INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Personalized News Agent

By

Yue Bao

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Science
Information and Systems Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

April, 2005

Abstract

While Internet use has been increasing, the overwhelming amount of information on the web can make it difficult to find relevant information. People spend more time searching for useful information than they do absorbing the material in which they are interested.

This thesis identifies information overload as the main problem, and proposes a personalized news agent to improve the efficiency and effectiveness of relevant knowledge acquisition and dissemination from online web sources. We analyze the design of the personalized news agent, focusing on issues such as information filtering schema, user profile structures, feedback technology and recommendation methods. Five components are defined to make up our news agent and a prototype news agent is implemented to demonstrate the capabilities of the proposed system.

In our agent, both explicit and implicit feedback are tracked and analyzed to record shifts in the user's interests. The user profile in our agent is represented using a 3-descriptor scheme, which can represent both short-term and long-term user interests. The agent improves its ability to learn user interests by adjusting its learning rate. Finally, an admin module is created in our agent to verify changes in user profiles.

Acknowledgements

I wish to sincerely thank Professor Weiss, my supervisor, for his kind guidance and support along the development of this thesis. Thank him for providing me with this opportunity to benefit from his knowledge.

My special thanks go to many of my friends who have supported and encouraged me during my graduate studies. I would especially like to thank Haibo Jiang, Hua Ye, Zhihong Li, Wanjun Sun, and Xingrong Huang.

I am grateful to my family for their consistent support and understanding. Thanks to my parents Zhili Xu and Gongshun Bao, my son Leo, for their encouragement and support all through my life. Special thanks to my dear husband Xuanang Zhu for his love, encouragement and patience throughout my study. Without his help, it would be much difficult for me to reach this point.

Ottawa, Ontario Yue Bao

March, 2005

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and motivation

Traditionally, the editor of the newspaper determines the extent to which reports reflect the interests of most readers, and then presents stories of high interest in the headlines. All readers view the same set of headlines, regardless of their level of interest. The rapid propagation of the Internet has altered the nature of traditional business. The Internet provides an excellent platform to deliver content information such as news. Since online information resources are continuing to grow, a typical user faces repetitive tasks such as searching, browsing and filtering to acquire relevant information from the Web. It is increasingly difficult for users to find information that satisfies their individual needs. The abundance of news providers forces people to spend more time sifting through potential information sources than they do absorbing the material that actually interests them.

Using existing search engines, relevant information is easier to obtain. However, getting to the right information is still a time-consuming task because a considerable amount of time is required to verify whether the information is relevant[33]. People

need software agents that explore information and deliver only the most relevant documents to them. News agents and other Internet content providers personalize services that offer content to different clients based on their interests.

A news agent is an application that maintains the profiles of its users as representations of their interests. It automatically gathers news articles from online newspapers during the night. When a user logs on to the system, the agent creates relevant news headlines and personal news pages based on the user's profile. Moreover, the agent keeps track of user behavior in order to detect shifts in user interests. The agent can learn which articles are read and how much time is spent on them by observing the user's actions. The agent can then adjust the user interests stored in the profile. Users can also provide explicit feedback, such as whether they like or dislike the articles they have read. The system then modifies the user's profile based on this feedback.

The main objectives of this thesis are to investigate the various techniques used by news agents, and to propose an architecture and implement a prototype of a personalized news agent that provides the latest articles from a news source, such as the Canada.com site, based on users' interests. Key considerations in designing a personalized news agent include the system architecture and components, document extraction and analysis from a variety of data sources, feedback methods (explicit and implicit), the structure of the user profile, and the recommendation methods.

1.2 Thesis contributions

This thesis identifies information overload as the main problem, and proposes an information filtering system to improve the efficiency and effectiveness of relevant

knowledge acquisition and dissemination from online web sources. In this thesis, we discuss the issues of creating a best of practice news agent. After selecting suitable techniques, a prototype of the news agent is then implemented to validate those techniques.

1. We analyze the design of creating a personalized news agent. In order to create a best-of-practice news agent, we should consider four parts of technologies: information filtering schema, user profile structure, feedback technology and recommendation methods. Each part of technology has several different approaches. Selecting a suitable approach for each part is important to build a news agent. In the thesis, we analyze the advantage and disadvantage of the different methods in those issues, select suitable approaches and combine them to create a personalized news agent.

2. The basic architecture of a news agent is discussed in this thesis. The news agent is consists of five modules: the retrieval module, the learning module, the recommendation module, the user interface, and the admin module. The retrieval module extracts relevant information from a variety of news sources. The learning module learns and maintains user interests, while the recommendation module creates recommendations for the user. The user interface facilitates interaction between the user and the system, and the admin module monitors user behavior and adjusts the configuration. The core module in our news agent is the learning module. It maintains the user profile and learns the user's short-term and long-term interests according to the explicit and implicit feedback.

3. A prototype of the personalized news agent is implemented to provide the user

with relevant articles from online news sources. In order to implement a best-of-practice news agent, both the short-term and long-term interests are learned from either explicit or implicit feedback, and the agent is designed to automatically adjust the learning parameters to improve the accuracy of prediction. The feature of our news agent is listed as follows:

- The scheme and the algorithm work on a vector space model. The term frequency inverse document frequency (TFIDF) method is adopted as an efficient presentation tool to analyze the articles extracted from the website.
- A 3-descriptor scheme is used to present both the user's short-term and long-term interests. Both interests and disinterests are contained in this schema.
- Content-based recommendations are used to improve the quality and relevance of information items.
- Both explicit feedback and implicit feedback are collected by our news agent. The user can explicitly provide an opinion using five ranks to score the article read. The system also observes the activities of the user (e.g. clicking the article, time spent reading the article) to generate implicit feedback.
- The Least Mean Squares (LMS) method is used to adjust the learning rate in order to improve the ability to learn user interests.
- An admin module is implemented to verify and monitor changes to the user profile.

1.3 Thesis organization

The remainder of this thesis is organized as follows:

Chapter 2 reviews the technologies used to design the news agent. It introduces the basic architecture of a retrieval/filtering system, the concept and method of document analysis, the two types of feedback (explicit and implicit), and the recommendation methods.

Chapter 3 describes the basic architecture and processes of our news agent. It describes the basic function for each of component, and introduces the system environment.

Chapter 4 describes the structure of the user profile and the design of the database. The user profile model and the learning process are important parts of a personalized news agent. First, this chapter describes the profile structure based on a 3-descriptor scheme. Then, corresponding to this schema, we present the design of the database for our agent. Finally, this chapter describes the learning process and the algorithm used to maintain and dynamically update the user profile.

Chapter 5 focuses on implementation aspects of the personalized news agent. Our system is composed of five components. This chapter describes the implementation in detail. First, we discuss system initialization, and introduce the method of extracting articles from the news web site. Then, we describe how articles are recommended and present the implementation of the learning module. Finally, this chapter presents examples of the user interface of the personalized news agent and describes how the admin module monitors the results of the system.

Chapter 6 presents the results of system tests. The procedure of testing how the rating influences the system, and the normalized distance-based Performance Measure

method are introduced.

In Chapter 7, we present our conclusions and plans for future work.

Chapter 2

Related Technology

The huge amount of information available on the Internet has forced users to spend a significant amount of time sifting through large volumes of material to find relevant and interesting information. This difficulty is commonly referred to as information overload. In order to protect users from information overload, Information Retrieval (IR) and Information Filtering (IF) systems are used to retrieve information relevant to the user's interests.

Generally, IR systems retrieve the selected data from a fixed data set while IF systems select the relevant information from a stream of incoming data over time[23]. Information retrieval focuses on the user's short term query while filtering focus on the user's long term interests, as stored in the user profile.

An intelligent IF system must satisfy the following three requirements[6]:

- Customization: it should be highly responsive to the particular interests of users. It should remember all users and adapt its performance to meet the needs of each user.

- Adaptation: it should exploit user feedback to identify changing individual preferences based on explicit indications and detection of system usage patterns.

In other words, the system should be able to follow user interests over time through the interaction between the system and user.

- Exploration: it should be able to explore interesting information and recommend items to users based on their interests.

The first section in this chapter presents the basic architecture of the retrieval / filtering system. The second section introduces some techniques used to document text analysis and measuring similarities. The third section discusses technologies used to obtain feedback. The recommendation methods are introduced in the fourth section, while some examples of news agents are given in the final section.

2.1 Architecture of retrieval/filtering system

The basic architecture of a retrieval / filtering system is given in Figure 2.1. As shown here, the system has three main components: representation, comparison and feedback[23]. Both the document set and the user's information are represented in a language that can be compared effectively by the system. Using the comparison algorithm, the query or user profile is compared with the document sets, producing the retrieved or filtered texts. The feedback mechanism is then used to return the texts to modify the query and user profile, thereby improving system performance. The feedback process can be automatic or manual.

Generally, the document set is generated by extracting data from the Internet through the retrieval/filter system. HTML files, in which most web page data is

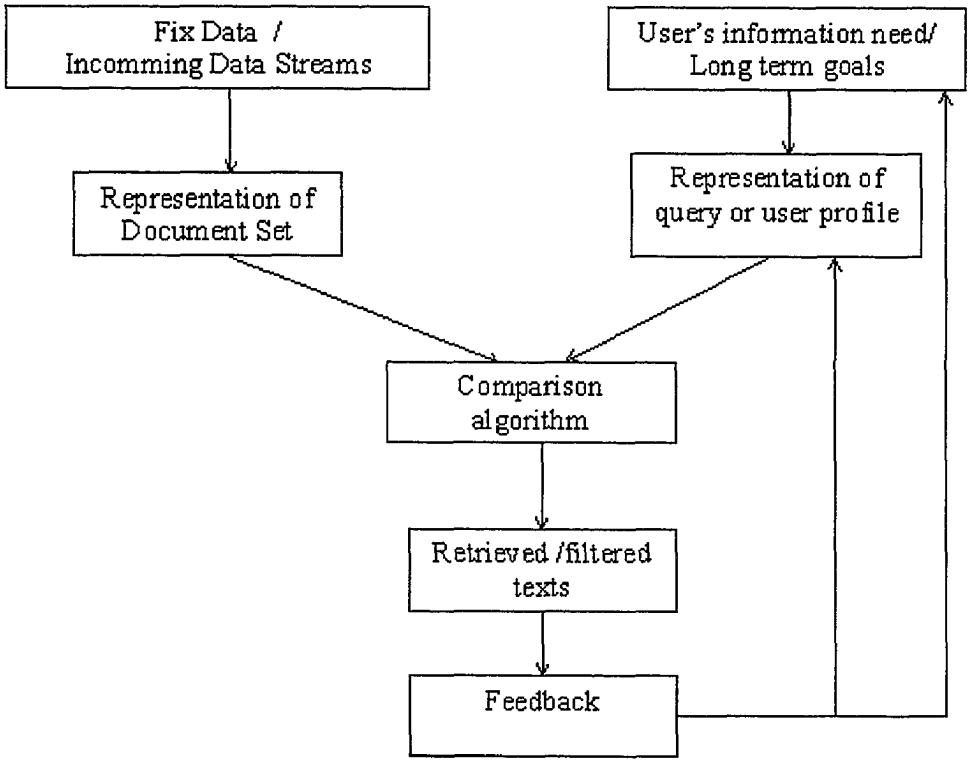


Figure 2.1: Architecture of retrieval/filtering system

stored, describe how the data is displayed rather than what it means[4]. This property poses a challenge for extracting information from web pages in data driven applications. However, since most information sources on the web are semi-structured, it is still possible to extract useful data. The data in HTML files are always wrapped in specific tags that can be identified by analyzing the source code[11]. This feature makes it possible to extract information from web pages. Once the data has been extracted, we can use the technology introduced below to represent and analyze documents.

2.2 Document text analysis and measuring similarity between documents

There are two models used to represent a document: the boolean model and the vector space model. The boolean model is popular because it can be understood easily for simple queries. In this model, a document is only represented as a set of words or terms. However, it is difficult to express user requests when a user needs complex information, and a challenge to process relevance feedback. In a news agent, it is better to use the vector space model, also called the statistical model, to represent a document. In this model, a document is typically represented by an n-dimension vector in which each dimension represents a distinct term extracted from the document[33].

A vector space is an n-dimensional space. Each dimension represents a term. Figure 2.2 shows an example with 3 terms and several documents. Each document is a point in this vector space.

This model raises two issues: how to weight the terms and how to determine the degree to which two documents (or the related vectors) are similar. The choice of weighting schema and measures of similarity are very important in designing a vector space model.

2.2.1 Weighting terms method

The important task in textual document analysis is to extract terms from a document and assign them weights. Since an author usually repeats certain words when presenting ideas, measuring word importance in a document by use-frequency

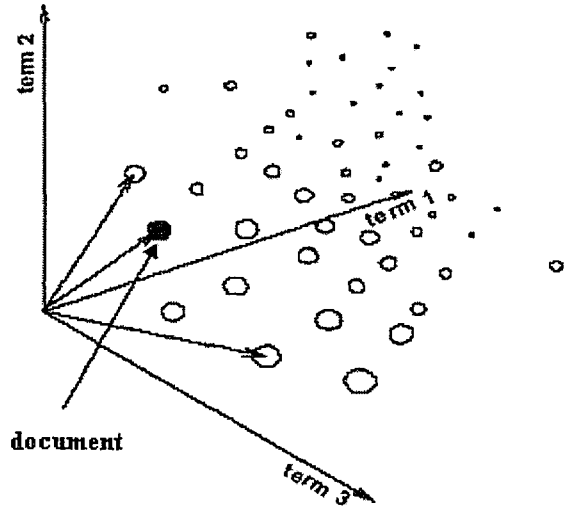


Figure 2.2: Illustration of vector space models

is justified[15]. Sorting the terms based on decreasing frequency, we can create three sets of terms (high, middle, low frequency). Based on Zipf's law, which states that "a product of frequency of a term and its rank order will be approximately equal to the product of the frequency of another term and its rank"[38, 33], the most relevant terms for identifying content in a document would be in the middle frequency set. Both extremely common and extremely uncommon words (the high and low frequency set) are not very useful for analyzing documents[15, 2],and eliminating common words such as "the", "a", and "is" greatly reduces storage costs.

Many methods have been contrived to weight document keywords, including term frequency, probabilistic weighting, inverse document frequency, signal weighting, discrimination value, term frequency inverse document frequency (TFIDF), and Latent Semantic Indexing (LSI)[12, 26, 9, 1]. The details of some of these methods are presented below.

(1) Normalized Term Frequency Weight method

In this method, the weight is derived directly from the document text itself without prior knowledge obtained from a document collection. The disadvantage of this method is that it does not take into account the context of the keyword in a document[33]. Given a document, there are two steps in calculating the weight of the keyword. The first step is to remove common words and stemming words in the document. The second step is to calculate the weight of each term using Equation 2.1. Let's define w_i^d as the weight of keyword i in document d. The weight of the keyword is computed from the frequency of keyword occurrence f_i^d divided by the length of document.

$$w_i^d = \frac{f_i^d}{\sum_{j=1}^{n} f_j^d} \qquad (2.1)$$

$\sum_{j=1}^{n} f_j^d$is denoted as the length of the document, which is the total number of keywords extracted from the document. Therefore, a document can be denoted as a feature vector represented by $f_d = (k_1, w_1), (k_2, w_2), \ldots, (k_n, w_n)$, where k_i is keword i and w_i is the weight of keyword.

(2)TFIDF method

TFIDF is the most popular keyword weighting method used in information retrieval. Keywords with smaller document frequencies will be assigned higher weights than those with larger document frequencies, since it is assumed that keywords that appear in fewer documents discriminate better than those appearing in more documents. In other words, if a term occurs frequently in a given document but rarely in other documents in the document collection, it will be given a high weight[26, 33].

The weight of keyword i in document d is a product of the term frequency and inverse document frequency as shown in Equation 2.2.

$$w_i^d = tf_i^d \cdot idf_i$$
$$idf_i = log\frac{N}{df_i} \qquad (2.2)$$

where tf_i^d is the term frequency, the number of term occurrences in the document divided by the length of document, idf_i is the inverse document frequency, N is the number of documents in the collection, and df_i, the document frequency of term i, is defined as the number of documents in the document collection that contain the term.

