEEE Design Automation Conference*, pages 175–181, 1982.

[Fischer and Stevens, 1991] Gerhard Fischer and Curt Stevens. Information access in complex, poorly structured information spaces. In *Conference on Human Factors in Computing Systems (CHI '91)*, pages 63–70, 1991.

[Fisk, 1996] Donald Fisk. An application of social filtering to movie recommendation. *BT Technology*, 14(4):124–132, October 1996.

[Foltz and Dumais, 1992] Peter W. Foltz and Susan T. Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, December 1992.

[Frakes, 1992] William B. Frakes. Stemming algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval Data Structures and Algorithms*, pages 131–160. Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.

[Frederking *et al.*, 1997] Robert Frederking, Teruko Mitamura, Eric Nyberg, and Jaime Carbonell. Translingual information access. In *Proceedings of the AAAI Spring Symposium on Cross-Language Text and Speech Retrieval.* AAAI Press, March 1997.

[Frei and Schäuble, 1991] H. P. Frei and P. Schäuble. Determining the effectiveness of retrieval algorithms. *Information Processing & Management*, 27(2/3):153–164, 1991.

[Freund, 1994] Yoav Freund. Sifting informative examples from a random source. In *Proceedings of the AAAI Fall symposium on intelligent relevance.* AAAI Press, 1994. (Available as AAAI Technical Report FS-94-02).

[Fuhr, 1989] Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems*, 7(3):183–204, July 1989.

[Goldberg *et al.*, 1992] David Goldberg, David Nichols, Brain M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[Gomoll, 1990] Kathleen Gomoll. Some techniques for observing users. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 85–90. Addison-Wesley, Reading, MA, 1990.

[Graber, 1988] Doris A. Graber. *Processing the News: How people tame the information tide.* Longman Inc., 1988.

[Gray, 1997] Matthew Gray. Web growth summary, January 1997. `http://www.mit.edu/people/mkgray/ net/web-growth-summary.html`.

[Grefenstette, 1991] John J. Grefenstette. Lamarckian learning in multi-agent environments. In *Proceedings of the 4^{th} International Conference of Genetic Algorithms*, pages 303–310, 1991.

[Grefenstette, 1995] Gregory Grefenstette. Comparing two language identification schemes. In *Proceedings of the 3^{rd} International Conference on Statistical Analysis of Textual Data (JADT 95)*, December 1995.

[Harman, 1994] Donna Harman. Overview of the third Text REtrieval Conference (TREC-3). In *Proceedings of the 3^{rd} Text REtrieval Conference*, November 1994.

[Harman, 1996] Donna Harman. Overview of the fifth Text REtrieval Conference (TREC-5). In *Proceedings of the 5^{th} Text REtrieval Conference*, November 1996.

[Harper, 1997] Christopher Harper. MIT, Mecca and the Daily Me. *American Journalism Review*, March 1997.

[Hendrickson and Leland, 1994] Bruce Hendrickson and Robert Leland. The Chaco user's guide: Version 2.0. Technical Report SAND94–2692, Sandia National Laboratories, 1994.

[Hendrickson and Leland, 1995] Bruce Hendrickson and Robert Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.*, 16(2):452–469, 1995.

[Hey, 1989] John B. Hey. System and method of predicting subjective reactions. Patent 4870579, United States Patent Office, September 1989.

[Hey, 1991] John B. Hey. System and method for recommending items. Patent 4996642, United States Patent Office, February 1991.

[Hill *et al.*, 1995] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Conference on Human Factors in Computing Systems (CHI '95)*, May 1995.

[Höök *et al.*, 1997] Kristina Höök, Åsa Rudström, and Annika Waern. Edited adaptive hypermedia: Combining human and machine intelligence to achieve filtered information. In *Workshop on Flexible Hypertext, Proceedings of the 8th Annual Hypertext Conference*, April 1997.

[Hudson and Smith, 1996] Scott E. Hudson and Ian Smith. SubArctic UI toolkit. User's manual, Graphics, Visualization and Usability Center, Georgia Institute of Technology, September 1996.

[Hughes, 1993] David J. Hughes. Mini SQL ("msql"), 1993. `http://www.bond.edu.au/Bond/Admin/ ITS/People/bambi/mSQL/`.

[Hutchins *et al.*, 1986] Edwin Hutchins, James Hollan, and Donald Norman. Direct manipulation interfaces. In Donald Norman and Stephen Draper, editors, *User Centered System Design*, pages 87–124. Erlbaum, Hillsdale, NJ, 1986.

[Ingwersen, 1992] Peter Ingwersen. *Information Retrieval Interaction*. Taylor Graham, 1992.

[Jennings and Higuchi, 1993] Andrew Jennings and Hideyuki Higuchi. A user model neural network for a personal news service. *User Modeling and User-Adapted Interaction*, 3:1–25, 1993.

[Kamba *et al.*, 1996] Tomonari Kamba, Krishna Bharat, and Michael C. Albers. An interactive, personalized newspaper on the WWW. In *Proceedings of Multimedia Computing and Networking*, 1996.

[Karakoulas and Ferguson, 1995] Grigoris J. Karakoulas and Innes A. Ferguson. A computational market for information filtering in multi-dimensional spaces. In *Proceedings of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, 1995.

[Kautz *et al.*, 1997] Henry Kautz, Bart Selman, and Mahul Shah. ReferralWeb: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, March 1997.

[Kautz *et al.*, 1998] Henry Kautz, Marko Balabanović, Kristian J. Hammond, Haym Hirsh, Joseph A. Konstan, Alexandros Moukas, and Loren Terveen, editors. *AAAI-98 Workshop on Recommender Systems (to appear)*. American Association for Artificial Intelligence, July 1998.

[Kemeny and Snell, 1962] J.G. Kemeny and J.L. Snell. *Mathematical models in the social sciences*. Blaisdell, New York, 1962.

[Kernighan and Lin, 1970] B. Kernighan and S. Lin. An efficient heuristic for partitioning graphs. *Bell System Technical Journal*, 29:291–307, 1970.

[Kibler and Aha, 1987] Dennis Kibler and David W. Aha. Learning representative exemplars of concepts: An initial case study. In *Proceedings of the 4^{th} International Workshop on Machine Learning*, 1987.

[Konstan and Bharat, 1996] Joseph A. Konstan and Krishna Bharat, editors. *Workshop on Integrating Personal and Community Recommendations in Collaborative Filtering*. 6^{th} Conference on Computer-Supported Cooperative Work, November 1996.

[Koster, 1994] Martijn Koster. A standard for robot exclusion, June 1994. `http://info.webcrawler.com/ mak/projects/robots/robots.html`.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, 1995.

[Lesk and Salton, 1971] M. E. Lesk and G. Salton. Relevance assessments and retrieval system evaluation. In *The Smart System—Experiments in Automatic Document Processing*, pages 506–527. Prentice Hall Inc., 1971.

[Lewis and Catlett, 1994] D.D. Lewis and J. Catlett. Heterogenous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 148–156, 1994.

[Lewis and Gale, 1994] D.D. Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[Lewis *et al.*, 1996] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–306, August 1996.

[Lieberman, 1995] Henry Lieberman. Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, August 1995.

[Malone *et al.*, 1987] T. W. Malone, K. R. Grant, F. A. Turbak, S. A. Brobst, and M. D. Cohen. Intelligent information sharing systems. *Communications of the ACM*, 30(5):390–402, May 1987.

[Maltz and Ehrlich, 1995] David Maltz and Kate Ehrlich. Pointing the way: Active collaborative filtering. In *Conference on Human Factors in Computing Systems (CHI '95)*, May 1995.

[Menczer, 1998] Filippo Menczer. Adaptive information agents in distributed textual environments. In *Proceedings of the 2^{nd} International Conference on Autonomous Agents*, February 1998.

[Miller *et al.*, 1997] B. Miller, J. Riedl, and J. Konstan. Experiences with GroupLens: Making usenet useful again. In *Proceedings of the 1997 Usenix Winter Technical Conference*, January 1997.

[Mitchell, 1995] Tom Mitchell. Personal communication, 1995.

[Mittelstaedt, 1971] Robert A. Mittelstaedt. Semantic properties of selective evaluative adjectives: Other evidence. *Journal of Marketing Research*, 8:236–237, May 1971.

[Monier, 1996] Louis Monier. Alta Vista. Stanford Digital Library Seminar, October 1996.

[Morita and Shinoda, 1996] Masahiro Morita and Yoichi Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the 19^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 272–281, August 1996.