Since this method requires a collection of documents to compute the keyword weighting, it would be difficult to extend to a dynamically changing collection of documents.

(3) LSI method

The method of weighting keywords discussed above still suffers from the problems of synonymy and homonymy. The terms occurring in a document are not only a single feature set, but are also related within the document. Synonymy refers to the use of various terms to describe the same object[23]. Different authors, based on their habits and knowledge backgrounds, may use different words to refer to the same information. Even the same author may use different words to state the same thing in different contexts. For example, the word "restaurant" may be used instead of "café" or the words "notebook", "laptop", "portable computer" may be used interchangeably. Homonymy refers to words that have more than one meaning[23]. In this case, it is possible that two documents contain the same keyword but are not relevant. For

instance, the word "shock" has different meanings in the fields of medicine, where it means the loss of full awareness, and mechanical engineering, where it refers to a strong wave.

Latent Semantic Indexing (LSI) based on standard vector retrieval methods is designed to solve the synonymy and homonymy problems. Considering the latent interconnections between terms, LSI attempts to decompose the matrix (terms of the document) using singular value decomposition (SVD) to an approximation of the original matrix[1, 6]. In other words, using SVD reduces the n-dimensional space to a k-dimensional one that keeps the k largest orthogonal factors to best approximate the original matrix. Using this method, the terms are no longer orthogonal and the synonyms will have the same dimensions or similar directions. Thus, although documents with similar term usage patterns do not share common terms, they will still score highly when measuring the similarity between them. The LSI method leads to more efficient comparisons by reducing vector size, but it is difficult to determine the k number of dimensions.

2.2.2 Measuring Similarity between documents

A similarity measure is a function that computes the closeness between two documents. The most commonly used similarity function is the cosine similarity measure[26], which calculates the difference between two vectors by calculating the cosine of the angle between them. Let's denote f_{di} and f_{dj} as vectors of two documents. The similarity between the two documents can be computed as the vector inner product:

$$Sim(f_{di}, f_{dj}) = \frac{\vec{f_{di}} \bullet \vec{f_{dj}}}{|\vec{f_{di}}| \cdot |\vec{f_{dj}}|} = \frac{\sum f_{di} \cdot f_{dj}}{\sqrt{\sum f_{di}^2} \sqrt{\sum f_{dj}^2}} \qquad (2.3)$$

For example, let's denote the feature vector of one document as f_{d1}={(term1,2), (term2,3), (term3,5)}, and denote the feature vector of the other document as f_{d2}={ (term1,0), (term2,0), (term3,2)}. The similarity between these two document can be calculated as follows:

$Sim(f_{d1}, f_{d2}) = \frac{2\times0+3\times0+5\times2}{\sqrt{2^2+3^2+5^2}\sqrt{0^2+0^2+2^2}} = 0.81$

If the value of the measure is close to zero, it means that the two documents are totally dissimilar. As the cosine value increases, the similarity of the two documents also increases. If the cosine value is close to one, the two documents are considered relevant.Since the cosine measure method measures how many terms in two documents matches, it is suffers from the problems of synonymy and homonymy. Also, it does not scale to large vectors.

2.3 Feedback technology

Relevance feedback techniques are used to improve information filtering and retrieval. Users may provide relevance judgments for the article they receive. The relevance feedback information is employed to re-construct user profiles. Two different feedback modes, explicit and implicit, are used to accomplish this.

Explicit feedback requires users to explicitly evaluate the recommended articles. A typical explicit feedback mechanism is to ask the user to rate the article after reading. Based on the feedback of the user and the current user profile, a new user profile could be created. This mechanism is used in the Intelligent News Filtering Organization System (INFOS) presented by Mock and Vemuri[18]. News filtering and recommendations can also be performed based on feedback from others. For example, the GroupLens system[22, 7, 28], an article recommendation system for Usenet news,

combines a hybrid collaborative filtering system with content-based filters named filterbots. Filterbots evaluate the new articles as they are published and the users rate the articles on a numerical scale (1-5), where a score of 5 means the article is relevant and worth reading. User ratings are posted to newsgroups as recommendations to other users. With summarized feedback from previous readers, the next reader can determine whether the article should be read. The only requirement that users must do is to rate some articles, so that profiles can be built for them. The more documents the user rates, the better the recommendation the user receives.

The advantage of explicit feedback is that the agent is confident about the information it provides to users and appropriate action can be taken. When the user gives a strong positive or negative rating of the article, it should result in a significant change to the user profile. For example, if the user does not like the article recommended by the agent, similar information should not appear in future recommendations. The problem with explicit feedback is that it requires more effort on the part of the user. If the user does not rate the article, the recommendation is poor.

Implicit feedback is a behavior-based technique. A news agent gives the user a list of article titles with sample sentences from the first paragraph of the article, so the user is able to predict roughly the content of the articles. If the user selects and reads the articles on the list, it may imply that he may be interested in the article. When the user reads the article, the system can also track behaviors such as scrolling, peeking, maximizing, opening articles in new windows, or saving information to a scrapbook, and can even record the time a user spends on an article. These activities probably mean a user is interested in that article. For example, ANATAGONOMY[25], a personalized on-line newspaper, learns reading preferences from the browsing behavior

of a user. The system is composed of a scoring engine and a learning engine. The scoring engine computes the importance of each article by comparing the article's document vector, representing words and their frequencies of occurrence, and the user profile. The learning engine learns user profiles. When the user scores a set of articles showing how interesting each article is, it builds a user profile.

Hung-Jen Lai, Ting-Peng Liang, and Y.C. Ku propose to build customer profiles from their browsing behavior by focusing on the time a user spends reading an article[13]. If the reader spends a reasonable amount of time to read the article, he may be interested in it, since users tend to spend more time reading interesting articles than uninteresting ones. In their system, the computer records the beginning and ending time that a user reads an article, and calculates the average reading speed of the user and the interest level of a user on the article. Based on the keywords of the article, the interest level, and the current user profile, the system will re-construct the user profile. However, if the reader leaves the terminal to take a phone call or to read newly arrived e-mail messages, the accuracy of the user profile may suffer. The advantage of implicit feedback is that it makes the agent more friendly and human. However, this mechanism inspires less confidence than explicit feedback because of the many exception conditions. Protection of privacy is also an issue in implicit feedback.

Meanwhile, user feedback can be positive or negative, indicating that users like or dislike the article. Table 2.1 illustrates some information filtering system based on the feedback type[34].

	Feedback			
	Positive	Negative	Explicit	Implicit
McEligot et.al.	X		X	
Wiener et.al.	X		X	
Menczer et.al.	X	X	X	
INFOS	X	X	X	
Fab	X	X	X	X
WebMate	X		X	
PIN	X	X	X	X
Amalthea	X	X	X	
NewT	X	X	X	

Table 2.1: Systems based on the types of feedback

2.4 Recommendation Methods

In recent years, recommendation systems have been used in many applications due to the growth of the web and the potentially overwhelming number of choices it offers. Using personalized information filtering technologies, the recommendation system can predict whether a particular user is interested in a particular item, or recommend some items that will interest the user. Typically, recommendation systems mainly apply two approaches: content-based filtering and collaborative filtering.

Content-based filtering compares the similarities between incoming items and user profiles and makes recommendations. The user's profile is maintained based on an analysis of the contents of the documents that the user has previously rated. The

user profile is represented as a feature vector containing keywords he may be interested in. The method of representing a document and measuring similarities between documents and user profiles was introduced in the previous section. Content-based filtering is good at text-based filtering. For example, SIFT is a content-based filtering system that filters information and identifies potentially interested Internet news users through user profiles[36]. However, only certain kinds of content can be analyzed. This application ignores information such as multimedia, text embedded in images, and network factors. Since user profiles in a content-based system are generated from scoring recommended items, the user is restricted to seeing items similar to those that have already been rated.

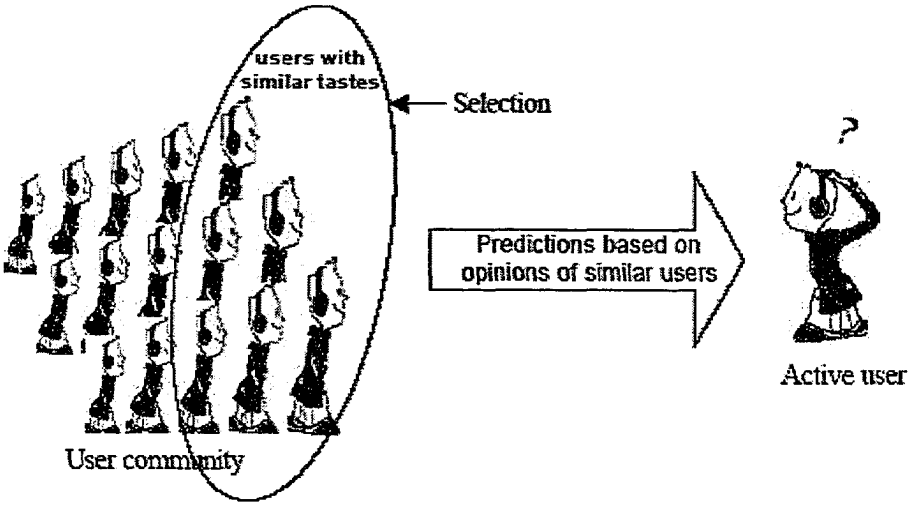


Figure 2.3: Collaborative filtering system

Collaborative filtering is the most common social filtering based on relationships between people. It tries to recommend items favored by other users with similar tastes[5]. Figure 2.3 illustrates the idea of collaborative filtering, which builds a database of items preferred by users. Each user in a collaborative filtering system has

a profile that records scores for all rated items. When a new user joins, the system discovers neighbors with similar interests based on the degree of matching against the database. The items that these neighbors like will then be recommended to the new user.

The opinions of users can be obtained by explicit or implicit measures. Explicit opinions are given by the user as a rating score, while implicit opinions are derived by tracing user activity. Similar users are identified by comparing user profiles, and a neighborhood list is formed by selecting users with high similarities. The system then recommends the items that those neighbors liked. Tapestry [3]was the first collaborative filtering system. In the Tapestry system, users can use content filters to select articles and can also select articles based on the recommendations of other users.

The schematic diagram of the collaborative filtering process is shown in Figure 2.4. There is a list of users denoted by $U = u_1, u_2, \ldots, u_m$ and a list of items denoted by $I = i_1, i_2, \ldots, i_n$. Each user has a list of items I_{ui}. Using the collaborative filtering (CF) algorithm, the process will output two types of results (predictions and recommendations) according to the active user u_a. The prediction expresses the predicted preference of the active user for item j, while the recommendation is a list of N items that the active user is expected to like[27].

The CF algorithm can be memory-based (user-based) or model-based (item-based)[27, 10]. The scheme for user-based algorithms relies on the fact that each user belongs to a larger group of similarly behaving individuals. It uses statistical techniques to find a set of users with similar interests in the entire user-item database to generate a prediction or set of top-N recommendations for the active user. The scheme for

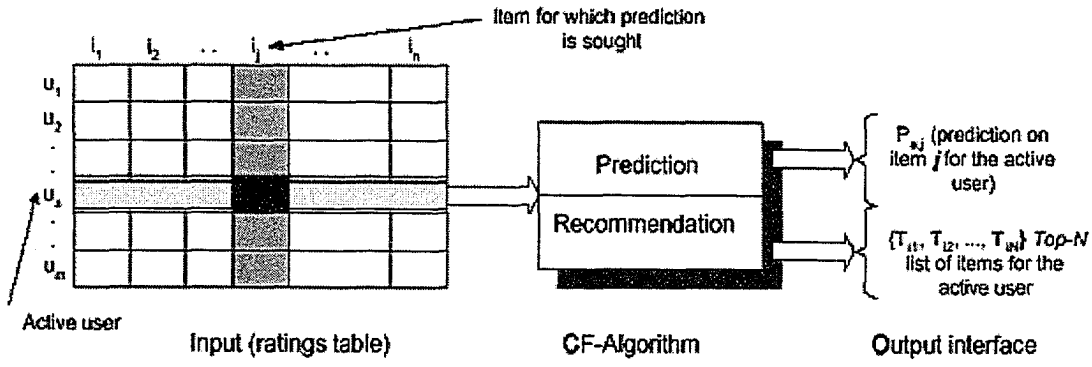


Figure 2.4: The collaborative filtering process

item-based algorithms is based on the fact that a user will more likely choose items that are similar or related to items he has already purchased.

Unlike the content-based system, a collaborative system can provide unexpected valuable content to the user and need not attempt a qualitative analysis of item content, since humans are much better at judging text qualitatively. However, collaborative filtering also has some problems:

- "Early rater problem": When an item appears for the first time in the system, since no user has tried it, it is possible that the item has no chance of being recommended by the system. Additionally, it is difficult for the system to find appropriate neighbors to give useful recommendations to a new user who has not yet rated any items.

- "Sparsity problem": When the amount of items available far exceeds the capability of users to rate them, it leads to the pool recommendation. For example, if a system contains millions of items and few people, the system generates a very large and sparse user-by-item matrix to contain the item's ratings, since different users often rate entirely different items. In this case, it will be fairly

difficult for the system to form a neighborhood.

- "Gray sheep problem": If the system is used by a small number of users, some users may not find a like-minded neighbor.

Since both content-based and collaborative filtering techniques have their own limitations, the best approach is to combine both to form a hybrid recommendation system. Generally, content-based techniques are used to build and maintain user profiles, and collaborative filtering techniques are applied on the user profiles to identify relevant users and produce recommendations[21]. In my thesis, content-based techniques are selected. The detail is discussed in chapter 4.

2.5 Examples of news agents

The techniques discussed in the previous sections can be used in industries that offer their clients digital content, such as consulting and news services. The Internet provides an excellent platform for news to be delivered, as a result of its advantages in timeliness, richness, and customization. All readers view the same set of headline news in traditional newspapers, but through news services on websites or through email delivery, e-newspapers can customize content and produce a personalized newspaper.

The personalized news agent is a computer program that performs information gathering and filtering on behalf of users. It maintains user profiles representing current user interests and retrieves relevant news from the World Wide Web according to the user profile.

In this section, we describe the definition and features of the news agent and give some examples.

(1) Agent paradigm

Agent technologies are an approach to developing complex applications applicable to many areas, including network management, business process management, information retrieval and management, and electronic commerce, etc[8].

According to the Shoham's definition[30], an agent is "a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes".

Agents have four characteristics[35]:

- Autonomy: Acting on behalf of users, agents can perform some tasks without the user's intervention.
- Reactivity: Agents interact with changes occurring in the environment, including through the user interface or from other agents.
- Pro-activeness: Agents exhibit goal-directed behavior to achieve their imposed goals.
- Social ability: In order to perform complex tasks, agents can interact with other agents to deal with problems they are unable to handle alone.

Agents can be categorized into collaborative agents, interface agents, mobile agents, information agents, reactive agents and hybrid agents[20]. The personal news agent has the characteristics of two different agent types: interface agents and information agents.

An interface agent assists users in operating an interactive interface[14]. In order to perform tasks for its owners, it monitors the activities of the user over a long period

of time. In the personalized news agent, the task of the interface module is to monitor the behavior of users to identify user interests and maintain user profiles.

Information agents are useful tools to deal with the information overload problem. The role of information agents is to manage, manipulate or collect information from a variety of distributed sources, such as the web[20, 32]. The personalized news agent can be viewed as information agent to some extent. It extracts the news article from the web and stores it in its own database.

(2) News agent examples

Some examples of news agents are given below:

- Mock presented the Intelligent News Filtering Organization System (INFOS) that effectively reduced the reader's load in selecting interesting articles from Usenet news articles[17]. In this system, it represents user interests by a feature vector. Using the content of the article and collaboration with other users, the system can predict what kind of articles a particular user will like.

- Tan and Teo developed Adaptive Resonance Associative Map techniques to personalize their news system called Personalized Information Network (PIN)[31]. In this system, they organize articles into categories according to their semantic similarities. With a relevance factor, a user profile is modeled by associating the information categories.

- Amalthea is a filtering information agent developed by Moukas and Zacharia[19]. It is composed of two different types of agents, an information filtering agent that tracks the user's interests, and an information discovery agent that retrieves

information likely to interest the user. User interests are encoded as a feature vector using the TFIDF method, and direct ratings from the user are used by the information discovery agent.

The systems discussed above assume that interests do not change during the evaluation process. However, in the real world, this assumption is not completely true. Changes in long-term and short-term interests can be distinguished. Furthermore, the way in which key factors used during the learning process, such as the learning rate and the relevance factor, are determined will influence the performance of the system. In this thesis, we implement a news agent which uses a 3-descriptor scheme to represent the user profile in order to track both long-term and short-term user interests, and the learning factor is automatically adjusted to improve the performance of the system. The basic architecture of the personalized news agent will be introduced in the next chapter.

Chapter 3

Personalized News Agent Overview

This chapter describes the architecture of the personalized news agent and its functional processes, based on the agent components. The system design and environment is also introduced.

3.1 Personalized news agent architecture

The personalized news agent is composed of five main components: the news retriever, learning module, recommendation module, user interface, and system admin module. The architecture of the agent is shown in Figure 3.1.

The function of the news retriever is to explore and collect web pages from online newspapers. This module periodically runs as a background process. It downloads articles from particular websites, then computes the keywords for each article and saves them to the database.

The learning module has two main functions. First, it builds and maintains the user profile. It creates a new user profile when a user fills out a subscription form. When the user provides explicit or implicit feedback on the articles, the module modifies the corresponding user profile. Second, it adjusts the learning rate according

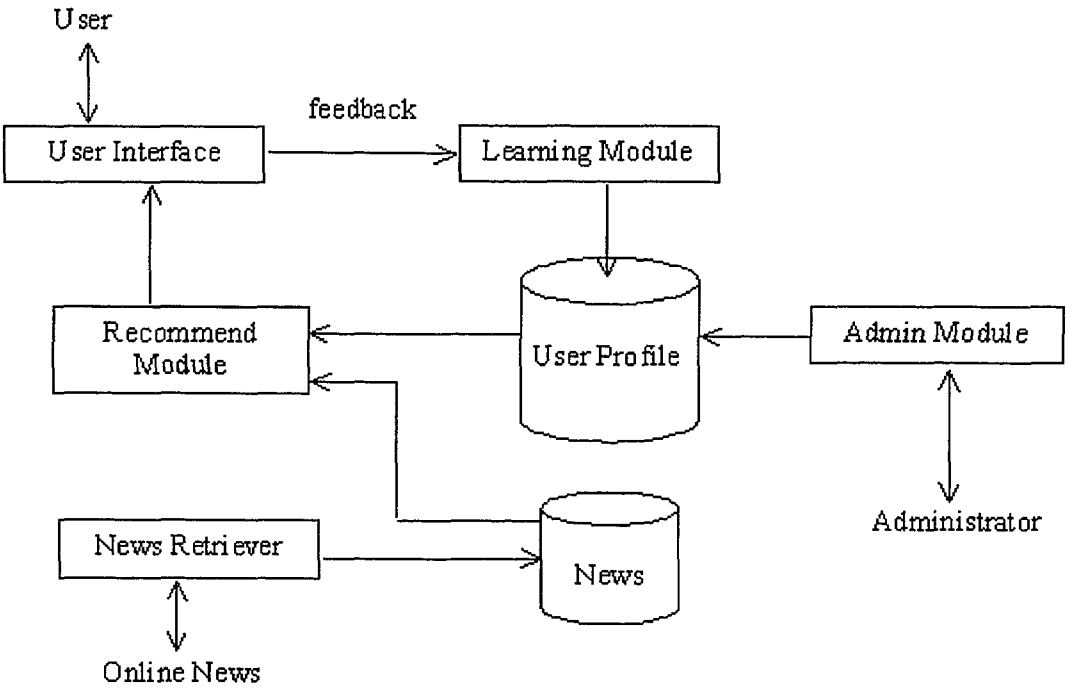


Figure 3.1: Architecture of News Agent

to the degree of satisfaction of the user with the articles the agent provided. The system randomly selects the user's rating for the article to adjust the learning rate by comparing the rating with the predicted rating for the article.

The recommendation module creates a personal news web page for each user according to the user profile using a recommendation algorithm. When a user logs on to the system, the user interface module sends a request to obtain the list of news articles. The recommendation module takes the user profile and latest news articles from the database. After computing the degree of interest for each news article, the recommendation module creates a personal news web page for the user.

The user interface manages the interaction between the user and the system. It provides a means for users to send a message to trigger a particular action.

The admin module is a tool administrators use to monitor user behavior and

configure the system parameters, such as number of keywords, learning rate, etc.

3.2 Basic process

There are two fundamental processes in the personalized news agent: the user-phase process and the admin-phase process.

In the user-phase process, the agent collects the user's feedback and provides a personalized news page for the user. New users first register in the system by completing a registration form that records some basic information and user interests. After the user logs on to the system, a display news request is sent to the recommendation module. The agent creates a list of articles that might interest the user and sends these to the user. While reading the articles, the user is able to provide explicit feedback. If the user rates an article, the rating is sent back to the agent and the learning module updates the user profile to adapt to the user's interests. If the user closes the window that displays the article without providing explicit feedback, the agent observes this behavior and returns an implicit rating, triggering the learning module to update the user profile. Figure 3.2 shows the sequence diagram for the user-phase process.

In the admin-phase process, the administrator first configures the system parameters, such as the learning rate and the value of the user's ratings. The agent then collects news articles periodically from online newspapers and magazines during the night, saves the meta content of those articles to its database, and updates the long-term interest information in the user profile based on the profile history record. The administrator also monitors dynamic changes in the user profile to adjust the system parameters.

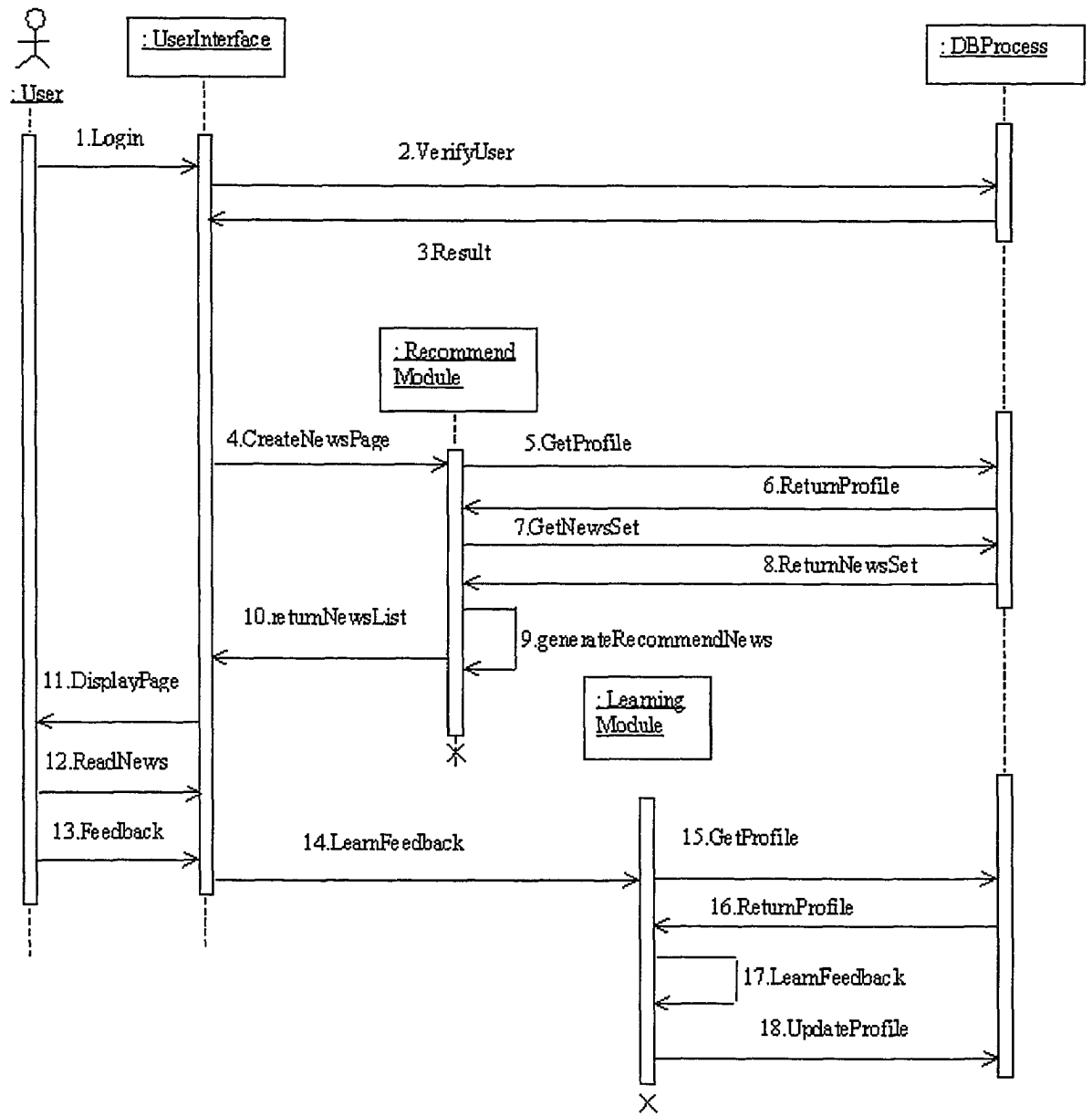


Figure 3.2: Sequence diagram for user-phase process

3.3 System environment

This news agent uses a four tier structural design made up of the client tier, web tier, application tier, and database tier. The client tier contains web pages generated by JSP running in the web tier. The web tier contains JSP programs which dynamically process requests and responses, and communicate with the client and application tier through the web server. The application tier contains the application developed using Java. It is responsible for exchanging data between the web and database tiers. The database tier stores system data. Figure 3.3 shows the four tiers of the system.

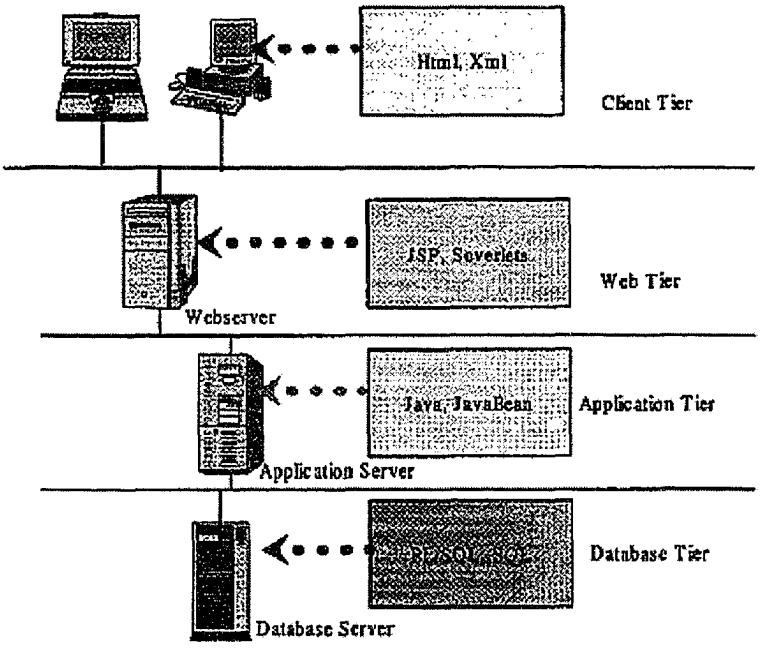


Figure 3.3: System tiers

In the implementation of the system, Tomcat 4.1.30, the servlet container, is installed as the web server. All components in the system are implemented in JAVA and can be run on any computer that supports JDK1.2 or above. The database of the

system is established under the HsqlDB. HsqlDB is a relational database engine. It is a pure-Java embedded relational database server that can be used in stand-alone mode (using direct file access) or in client/server mode. Using this database, developers do not need to install a processor, memory, or a disk-hungry database server in their development workstation. The system connects the database via JDBC.

Chapter 4

User Profile Model and System Design

The user profile model and the learning process are important parts of the personalized news agent. This chapter introduces the profile structure we used in the system and the corresponding design of the database. The learning process and algorithm are then described.

4.1 Structure of user profiles

Different user profile models are introduced in several information filtering systems. For example, some information filtering systems, such as INFOS, use a feature vector to represent user interests[17]. This vector is composed of a list of keywords and their weights. Some systems, such as NewT developed by Shelt, design a profile containing information about the URL addresses of articles, and other information, such as the author, number of lines, or news group category[29]. A profile defined by Nick and Dimitris classifies all terms into two groups[21]: "LONG-TERMS," which contain "heavier" terms, and "SHORT-TERMS," which contain "lighter" terms. Each term

is valued by weight, which indicates the importance of the term in the user's interests. The system gives a threshold to decide the term belong to which group. If the weight of a term is larger than the threshold, it will be considered as a "heavier" term, and vice versa. Depending on the positive or negative rating on a document, the weight of some terms will either increase or decrease. When the weight of a term exceeds the threshold, the term will be put into "LONG-TERMS" group. And the term will be put into"SHORT-TERMS" group when it falls below the threshold.

In this thesis, a 3-descriptor interest category scheme developed by Widyantoro is employed[33]. Since the user's interests always change over time in real world, it is better to present both the user's short-term and long-term interests in user profiles. This scheme allows handling long-term as well as short-term interests simultaneously. Meanwhile, it also presents users positive or negative interests.

Users may have multiple interest categories at the same time and each interest category represents one domain of interest. According to changes in user interests, the number of categories can grow or shrink. Positive or negative feedback will change the interest level of a category and its corresponding keyword weights. The basic structure of the user profile is shown in Figure 4.1.

From Figure 4.1, we can see that the basic structure of a user profile is represented as a feature vector, P. It consists of a list of categories.

$$P = Category_1, Category_2, \ldots, Category_m$$

An interest category is represented with a feature vector. The category C is composed of three descriptors: a positive, negative, and long-term descriptor. The positive and negative descriptors maintain short-term feature vectors learned from articles with positive and negative feedback, while the long-term descriptor maintains

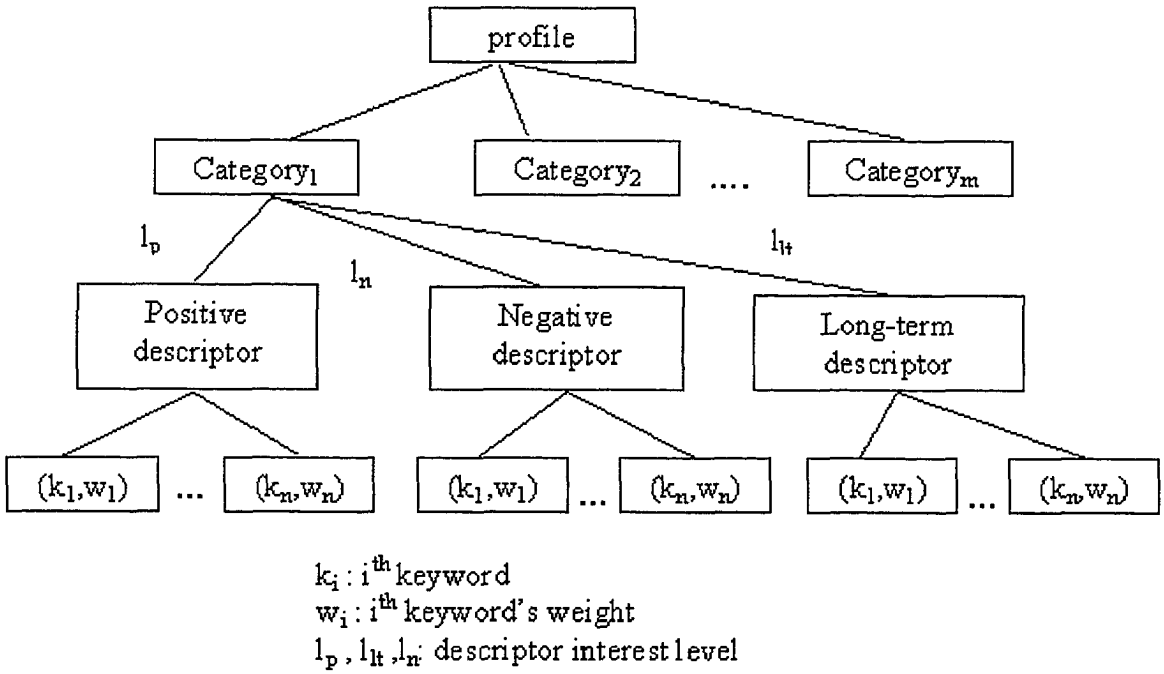


Figure 4.1: Structure of user profile

the long-term feature vector obtained from the history record of the user profile for both types of feedback. Formally, the representation of an interest category can be written as follows:

$$Category =< (l_p, d_p), (l_n, d_n), (l_{lt}, d_{lt}) >$$

where d_p , d_n and d_{lt} represent the positive, negative and long-term descriptors. Each descriptor could be represented as a vector consisting of a list of weighted keywords, $(k_1, w_1), (k_2, w_2), \ldots, (k_n, w_n)$. Interest weights l_p, l_n, and l_{lt} are used to describe the interest level of the positive, negative and long-term descriptors of the interest category. The value of l_p, between zero and one, represents the degree of interestedness and the value of l_n, again between zero and one, represents the degree of uninterestedness. The value of l_{lt} is between -1 and 1. This long-term descriptor

indicates lack of interest for negative values of l_{lt}, and shows interest for positive values of l_{lt}.