[Moukas and Zacharia, 1997] Alexandros Moukas and Giorgos Zacharia. Evolving a multiagent filtering solution in Amalthaea. In *Proceedings of the 1^{st} International Conference on Autonomous Agents*, pages 394–403, February 1997.

[Mullen and Wellman, 1995] Tracy Mullen and Michael P. Wellman. A simple computational market for networked information services. In *Proceedings of the 1^{st} International Conference on Multiagent Systems*, June 1995.

[Ng *et al.*, 1997] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perceptron learning, and usability case study for text categorization. In

Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 67–73, July 1997.

[Nielsen, 1994] Jakob Nielsen. Guerilla HCI: Using discount usability engineering to penetrate the intimidation barrier. In Randolph G. Bias and Deborah J. Mayhew, editors, *Cost-Justifying Usability*, pages 245–272. Academic Press, Boston, MA, 1994.

[Oard, 1997] Douglas W. Oard. The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, 7(3):141–178, 1997.

[Page and Brin, 1998] Lawrence Page and Sergey Brin. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World-Wide Web Conference*, April 1998.

[Pazzani *et al.*, 1996] Michael Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & Webert: Identifying interesting web sites. In *Proccedings of the 13th National Conference on Artificial Intelligence*, August 1996.

[Porter, 1980] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[Resnick and Miller, 1996] Paul Resnick and Jim Miller. PICS: Internet access controls without censorship. *Communications of the ACM*, 39(10):87–93, 1996.

[Resnick and Varian, 1997] Paul Resnick and Hal R. Varian. Recommender systems: Introduction. *Communications of the ACM*, 40(3):56–58, March 1997.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1994.

[Rich and Knight, 1991] Elaine Rich and Kevin Knight. *Artificial Intelligence*. McGraw Hill, 1991.

[Rich, 1979] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

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

[Robertson, 1977] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, December 1977.

[Robinson, 1996] David Robinson. The WWW common gateway interface version 1.1. IETF Internet Draft, February 1996.

[Rocchio, 1971a] J.J. Rocchio, Jr. Performance indices for document retrieval systems. In *The Smart System—Experiments in Automatic Document Processing*, pages 57–67. Prentice Hall Inc., 1971.

[Rocchio, 1971b] J.J. Rocchio, Jr. Relevance feedback in information retrieval. In Gerard Salton, editor, *The Smart System—Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall Inc., 1971.

[Rorvig, 1988] Mark E. Rorvig. Psychometric measurement and information retrieval. In Martha E. Williams, editor, *Annual Review of Information Science and Technology*, volume 23, pages 157–189. Elsevier Science Publishers B.V., 1988.

[Rose *et al.*, 1993] Daniel E. Rose, Richard Mander, Tim Oren, Dulce B. Ponceleón, Gitta Salomon, and Yin Yin Wong. Content awareness in a file system interface: implementing the "pile" metaphor for organizing information. In *16th International*

ACM/SIGIR Conference on Research and Development in Information Retrieval, pages 260–269, July 1993.

[Rosenblatt, 1960] F. Rosenblatt. On the convergence of reinforcements in simple perceptrons. Technical Report VG-1196-G-4, Cornell Aeronautical Laboratory, Ithaca NY, 1960.

[Rucker and Polanco, 1997] James Rucker and Marcos J. Polanco. Siteseer: Personalized navigation for the web. *Communications of the ACM*, 40(3):73–75, March 1997.

[Sakagami and Kamba, 1997] Hidekazu Sakagami and Tomonari Kamba. Learning personal preferences on online newspaper articles from user behaviors. In *Proceedings of the 6^{th} International World-Wide Web Conference*, April 1997.

[Salton and Buckley, 1990] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, June 1990.

[Salton and McGill, 1983] Gerard Salton and Michael J. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[Saracevic, 1975] Tefko Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science*, 26(6):321–343, 1975.

[Saracevic, 1995] Tefko Saracevic. Evaluation of evaluation in information retrieval. In *18^{th} International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 1995.

[Schamber and Bateman, 1996] Linda Schamber and Judy Bateman. User criteria in relevance evaluation: Toward development of a measurement scale. In *Proceedings* *of the Annual Meeting of the American Society of Information Science*, pages 218–225, October 1996.

[Schütze *et al.*, 1995] Hinrich Schütze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 1995.

[Seltzer and Yigit, 1991] Margo Seltzer and Ozan Yigit. A new hashing package for UNIX. In *Proceedings of the USENIX Technical Conference*, Winter 1991.