Positive Descriptor $l_p = 0.5885$	Negative Descriptor $l_n = 0.0532$	Long-term Descriptor $l_{lt} = 0.1964$
Surveyed 0.0162	Flowers 0.0779	surveyed 0.0162
angle 0.0193	music 0.0802	angle 0.0193
keyset 0.0169	usb 0.0779	keyset 0.0269
robots 0.0165	genealogy 0.0779	robots 0.0165
hell 0.0170	plants 0.0657	simmons 0.0176
doom 0.0677	bird 0.0779	hell 0.0173
roomba 0.033	school 0.0436	network 0.0577
digital 0.0167	mug 0.0779	digital 0.0467
den 0.0203	information 0.0436	disk 0.0365
zboard 0.0226	ibm 0.0779	eros 0.0153
...	...	...

Table 4.1: Example of the content of interest category

Table 4.1 portrays an example of the content of interest category "technology" in the profile.

4.2 Database design

Based on the profile structure described in the previous section, the profile table is designed to store user profile information. The schema diagram is shown in Figure 4.2. The table users is made up of basic user register information such as name and password. The attribute id is the primary key for this entity. The attribute readSpeed indicates the length of time the user should take to read a line of an article. The attribute learningRate is the coefficient of the learning rate used during the learning process. The attributes positiveW, negativeW, and longtermW correspond to the interest levels of descriptors l_p, l_n, and l_{lt}. Table 4.2 shows all the database tables

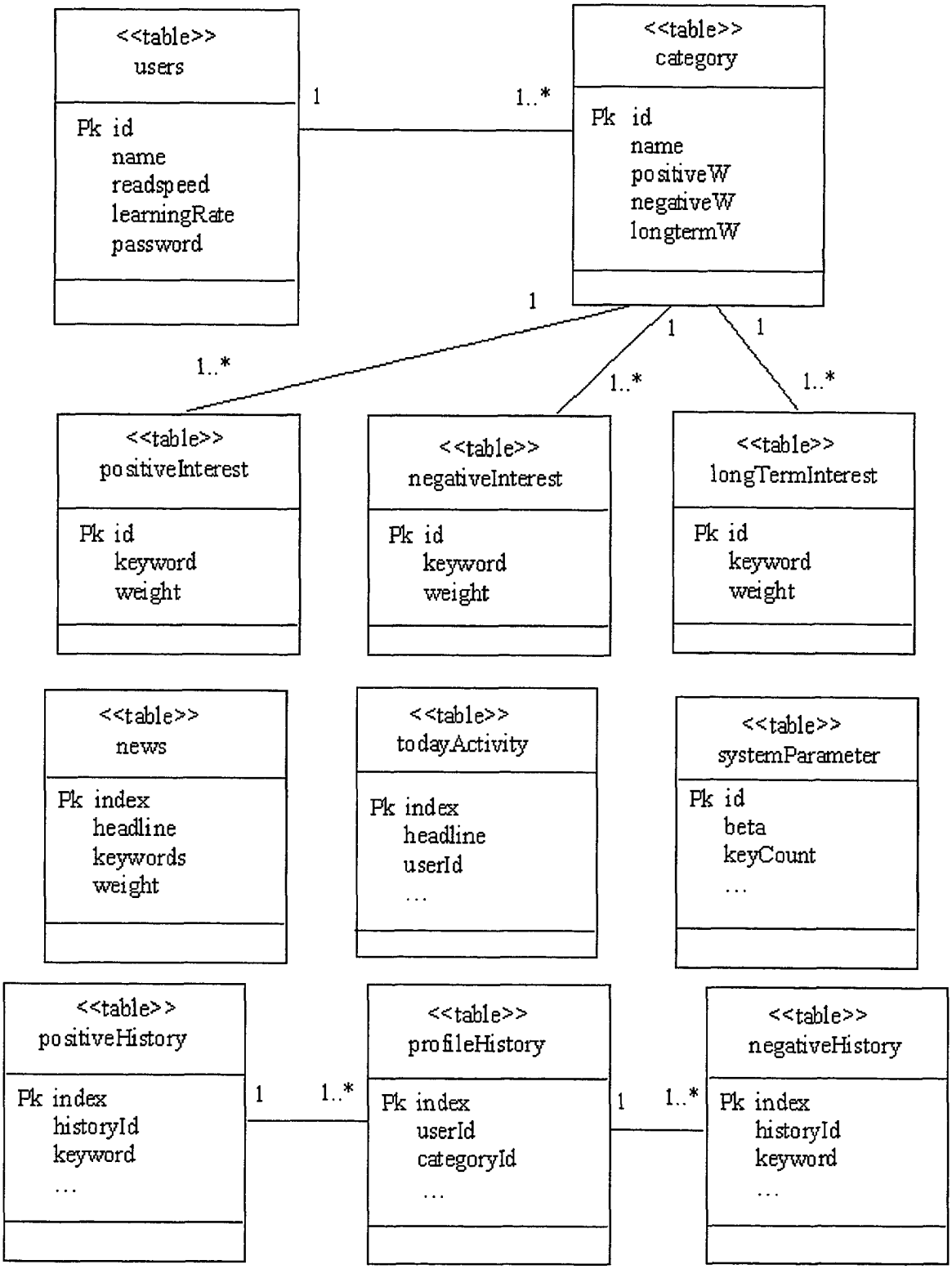


Figure 4.2: Database design diagram

Table Name	Comment
users	Store user's information, such as user name, password
category	Store user's interest category
positiveInterest	Store the positive descriptor information of the interest category
negativeInterest	Store the negative descriptor information of the interest category
longTermInterest	Store the long-term descriptor information of the interest category
profileHistory	Store the change history of the interest category
positiveHistory	Store the history record in the positive interest table
negativeHistory	Store the history record in the negative interest table
News	Store the keywords of news articles
todayActivity	Store the activities of user in a particular time period, such as the headlines of articles the user read
SystemParameter	Store the parameters used to initialize system

Table 4.2: Tables used in the system

used in the system. The tables user, category, positiveInterest, negativeInterest, and longTermInterest are used to store the current user profile. The tables todayActivity, profileHistory, positiveHistory, negativeHistory, and longtermHistory are designed to store all history records. In order to find the kinds of articles the user dislikes, the table todayActivity stores the names of all the articles the user has read.

Some additional tables are also used in the system. The table news is used to store the attributes for each news item, including the title and keywords. The table systemParameter stores the parameters the system is using. Parameter details are

presented in the next chapter.

4.3 Learning user profile

Based on the structure of the user profile described in the previous section, the method of learning the user profile and adjusting the learning rate is described in this section.

4.3.1 Explicit and implicit feedback

In this system, we use both explicit feedback and implicit feedback to modify the profile. At first the system tries to get the explicit feedback. The explicit feedback can be obtained by asking the user to rate the article after reading. Users score the article using five ranks. If the user provides explicit feedback, the system will modify the profile based his feedback. However, if the user does not rating any articles, the system will not modify the profile. In this situation the agent can not learn the interests of the user. We need get implicit feedback by observing the behavior of the user. If the user selects and spends a reasonable amount of time reading the articles on the list, it may imply interest in the article. Users tend to spend more time reading interesting articles than uninteresting ones[13]. Based on the time of spending on reading, five ranks are used to weight the implicit feedback in the system. When a user registers to the agent, the system will test his reading speed and save to the database. Once a user clicks on an article, the system computes the length of the article and records the start and finish times of the reading period. The system also calculates the estimated time for reading the article. The estimated reading time

is the product of the user's reading speed stored in the database and the length of the article. By comparing the estimated reading time with the actual reading time, the system generates an implicit rating for the article. For example, if the actual reading time is approximately equal to the estimated reading time, the system learns that the user read the article at normal reading speed; if the actual reading time is much longer than the estimated time given a reasonable range of error, the system assumes the user read the article carefully. If the reader leaves the terminal to take a phone call or to read newly arrived e-mail messages, then the estimated time will be quite different with the actual reading time. The system will not learn the implicit feedback to update the profile since it is not accurate. The ranks for explicit and implicit feedback are described in the next chapter.

Unlike explicit feedback and implicit positive feedback, implicit negative feedback can not be obtained while the user is using the system. If a user reads many articles in one category, we cannot assume that the user might not be interested in articles that have simply not been read yet. Instead, the system generates implicit negative feedback by a scheduler using the steps described below.

- Find all users who logged on to the system and read articles during the specified day and create a latest active user list.
- For each user in the active user list, the system generates a list of unread news articles.
- The system gives an implicit negative rating for articles in the unread list. Based on this rating and the features of the article, the system modifies the profile.

The process of learning short-term and long-term interest is introduced below.

4.3.2 Learning short-term interest

Based on the profile structure, the learning process developed in the news agent relies on feedback provided by the user. Using the feedback information, the system changes the profile of the corresponding user. The feedback information consists of a document to be learned, the relevant category C in P based on the document, and the learning rate. The document is represented as a feature vector f_d which is a list of weighted keywords, where $f_d = (k_{d1}, w_{d1}), (k_{d2}, w_{d2}), \ldots, (k_{dn}, w_{dn})$. The value of the learning rate, α, indicates whether the user likes or dislikes the article content. The learning algorithm is defined as follows.

```
LearnUserFeedback(f_d, C, r) :
Input:feature vector ofdocument f_d, interest category C of the profile,
      feedback rating r
Output:  new profile
compute the learning rate α based on the feedback rating
update l_p and l_n, the weight of interest levels of positive and
      negative descriptors
if (α>0)
      update the interest keywords weight in the positive descriptor
else
      update the interest keywords weight in the negative descriptor
```

Given the feature vector f_d, feedback rating, and the corresponding interest category C in the profile, the modification of C is performed by the above function. The learning rate can be computed based on the feedback rating. Each feedback rating is

corresponding to a learning rate stored in the database. If it is a negative feedback, we set the learning rate negative. As described earlier, a profile in this system is composed of a set of interest categories, with three descriptors in each interest category (positive, negative and long-term) representing the area of interest. Since these descriptors represent the same group of interest category, there is a trade-off between the interest level l_p and l_n. Increasing the value of l_p may reduce the value of l_n, and vice versa. The details of this process are explained below.

A. Depending on the value of the learning rate, the interest levels of the positive and negative descriptors are modified. If the learning rate is positive, the interest level of the positive descriptor is increased using Equation 4.1. After computing the similarity of the learned document with the negative descriptor, Equation 4.2 is used to decrease the weight of the negative descriptor.

$$l_p(new) = l_p(old) + (1 - l_p(old)) * \alpha \quad (4.1)$$

$$l_n(new) = l_n(old) + (1 - \alpha * Sim(d_n, f_d)) \quad (4.2)$$

If the learning rate is negative, the interest level of the negative descriptor is increased and the interest level of the positive descriptor is decreased using Equations 4.3 and 4.4.

$$l_n(new) = l_n(old) + (1 - l_n(old)) * \alpha \quad (4.3)$$

$$l_p(new) = l_p(old) + (1 - \alpha * Sim(d_p, f_d)) \quad (4.4)$$

B. Depending on the value of the learning rate, the interest keyword weights are updated in the positive or negative descriptors. The keywords in the document contribute to the positive interest keyword in the corresponding category and should be modified if the learning rate is positive. Conversely, if the learning

rate is negative, the interest keywords in the negative descriptor should be modified.

The general equation to modify the keyword and its weight is as follows:

$$w_i^p(new) = \begin{cases} w_i^p(old) * (1-\alpha) + w_j^d * \alpha & if\ k_i^p = k_j^d \\ w_i^p(old) * (1-\alpha) & if\ for\ all\ k_j^d, k_j^d \neq k_i^p \\ w_j^d * \alpha & if\ for\ all\ k_i^p, k_i^p \neq k_j^d \end{cases} \quad (4.5)$$

where

$w_i^p = i^{th}\ weight\ of\ keyword\ in\ either\ d_p, d_n\ or\ d_{lt}$

$w_j^d = j^{th}\ key\ weight\ in\ document\ f_d$

$k_i^p = i^{th}\ keyword\ in\ either\ d_p, d_n\ or\ d_{lt}$

$k_j^d = j^{th}\ keyword\ in\ document\ f_d$

$\alpha :\ learning\ rate$

Three conditions are shown in the above equation. When both the document and the descriptor of the interest category have the same keyword, the keyword weights of both are computed to modify the weight of the keyword in the descriptor. If the descriptor owns a keyword that is not owned by the document, the learning rate (1-α) and the weight of the keyword from the descriptor are incorporated to modify the keyword weight in the profile. If the keyword belongs to the document but not to the descriptor, the keyword weight in the profile is modified by the learning rate α and the weight of the keyword in the document. The range of α is between zero and one. After using Equation 4.5 to calculate the new keyword weights in the category, the system sorts the weight of the keyword and stores the top n weighted keywords to the descriptor of category

in the profile.

The process discussed above is applied to the general situation. If a user reads an article whose feature vector cannot be classified into a category in the user profile, a new category is added to the user profile. The process of creating a new profile category is as follows.

A. Either set the positive descriptor to the feature vector of the document and the value of l_p to α for positive feedback, or set the negative descriptor to the feature vector of the document and the value of l_n to α for negative feedback as follows:

   $d_p = f_d$ and $l_p = \alpha$ for positive feedback

   $d_n = f_d$ and $l_n = \alpha$ for negative feedback

B. Either set the positive descriptor to an empty vector and the value of l_p to zero for negative feedback, or set the negative descriptor to an empty vector and the value of l_n to zero as follows:

   d_n={} and l_n=0 for positive feedback

   d_p={} and l_p=0 for negative feedback

C. Set the long-term descriptor to an empty vector and l_{lt} to zero as follows:

   d_{lt}={} and l_{lt}=0

The learning rate α in the process is provided by the activities of users. Since the learning rate for strong feedback is high, explicit feedback can result in significant changes to the descriptors and their interest levels. Since the learning rate from

implicit feedback is lower, implicit feedback affects the descriptors very slightly. The value of the learning rate is equal to the product of the actual rate and the coefficient of the learning rate, which is stored in the database. Different users will have different coefficients. The actual rate is the rank of the explicit feedback provided by users or the rank of the implicit feedback generated by the system.

The learning rate α is the factor that influences the predicted score of the article. Selecting a good α will improve the accuracy of the predicted score. In our agent, the learning rate is automatically adjusted. We will discuss it in section 4.3.4.

4.3.3 Learning long-term interest

The long-term descriptors in the interest category of the profile represent the long-term interest of the user. Due to differences in the nature of long-term and short-term interests, changes in the short-term interest descriptor tend to accommodate changes in user interest over a short period of time, while the long-term interest descriptor captures user interests over a long time period. In order to modify the long-term interest descriptor, the system records all change in the user profile. The algorithm for learning long-term interest is defined as follows.

```
LearningLongTermFeedback(P):
Input:user profile P
Output:new user profile P
For each category C
      Find all history records of positive descriptor and
           negative descriptor in C
      Compute the sum of l_p and l_n
```

```
If sum of l_p > sum of l_n
    Set l_lt to the average of the sum of l_p and l_lt(old)
    Find most frequently appeared keywords in the positive
        history descriptor to update long-term descriptor
Else
    Set l_lt to the average of the sum of l_n and l_lt(old)
    Find most frequently appeared keywords in the negative
        history descriptor to update long-term descriptor
```

When the long-term interest learning process is triggered, the system goes through all users in the database and computes the long-term interest for each user. Let's denote the profile history of one user as $P_{history} = (P_1, P_2, \ldots, P_n)$, where P_n, where P_n is the nth day's profile. Based on the profile history, the system computes the sum of positive and negative interest levels. The decision as to which short-term descriptor influences a given long-term descriptor is determined by the sum of interest levels. If the sum of positive interest levels is larger than that of the negative interest levels, it means that the positive descriptor will influence the long-term descriptor more than the negative descriptor, and vice versa. After determining whether to use the positive or negative interest descriptor to update the long-term descriptor, the system then finds the keywords based on the frequency of keywords in $P_{history}$.

4.3.4 Adjust learning rate coefficient

The recommendation module provides suggestions to users about news they may want to read. In the real world, reader interests can be very dynamic. Therefore,

the system needs to consider whether the user is satisfied with the news. There are two ways to learn this. First, the system asks the user to provide feedback. Second, the recommendation mechanism is adjusted by comparing the actual rating and the predicted score for the recommended article. Since the first method requires significant user effort, we use the second method in our system. The learning rate is the factor that influences the predicted score of the article. In order to improve the accuracy of the predicted score, the system uses a Least Mean Squares (LMS) or Temporal Difference learning algorithm to adapt the learning rate a parameter[16, 24]. When a new user register to the system, the agent sets a learning rate to him, such as 0.5. Each user has own particular learning rate since the activities of different users are different. Let's denote a coefficient of learning rate,β, which is used to adjust the learning rate. β is stored as a system parameter in the database. Based on the actual rating value and the predicted score, the new learning rate can be calculated by the equation 4.6.

$$\alpha \text{ (new)} = \alpha(\text{old}) + \beta(\text{AcutalRating-PredictedScore}) \qquad (4.6)$$

PredictedScore can be obtained by computing the similarity between the profile and document, while ActualRating is obtained from the user.

If the user is not satisfied with the news, a large difference between the PredictedScore and ActualRating exists. Under this condition, the accuracy of predicted score is low and the learning rate is no longer eligible. The system should select a new learning rate that is quite different from the old one. From equation 4.6, we see that the difference between the new α and old α will be large when there is a large discrepancy between the PredictedScore and ActualRating. If the user is satisfied

with the news, the learning rate is still suitable. Then the new learning rate close to the old one is selected based on the equation 4.6.

4.4 Information filtering method

As discussed in chapter 2, there are two kinds of information filtering techniques, content-based filtering and collaborative filtering. Since we focus on learning user profiles and the collaborative filtering has the "Early rater problem", we choose content-based filtering in our thesis. When a new user registers to the system, his profile is empty. The agent can not know his interests. Using content-based filtering technique, the agent can get the user's interests from the articles that the user has read.

Based on the profile structure described in the previous section, the process of finding relevant documents from a set of documents is described below.

For each document in the article set, the level of user interest in the content of a document is assessed by matching the degree of interest in the category of the profile. The assessment is computed as a numeric value assigned to score the document. Based on these scores, documents are ranked and the n most relevant documents are obtained from the n top ranked documents.

The value of the score for a document is between -1 and 1, since an interest category has positive and negative interest descriptors, both interests can indicate the opposite meaning, and the score of long-term interest contributes to either positive or negative interest depending on the sign of its value. For a document, a positive score means that the user may be interested in the document. Conversely, a negative score indicates a lack of interest. Based on a profile P, the score of a document denoted as a feature vector, f_d, is computed as follows.

A. Find category C and its corresponding descriptors in P based on the category of the document.

B. Calculate the score of each descriptor in interest category C with the greatest relevance:

$$w_{pos} = l_p * Sim(d_p, f_d)$$

$$w_{neg} = l_n * Sim(d_n, f_d)$$

$$w_{long} = l_{lt} * Sim(d_{lt}, f_d) \qquad (4.7)$$

$$Sim(d, f_d) = \frac{\sum d \cdot f_d}{\sqrt{d^2}\sqrt{\sum f_d^2}}$$

C. Compute the final score of the document as:

$$Score(P, f_d) = max(w_{long}, w_{pos}) + min(w_{long}, -w_{neg}) \qquad (4.8)$$

After computing scores for all the documents in the set using the above process, the documents are sorted and the n most relevant documents are selected.

After describing the structure of the user profile and the learning algorithm, we are ready to implement the components of the news agent. The detailed implementation of each component is introduced in the following chapters.

Chapter 5

Implementation

As described in the previous chapter, the news agent is made up of five components. This chapter describes the detail implementation of these components.

In our agent, there are three packages that store a group of classes according to their functionalities: "IR", "NewsPackage", and "UserProfile". The IR package includes all classes that are used to download HTML pages. NewsPackage includes classes that are used to implement the news articles, and UserProfile includes the classes used to maintain the user profile and communicate with the database. The detailed classes will be described with their corresponding components.

This chapter is organized as follows. Section 5.1 describes the system parameters in the news agent. Section 5.2 presents the approach for retrieving articles from the web site. Section 5.3 presents the method of recommending articles. Section 5.4 introduces the implementation of the learning module. Section 5.5 introduces the user interface of the personalized news agent, and Section 5.6 presents the process used by the admin module.

5.1 System initialization

When the system is run for the first time, some parameters need to be set. Here is a list of the system parameters:

- Amount of keywords: Defines how many keywords are stored in the interest descriptor of the user profile.
- Max news Amount: Defines the maximum amount of news in the news set when the system calculates the keyword's weight.
- Error for reading speed: Defines the maximum error between the estimated reading time and the actual reading time when the user reads the article in the normal way.
- Basic reading time: Defines the minimum error between the estimated reading time and the real reading time when the user just goes through the article.
- Period of long-term: Defines how long the system updates long-term interests for the user (for example, input of two means two days).
- Beta: the coefficient of the learning rate.
- Explicit and implicit learning rates: Define the value of explicit and implicit feedback learning rates. Learning rates have five ranks, as shown below.

  The explicit feedback ranks are:

  - "Always show articles like this" for strong positive feedback.
  - "Interesting" for moderate positive feedback.

- "Not bad" for weak positive feedback.
- "Not interesting" for moderate negative feedback.
- "Never show articles like this" for strong negative feedback.

The system generates implicit feedback by observing user behaviour, which is classified into the five following ranked categories:

- "Click the article": user clicked the article but spent little time reading it.
- "Read carefully": user read the article carefully, taking more time. The system assumes that the user is very interested in the article and gives strong positive implicit feedback.
- "Read in normal speed": user read the article using his normal reading speed. The system assumes that the user is interested in the article and gives moderate positive implicit feedback.
- "Maybe just go through the article": user spends less time reading the article than estimated time needed. The system gives weak positive implicit feedback.
- "Do not read": user does not read the article. The system gives negative implicit feedback.

5.2 News retriever

The task of the news retriever is to import the latest news from the web site at www.canada.com. The activity of the news retriever is run by a scheduler, which

determines when the news retriever begins collecting articles. There are three steps to this task:

- Download articles from web site
- Analyze articles and save to database
- Computer the keywords and their weights for each article

The news retriever first obtains the content of the web site and stores its web pages into a temporary folder. The articles stored at this web site are organized based on the news category. In this website, articles are classified into several categories, including national, world, business, sports, entertainment, health, and technology. Every article in this web page is organized similar to the following example:

http://www.canada.com/news/national/story.html?id=81134a87-77b1-4ab7-b5b4-59d4236c3f3a

After splitting it, we discover that the category of this story is "national."

The second step is to analyze the content of the article. The HTML page for each story in this web page also has some basic rules. The page contains some flags denoting the meaning of the content, such as:

<DIV CLASS="storyheadline">

<DIV CLASS="storydate">

<DIV CLASS="storytext">

<FONT CLASS="storybyline">

<FONT CLASS="storypub">

Based on these flags, we check whether the document contains news articles. If the document does not contain news articles, it is removed. If the document does contain a news article, the process of finding the useful information from the document begins. During this process, formatting information (such as HTML tags, Java script, and links) is removed. The output of this process contains the title, date, author, category, and content of the article, and this useful information is saved to our database. In the system, we use XML files to save the article information. The file name is given by the time of download. The tags defined for this XML file contain <News>, <Category>, <Headline>, <Byline>, <StoryPub>, <StoryDate>, and <Content>. Here is an example of the news file:

<News><Category>national</Category>

<Headline>Police sniper kills man holding woman at gunpoint in downtown Toronto</Headline>

<Byline>Colin Perkel</Byline>

<StoryPub>Canadian Press</StoryPub>

<StoryDate>Wednesday, August 25, 2004</StoryDate>

<Content> TORONTO (CP) - A police marksman shot and killed a man holding a woman passerby at gunpoint in front of busy Union Station at the height of Wednesday's morning rush hour, leaving the hostage shaken but unharmed and the normally bustling area eerily quiet.

.....

</Content></News>

<News><Category>national</Category> <Headline>Doping allegations continue to take shine off strong and emotional performances</Headline>

<Byline></Byline>

<StoryPub>Canadian Press</StoryPub>

<StoryDate>August 25, 2004</StoryDate>

<Content>

ATHENS (CP) - Another day, another doping allegation in Athens.

.....

</Content></News>

Finally, the retrieval module computes the keywords and their weights for each article using the TFIDF method introduced in Chapter 2. Three steps are developed to compute the keywords.

- Remove common words and stemming words.
- Compute the keywords and weights for an article using Equation 2.2, thereby deriving the feature vector of the document.
- Sort the keywords and find the N highest weight keywords, and save the document title, keywords, and weight into the news table in the database. Parameter N is the amount of keywords stored in the systemParameter table.

We define several classes to complete the above tasks of the retrieval module. Figure 5.1 shows the class diagram of the retrieval module. We extend the downloadNews class from the directorySpider class which is stored in the IR package. It is used to download articles from the specific web sites. The NewsRetriever class is used to set the URL of the specific web site, and call the downloadNews method to download

the news from the web sites and saves to the template folder. The NewsReader class is used to obtain the content of the news articles and compute the article keywords and their weights. It calls the addNews method in the DBProcess class to request that DBProcess add the article features to the database.