[Shardanand and Maes, 1995] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *Conference on Human Factors in Computing Systems (CHI '95)*, May 1995.

[Shardanand, 1994] Upendra Shardanand. Social filtering for music recommendation. Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1994.

[Sheth and Maes, 1993] Beerud Sheth and Pattie Maes. Evolving agents for personalized information filtering. In *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications*, March 1993.

[Shipman and Marshall, 1994] Frank M. Shipman and Catherine C. Marshall. Formality considered harmful: Experiences, emerging themes and directions. Technical Report ISTL-CSA-94-08-02, Xerox Palo Alto Research Center, 1994.

[Spertus, 1997] Ellen Spertus. ParaSite: Mining structural information on the web. In *Proceedings of the 6th International World-Wide Web Conference*, April 1997.

[Spink, 1993] Amanda Spink. Interaction with information retrieval systems: Reflections on feedback. In *Proceedings of the 56th Annual Meeting of the American Society for Information Science*, pages 115–121, October 1993.

[Stadnyk and Kass, 1991] Irene Stadnyk and Robert Kass. Modeling decision making of Usenet news readers. Technical Report CFAR-91-003, EDS Center for Advanced Research, September 1991.

[Stanford digital library group, 1995] Stanford digital library group. The Stanford digital library project. *Communications of the ACM*, 38(4):59–60, April 1995.

[Stephenson, 1967] W. Stephenson. *The play theory of mass communication.* University of Chicago Press, 1967.

[Su, 1992] Louise T. Su. Evaluation measures for interactive information retrieval. *Information Processing & Management*, 28(4):503–516, 1992.

[Sumner and Shaw, 1996] Robert G. Sumner, Jr. and W.M. Shaw, Jr. An investigation of relevance feedback using adaptive linear and probabilistic models. In *Proceedings of the 5th Text REtrieval Conference*, November 1996.

[Taylor, 1962] Robert S. Taylor. The process of asking questions. *American Documentation*, 13(4):391–396, October 1962.

[Terveen *et al.*, 1997] Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, March 1997.

[Tiamiyu and Ajiferuke, 1988] Mutawakilu A. Tiamiyu and Isola Y. Ajiferuke. A total relevance and document interaction effects model for the evaluation of information retrieval processes. *Information Processing & Management*, 24(4):391–404, 1988.

[van Rossum, 1995] Guido van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Netherlands, May 1995.

[Wang and Soergel, 1996] Peiling Wang and Dagobert Soergel. Beyond topical relevance: Document selection behavior of real users of IR systems. In *Proceedings of the 59th Annual Meeting of the American Society of Information Science*, pages 87–92, October 1996.

[Watters *et al.*, 1998] C. R. Watters, M. A. Shepherd, and F. J. Burkowski. Electronic news delivery project. *Journal of the American Society for Information Science*, 49(2):134–150, 1998.

[Wittenburg *et al.*, 1995] Kent Wittenburg, Duco Das, Will Hill, and Larry Stead. Group asynchronous browsing on the world wide web. In *Proceedings of the 4th International World Wide Web Conference*, December 1995.

[Wong and Yao, 1990] S.K.M. Wong and Y.Y. Yao. Query formulation in linear retrieval models. *Journal of the American Society for Information Science*, 41(5):334–341, 1990.

[Wong *et al.*, 1988] S.K.M. Wong, Y.Y. Yao, and P. Bollman. Linear structure in information retrieval. In *Proceedings of the 11th International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 219–232, June 1988.

[Wong *et al.*, 1991] S.K.M. Wong, Y.Y. Yao, G. Salton, and C. Buckley. Evaluation of an adaptive linear model. *Journal of the American Society for Information Science*, 42(10):723–730, 1991.

[Yan and Garcia-Molina, 1995] Tak W. Yan and Hector Garcia-Molina. SIFT—a tool for wide-area information dissemination. In *Proceedings of the USENIX Technical Conference*, pages 177–186, January 1995.

[Yang, 1994] Yiming Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22, July 1994.

[Yang, 1997] Yiming Yang. An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-97-127, Carnegie Mellon University, School of Computer Science, April 1997.

[Yao, 1995] Y. Y. Yao. Measuring retrieval effectiveness based on user preference of documents. *Journal of the American Society for Information Science*, 46(2):133–145, 1995.