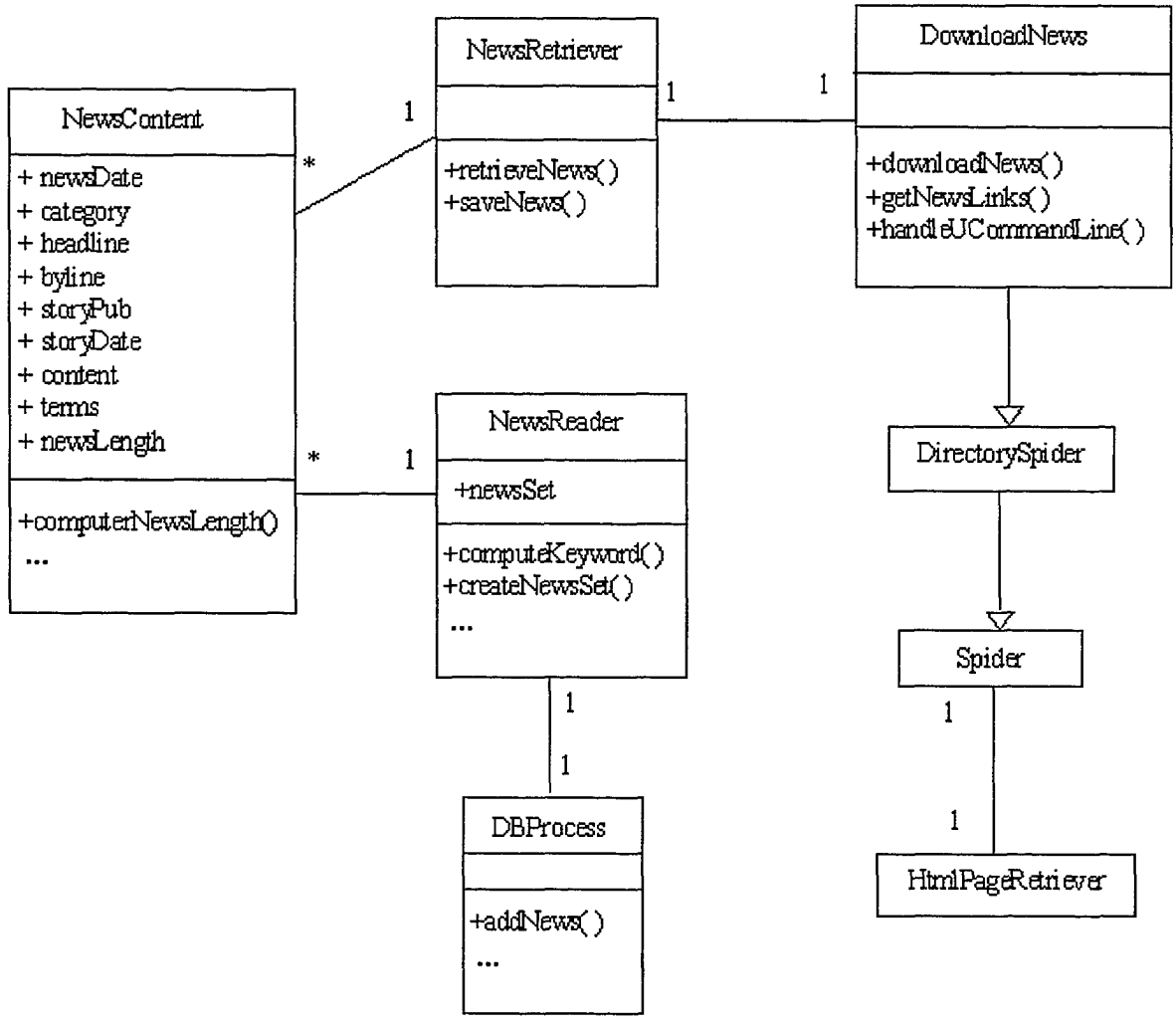


Figure 5.1: Class diagram for retrieval module

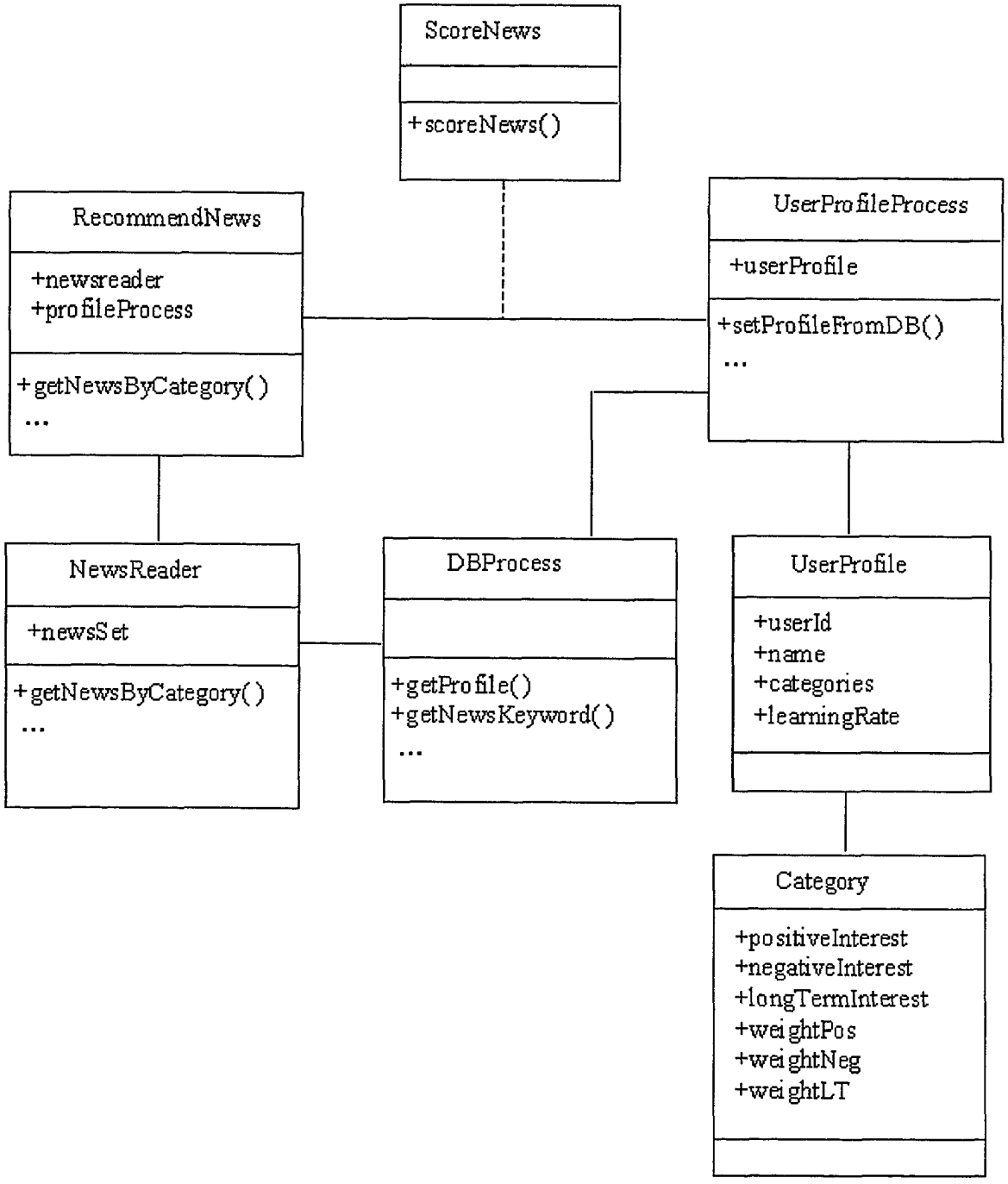


Figure 5.2: Class diagram for recommendation module

5.3 Recommendation module

The task of the recommendation module is to create news web pages for users. This module is triggered when one user sends a request to display the list of the latest articles. Based on the profile structure and information filtering method described in the previous chapter, this module scores and ranks each document according to the user profile. The documents are then ordered and kept based on decreasing document score values. Figure 5.2 shows the class diagram for the recommendation module, and Figure 5.3 illustrates the process of the recommendation module. The recommendNews class is used to create the news article list based on the current user profile and the news articles in the database. It calls a function in the newsReader class to get the list of news articles. The newsReader class obtains the features of the articles (such as keyword and weight) from the database by calling a function from the DBProcess class. The recommendNews class calls a function in the UserProfileProcess class to get the current user profile information. The UserprofileProcess class is used to maintain the user profile. It communicates with the database through the class DBProcess. With the association class ScoredNews, the RecommendNews class can score and sort the news items.

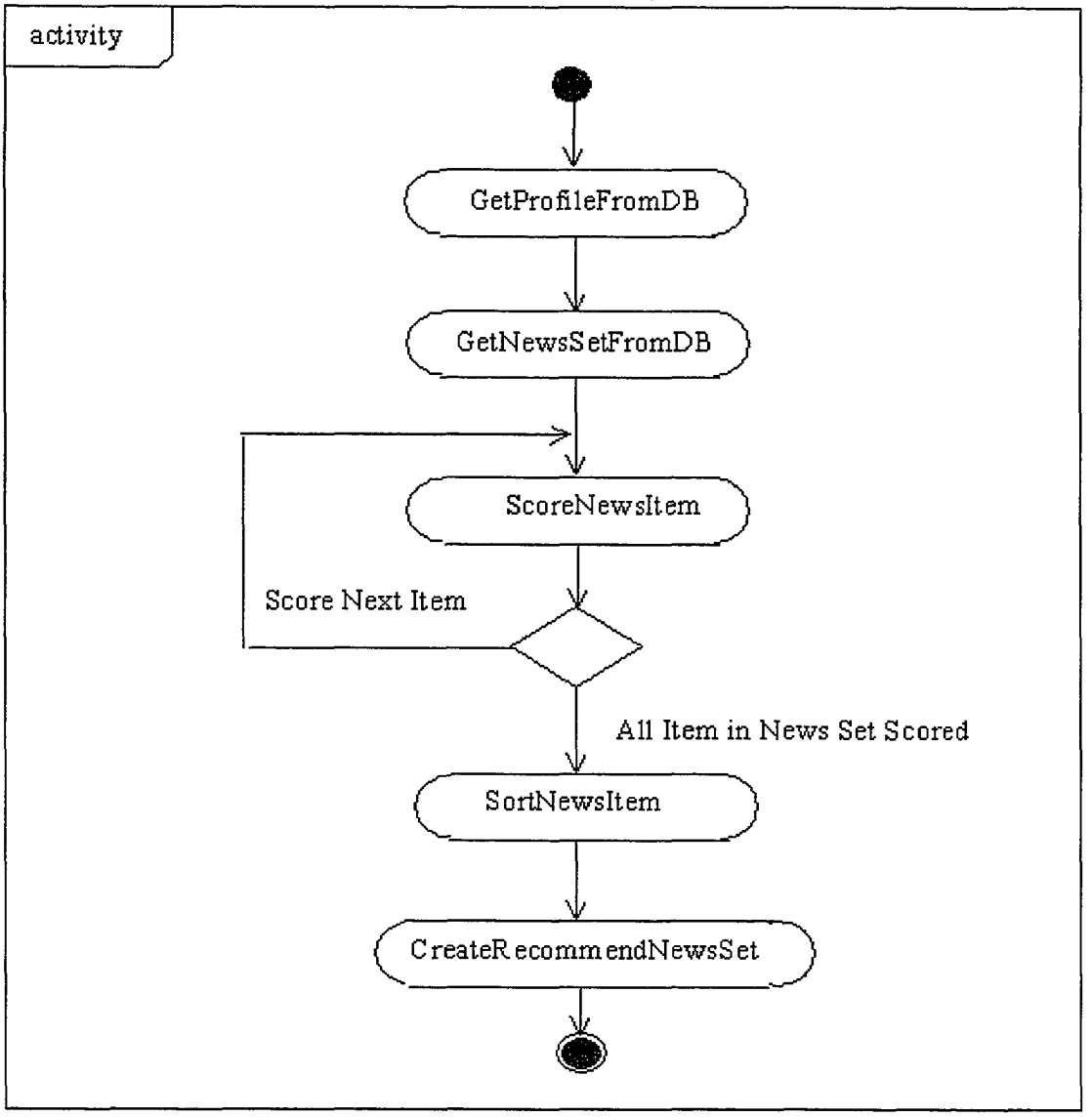


Figure 5.3: Activity diagram for recommendation module

5.4 Learning module

The task of the learning module is to maintain the user profile and adjust learning rate. As mentioned in Chapter 4, the agent learns the user profiles from explicit and implicit feedback. When users read the articles or the system extracts the latest articles, the learning module is triggered. The explicit positive and negative feedback is obtained from the user's ratings and the implicit positive feedback is defined by observing the user's browser when the article page is closed. Implicit negative feedback is triggered by a scheduler, which is performed once a day before extracting new articles. In order to ensure that similar documents in future recommendation processes will be scored high for positive feedback or low for negative feedback, the learning rate should be set to a high value for strong explicit feedback or a low value for week explicit feedback. Since implicit feedback gradually changes the profile, a lowest learning rate value will be set. According to the experiment we will discuss later, the learning rate for strong explicit feedback is set to 0.9, the learning rate for moderate explicit feedback is set to 0.5, and a learning rate of 0.2 is assigned to weak positive feedback or negative feedback. A learning rate value of less than 0.2 is assigned to implicit feedback. With different learning rates, the learning module runs the process of learning short-term interests. Long-term interests are learnt by a scheduler when the system runs the retrieval module to extract the latest news from the web site.

Figure 5.4 shows the class diagram for the learning module. Figure 5.5 illustrates the process used by the learning module. In Figure 5.4, the recommendNews class is used to obtain the current rated news article features and the predicted rating. The userProfileProcess class is used to learn changes in user interest or learning rate.

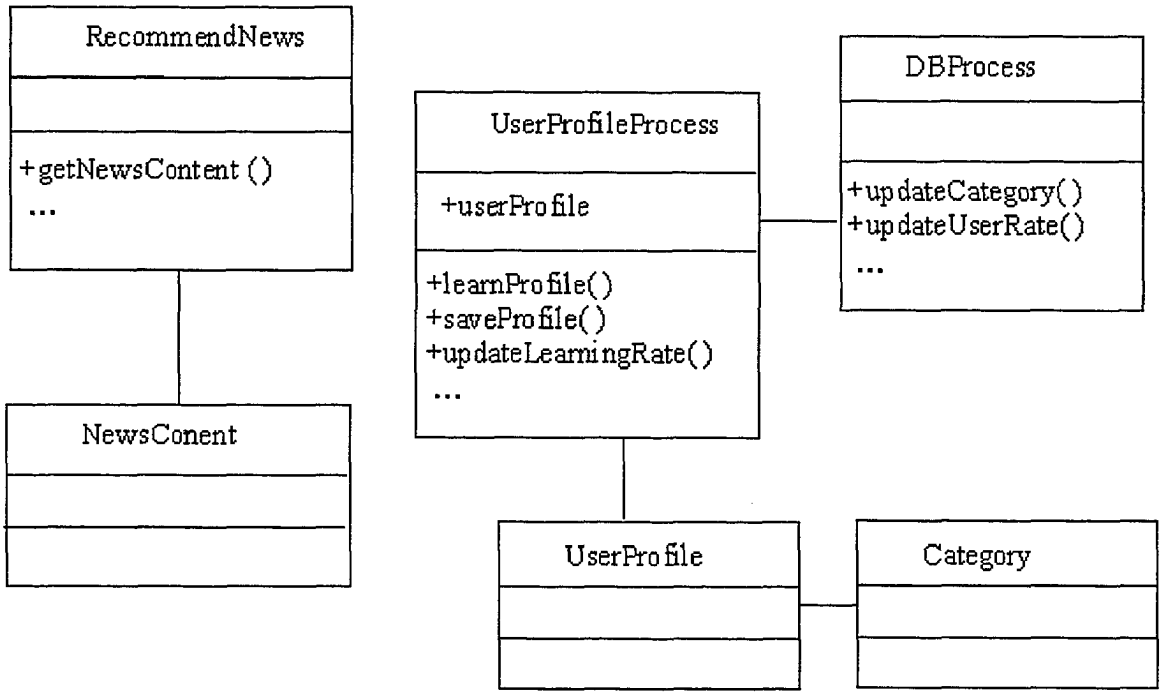


Figure 5.4: Class diagram for learning module

It communicates with the database through the DBProcess class. When a new user registers in the system, the user profile is initialized. In the subscription form, the user is required to select a list of interest categories. For each interest category, the userProfileProcess class calls a function to build the initial profile of the new user after the completed subscription form is submitted.

Additionally, our system separates feedback data into two data sets, the training set and test set. The test set is used to assess the quality of the training results. While 80% of feedback data are put into the training set to update the user profile, the other 20% are used to adjust the learning rate. If the feedback data is in the test set, the agent will compare the actual rating and the predicted rating generated by the system. When users change their short-term interests, the discrepancy between the actual rating and the predicted rating will be large. It means that the learning

rate is no longer suitable for the future prediction. As we describe in the previous chapter, the new learning rate can be generated based on the equation 4.6. When the discrepancy between the actual rating and the predicted rating increase, the difference between the new learning rate and the old one increase.

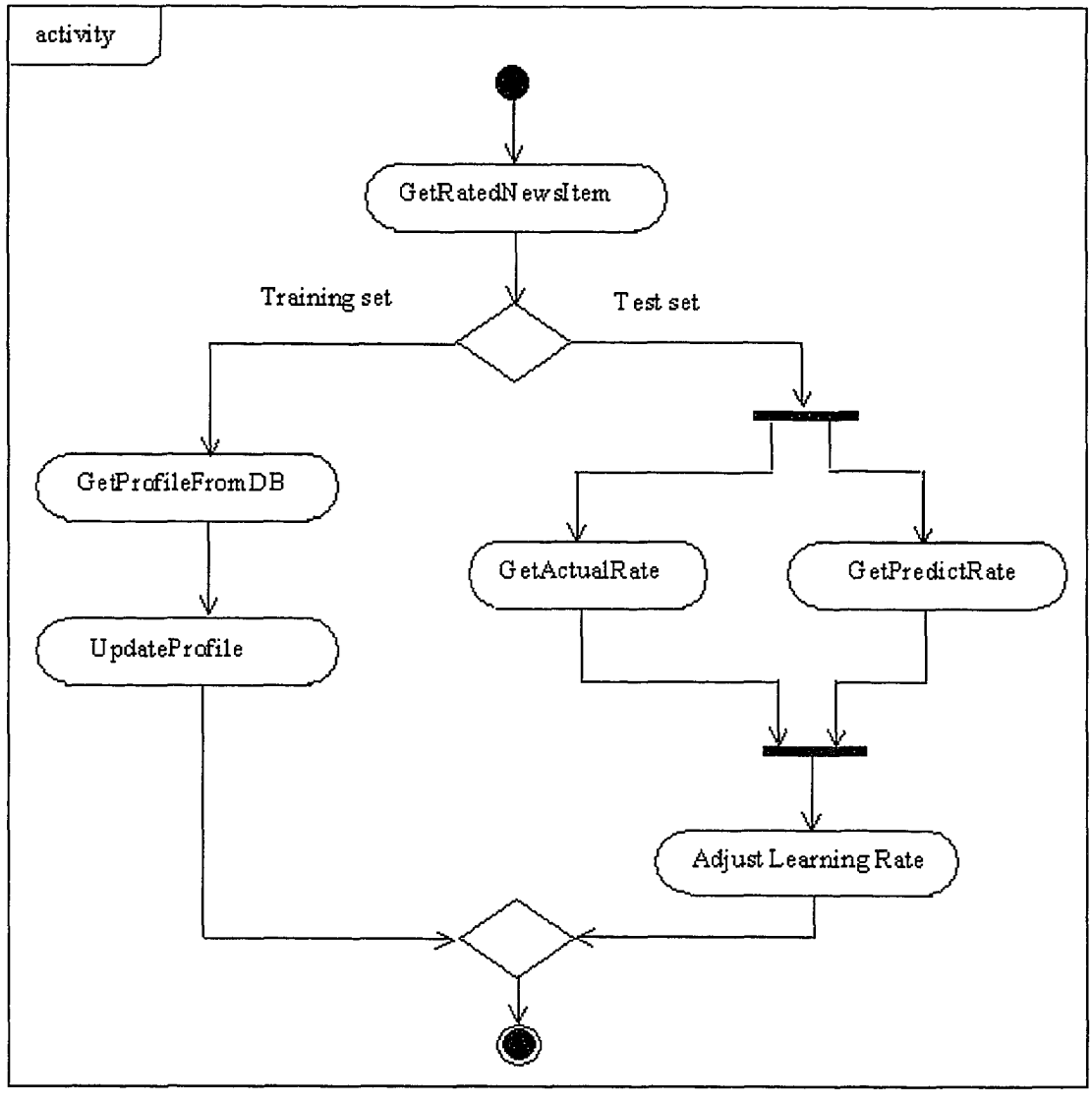


Figure 5.5: Activity diagram for learning module

If the feedback is in the training set, the userProfileProcess class calls the function

that learns the changes in user interests. Otherwise, the function that learns the changes in the user learning rate is called, and the recommendNews class calls the function to get the predicted ratings for the articles.

5.5 User interface

The user interface of our personalized news agent manages the interaction between the user and the system.

On the client site, the user interface is a set of HTML documents running on the user's browser. The interface provides a means for users to register, sign on to the system, browse the titles of recommended news, select and read news stories, and give explicit feedback to system. On the client site, the user interface also tracks the behavior of users in order to collect implicit feedback.

On the server site, the user interface consists of a set of JSP documents. It creates an HTML interface and sends information to the user through an Internet browser such as Internet Explorer or Netscape. The interface parses the information sent by the user and processes it according to user requests, such as processing a subscription form and user feedback, validating user identification and passwords, opening a personalized news list page, or opening a news article page.

5.5.1 New user Registration

Users must register with the system the first time they use the news agent. A new user can subscribe and obtain an account for using the personalized news agent by clicking the register link from the home page of the agent. When the user clicks to

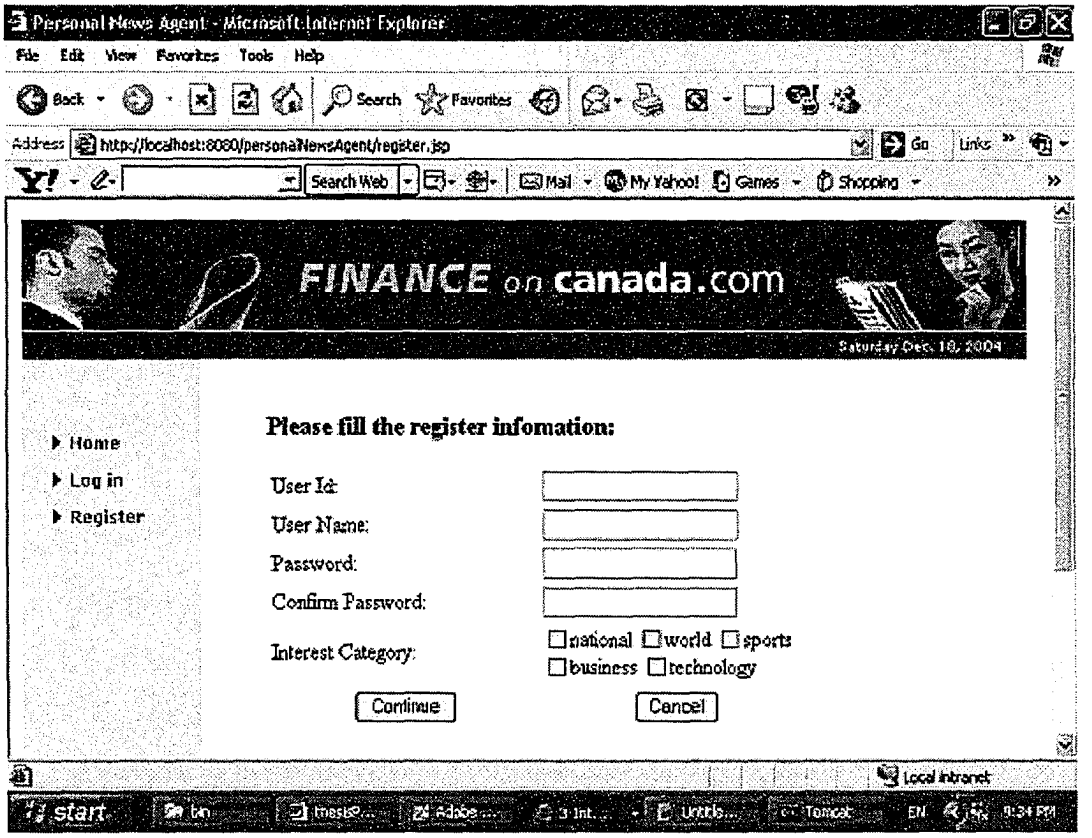


Figure 5.6: Register window

register on the register page, a subscription form is presented to the user to complete. The user is required to provide information such as id, name, password, and interest category. The identification and password must be supplied, while identifying interest categories is optional. The user then clicks the continue button to go to the next page. The second page of registration asks the user to read one paragraph to test reading speed. When the user clicks the submit button on the second page, all information is sent to the server. The corresponding JSP file will parse the information and check the validity of the user's identification and password. If the user's identification is unique, all user information is stored in the database. The learning module is then invoked to create a new user profile. Figure 5.6 shows the first page of registration.

After completing registration processing, the recommendation module is invoked

to create a personalized news list page.

5.5.2 Display personalized news page

After a user provides valid user identification and password, a personalized page containing recommended news titles grouped by category can be opened. Validation is performed by checking the log-on information in the database. Figure 5.7 shows a personalized news page. The article titles are ordered by decreasing matches with the user profile. The title of the news articles are used as links to open the corresponding articles. When the user selects an article, the system pops up a window displaying the content of the article. After finishing reading the article, the user can close the window or provide feedback to return to the previous window.

5.5.3 Handling user feedback

When the user opens a news article by clicking on the title, a request to display the article is sent to the server. The server finds the corresponding article content and creates a page that contains a feedback control and an area used for displaying the news article. Users can scroll up and down the news article without affecting the appearance of the feedback control.

The feedback control contains five different ratings for user feedback ranging from strong positive feedback to strong negative feedback. Clicking the comment button will invoke the server to process the user's feedback. Based on the rating and the content of article, the learning module is then triggered for further processing and the current article page is closed. At this point, the administrator can see the effect of the user's feedback by checking the user's profile.

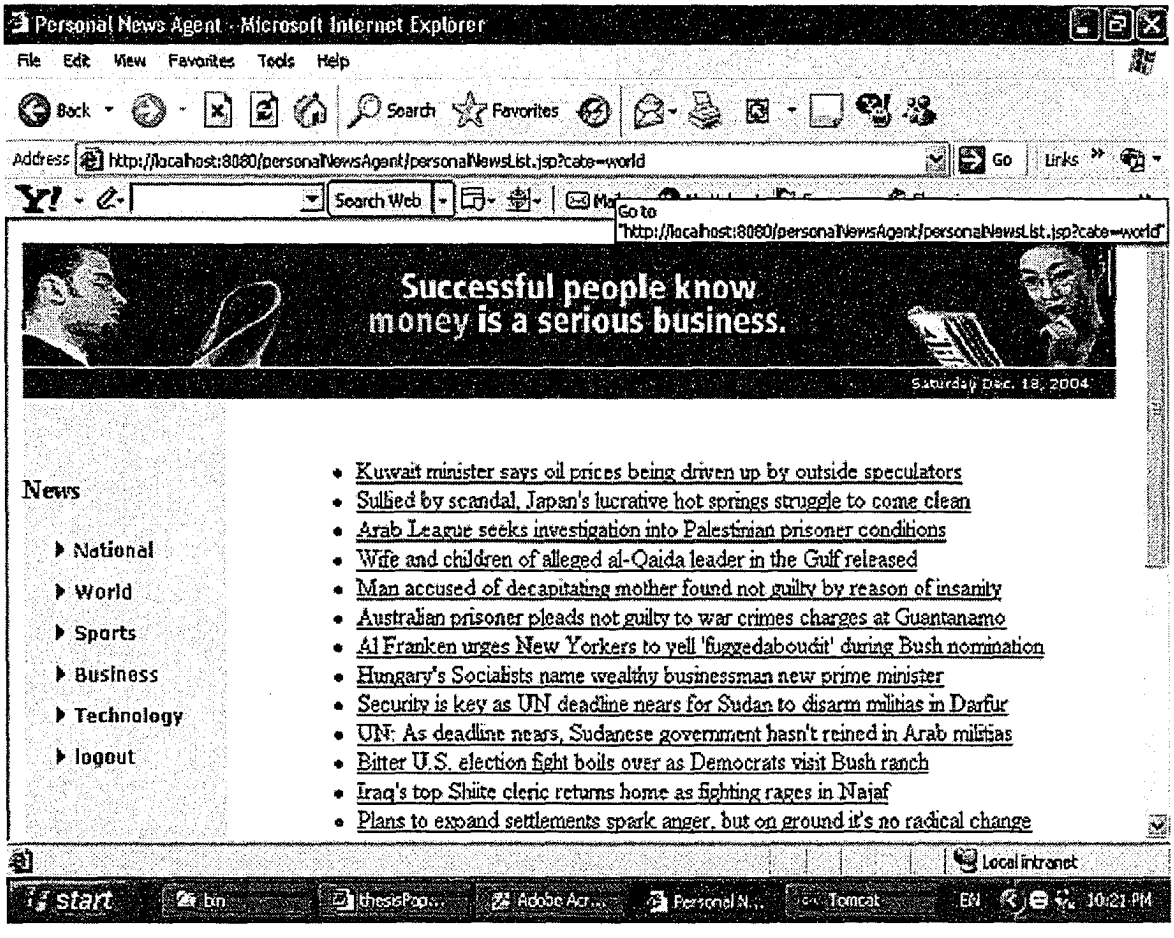


Figure 5.7: Display recommended news

5.6 Admin module

The admin module is a tool to configure the system parameters and display all user profile information.

On the client site, the admin interface is a set of HTML documents running on a browser. The interface provides a means for administrators to do configuration, check the log of users and their current profiles, and retrieve articles from the online website.

On the server site, the admin module parses the requests sent by the administrator

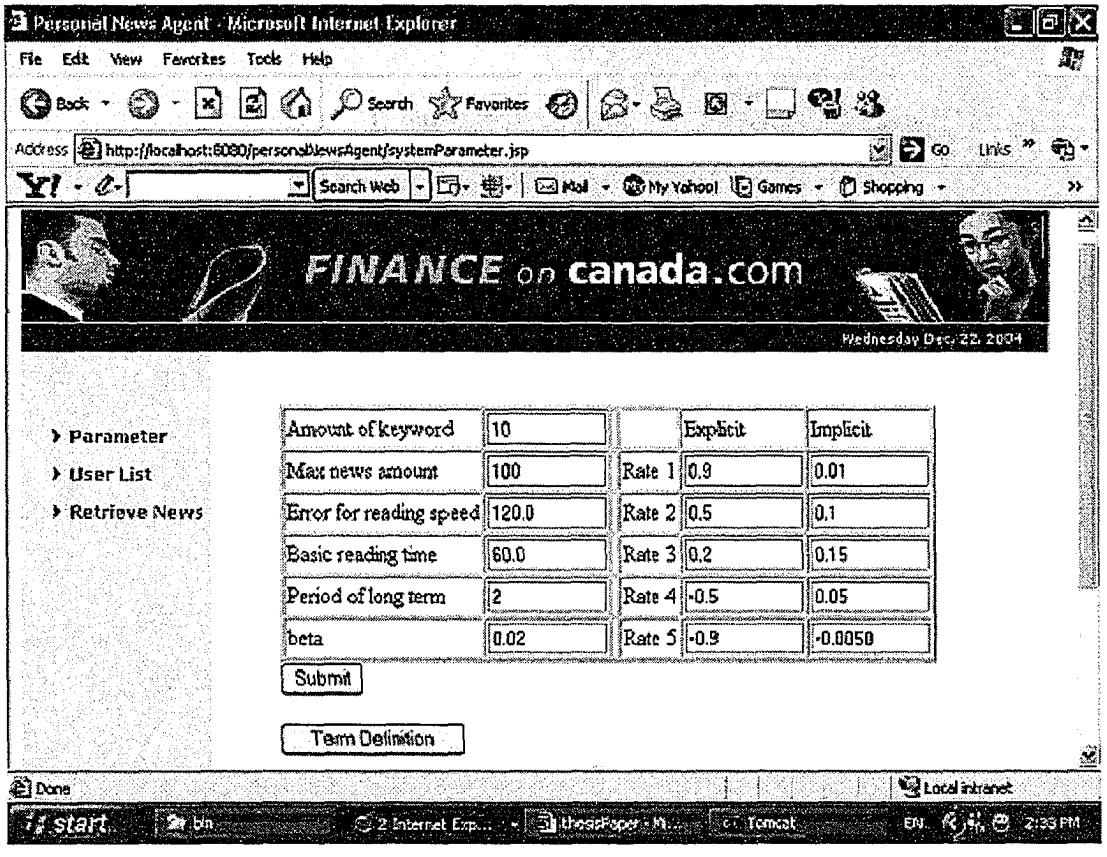


Figure 5.8: Configure system parameters

and processes them. This processing could include saving new values of the system parameters, finding user interest categories or the information in the interest category, or retrieving online news articles.

5.6.1 Configure parameters

When the personalized news agent runs for the first time, the system configuration function should be run to initialize all parameters, such as the learning rate and values of ratings for different ranks. The details of these parameters were introduced in previous sections. When the administrator clicks the link to set parameters in the

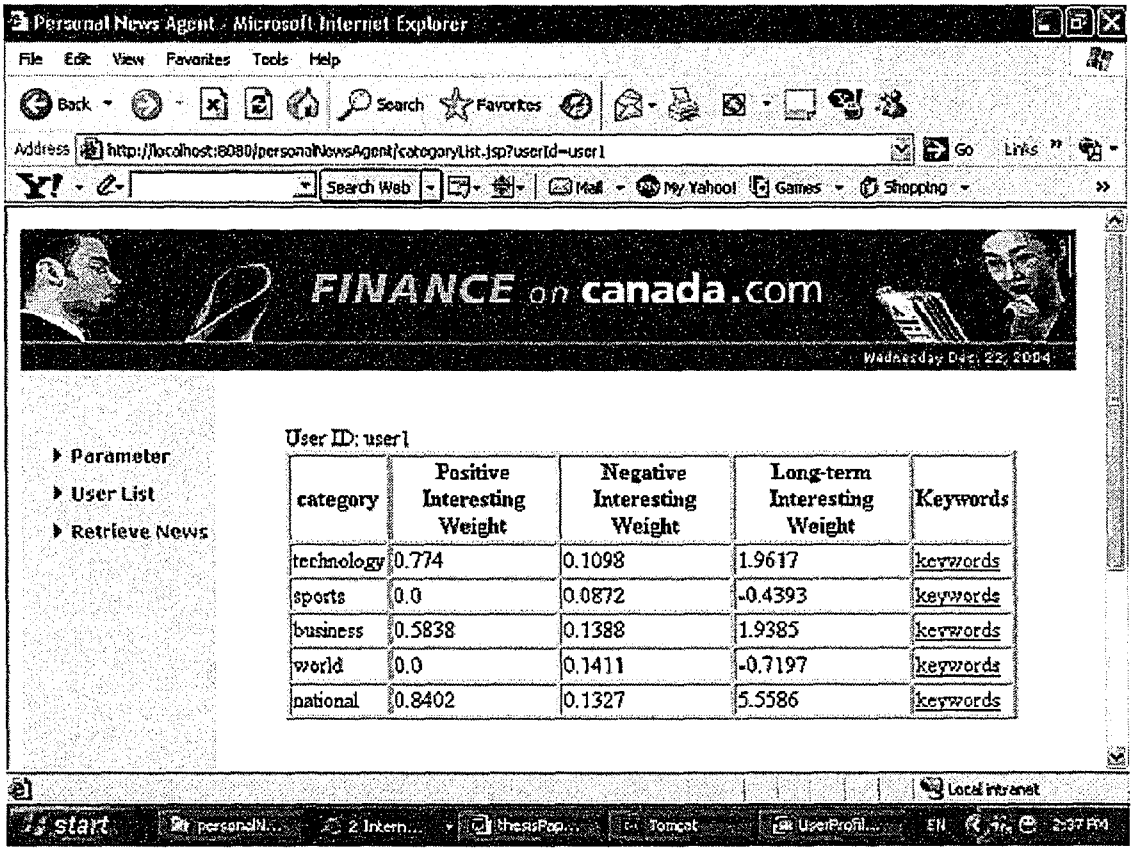
home page of the admin module, a form containing current parameter information is displayed. The administrator can modify the values of parameters, and then click the submit button to request the server to save modifications to the database. Figure 5.8 shows the configuration window. The term definition button is used to open the help window to explain the meaning of the parameters.

5.6.2 Check user profile

The user profile in the system is changed when a new user registers or provides feedback. The admin module is able to monitor these changes while the user is using the system. The administrator can click the user list link in the home page of the admin module open a page displaying all users. The server parses the request to get all user information, such as user name and read speed, from the database and creates a dynamic HTML page. This web page has a table containing user information and two links (log and category).

If the administrator wants to look up the profile history, the user log can be clicked. An example of a log is presented in Appendix A. The log records all changes to the user profile and learning rate. If the administrator wants to examine the current profile for a particular user, the corresponding category link of that user can be clicked to open the page displaying the list of interest categories. Based on the user identification, the server searches the interest category information from the database. Figure 5.9 is an example to show the category information for "user1". The link "keywords" in the page is used to display detailed information for the corresponding category. When the administrator requests the keywords of the interest category, the server searches the database based on the user identification and category name to find the positive,

negative, and long-term keywords and weights.



category	Positive Interesting Weight	Negative Interesting Weight	Long-term Interesting Weight	Keywords
technology	0.774	0.1098	1.9617	keywords
sports	0.0	0.0872	-0.4393	keywords
business	0.5838	0.1388	1.9385	keywords
world	0.0	0.1411	-0.7197	keywords
national	0.8402	0.1327	5.5586	keywords

Figure 5.9: Example of category information

5.6.3 Retrieve articles

Another function of the admin module is to retrieve articles from online web sites. The administrator can click a link to retrieve news in the home page of the admin module and download the articles from the Canada.com website. The server then invokes the retrieval module to download the latest articles from the online website and analyze those articles. Meanwhile, the negative implicit feedback and long-term feedback mechanisms are also invoked to modify the user profile.

Chapter 6

Experiment

A personalized news agent must be able to learn and represent current user interests and adapt itself to changes in user interests over time. In this chapter, we present the tests of our system.

6.1 Collect information on news source

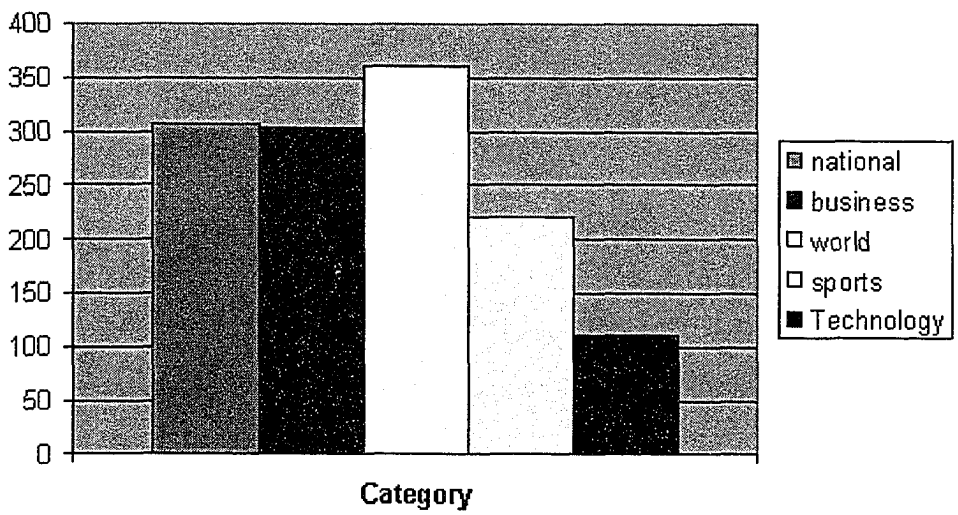


Figure 6.1: Articles in the database for the test

We first collected and observed news items provided by the Canada.com between August 25 and September 30. These news items are categorized into five categories (technology, sports, business, world, and national). Article numbers for each category in the database are shown in Figure 6.1. There are a total of 1,302 articles in five categories. During the experiments, no new articles were inserted or deleted for the simulated users.

6.2 Ranks of rating test

In the system, we use five ranks to represent explicit and implicit feedback. Experimenting with the sensitivity of the ranks of rating is done to discover the extent to which the position of a document is affected after the system has learned from the feedback.

According to the procedure provided by Widyantoro[34], the steps of the experiment to determine the ranks of rating based on the articles in the database are as follows.

1. Create a document set D from database.
2. Generate a profile using random feedback with a random document from D.
3. Compute the similarity between the document and profile for all documents in D based on the profile generated in step 2.
4. Sort the documents in decreasing order based on the similarity score.
5. Select document d in D and use it for feedback with a rating value to be observed.

6. Learn the feedback.

7. Compute the similarity again for all documents and record the new position of document d with the modified profile.

8. Check if the document position has changed.

9. Repeat the above steps using other different ratings from 0 to 1.

The experimental procedure above can be used to observe both positive and negative feedback ratings. If the procedure is used to observe positive ratings, the last document with the lowest similarity score is selected and a positive feedback is given in step 5. If the procedure is used to observe a negative rating, the first document with the highest similarity score is selected.

In this experiment, artificial users are used instead of human users to:

- Ensure that the documents are consistently rated.

- Cut down the evaluation time, as it could take several months to test the system with human users.

We begin by creating ten different document sets, and experiment on each set using the above procedure. We count the number of documents that change positions for a particular rating. The result is shown in Figure 6.2. From this figure, we find that when the rating increases, the number of documents whose position is changed after learning the feedback increases. This means when the rating value is high, the new predicted score for the document has high chance to be different from the old one. According to the expected effects for feedback, the results of our test give an idea of choosing the value of learning rate. In order to get a significant effect, ratings

above 0.8 are designated as strong feedback. Ratings between 0.4 and 0.5 are set as moderate feedback, and weak feedback uses a rating of 0.2. Since implicit feedback affects user profiles weakly, ratings for implicit feedback use values below 0.2.

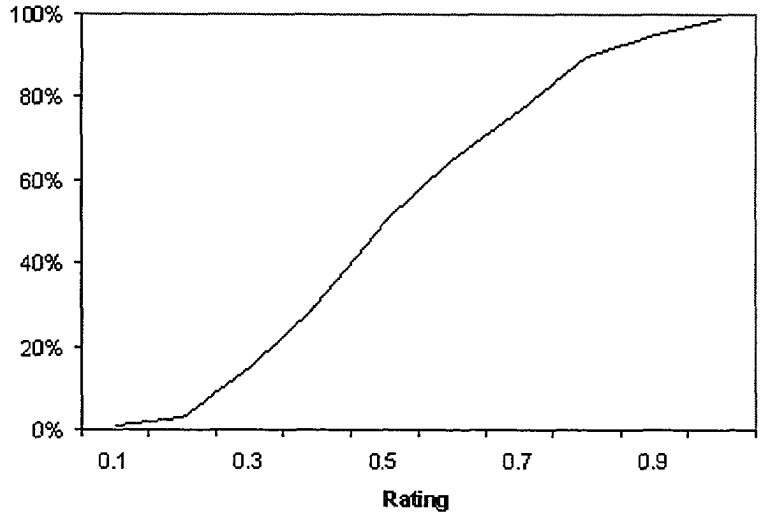


Figure 6.2: Effect of rating on changing position of document

6.3 Adapting experiment

A personalized news agent repeatedly interacts with users to track the changes in user interests over time. It can adapt to change in user interests by maintaining user profiles. The agent recommends relevant articles to users based on the user profile. This experiment measures the adaptive component through the distance between the user's ranking of a set of documents and the system's ranking of the same documents. This method of measurement is called the NDPM measure (Normalized Distance-based Performance Measure), introduced by Yao[37].

6.3.1 Normalized Distance-based Performance Measure

Precision and recall measures are generally well known and commonly used to evaluate the performance of standard IR systems. Precision is defined as the proportion of relevant articles to all retrieved articles and recall is defined as the proportion of relevant articles retrieved to all relevant articles. Since users have difficulty in making consistent relevance judgments over a long period of time when they are asked to rate documents on an absolute relevance scale, the usual way is to test systems on standardized collections of documents and queries. However, we cannot rely on a standardized collection since our domain is the web. We can only attempt to measure how well information needs meet with user preferences. In our system, the NDPM measure based on relative judgments from users is more appropriate than precision and recall, which are based on absolute relevance judgments. Furthermore, directly comparing user and system rankings gives us a single-valued measure and a simple interpretation of the results.

The NDPM is defined in Equation 6.1.

$$ndpm = \delta(\succ U, \succ S)/\delta_{max} \qquad (6.1)$$

where δ is the distance performance measure, $\succ U$ and $\succ S$ are the user and system rankings, respectively, and δ_{max} is the value corresponding to the maximum of the δ function.

6.3.2 Procedure and result

In this experiment, a special list of documents containing many different topics is selected randomly from the database. The system ranks the documents according

to how well they match the current user profile, which is the system's prediction of the user's rankings. The user ranking can be obtained from the user's feedback. The experiment proceeds as follows:

1. Select m users and initialize their profiles to empty.
2. Set the profile size to n.
3. Generate a special list of documents from the database.
4. Rank the documents according to the current user profile.
5. Obtain the feedback rating for a document in the list from the current user.
6. Compute and record the distance between the system ranking and the user rating for the document.
7. Update the user profile based on the feedback rating in step 6.
8. Repeat steps 4 to 7 for all m users.
9. Compute the average distance over all users and the maximum distance.
10. Compute the NDPM.

In the experiment, the procedure should be run several times, and steps 1 and 2 are run only the first time. Step 2 sets n, the number of keywords for the descriptor in the profile. The value of n is also used to represent the size of document keywords. Step 4 generates the predicted ranks for the system based on Equation 4.8.

The evaluation of this experiment should be run several times. In my thesis, we give an example whose evaluation number is equal to 20. The more evaluation does,

the better the experiment result will be. The results for this experiment are shown in Figures 6.3 and 6.4. Figure 6.3 shows the average NDPM at each evaluation point where the size of profile, n, is equal to five. Figure 6.4 shows the average NDPM at each evaluation point where n=10. The NDPM distance between the user's and the system's rankings decreases gradually over time, as the system's model of the user becomes more accurate. In other words, the gradual downward progression indicates that the system is successfully learning increasingly accurate profiles of users over time. The best result of the test is the value of the NDPM is dropped to zero quickly. It is good that the value of NDPM dropped under 0.5 after 15 iterations in our test. There are several reasons that the value of the NDPM in our test does not reach to zero and exists upward and downward swings. First, the news articles we collected are arbitrary topics. Second, the profile is empty when we start our test, and the interest keywords are collected from the news articles. Further, when the system learns the short-term feedback, the positive and negative descriptors in the profile have reactive behavior, and can potentially forget the feature vectors learned from previous feedback.

From Figures 6.3 and 6.4, we find that the size of keywords affects the performance of the system. The larger the number of keywords representing the user's interests and the features of the article, the better the prediction the system generates.

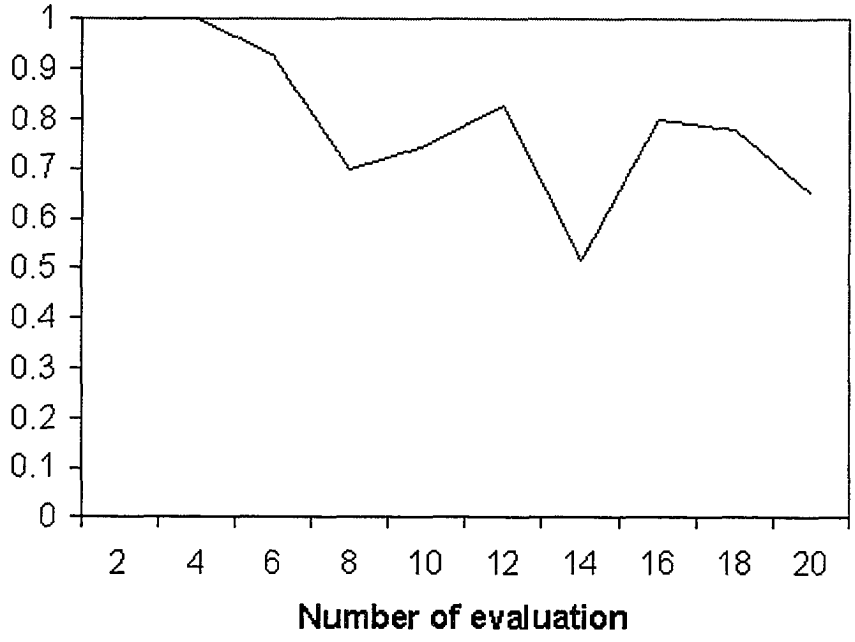


Figure 6.3: Average NDPM distance between user and system rankings (n=5)

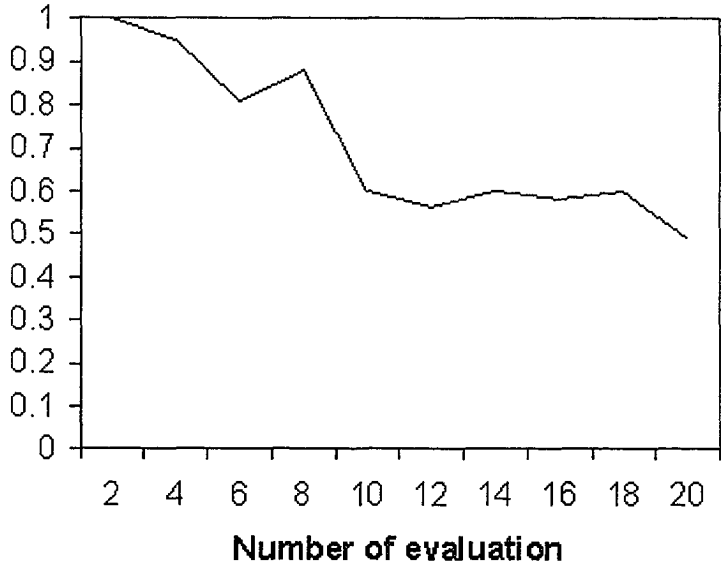


Figure 6.4: Average NDPM distance between user and system rankings (n=10)

Chapter 7

Conclusion

This chapter summarizes the highlights of this research and discusses future work.

7.1 Summary

In order to address difficulties in finding interesting news for users given widespread information overload, this thesis presents a solution to maintain user interests and predict which articles might interest the user. To do this, we developed a personalized news agent.

The basic function of the personalized news agent is to provide articles from online websites (e.g.Canada.com) for users. The agent creates a personalized news page for each user according to interest information stored in the profile. Retrieving and analyzing articles from the website, predicting the degree of interest in an article for a particular user, and tracking dynamic changes in user interests are key tasks the personalized news agent must accomplish. To succeed in these tasks, the news agent we developed is composed of a retrieval module, recommendation module, learning module, user interface, and admin module, where the retrieval, recommendation, and learning functions are core parts of the system. The retrieval module retrieves articles

from a specific online web site and stores the keyword information in the database. To accurately predict user interest, the recommendation module is implemented to score the degree of the interest in an article, and the learning module maintains and tracks changes in user interests.

A number of models and algorithms to learn the user profile have been proposed. In this thesis, the 3-descriptor scheme, which is feasible and effective to derive long-term and short-term user interests, is selected. Based on this approach, the learning algorithm is able to handle both long-term and short-term interests simultaneously. In order to improve the accuracy of the recommendations, 20% of feedback information is used to adjust the value of the learning rate by comparing the actual rating with the predicted rating for the article.

In this thesis, the scheme and the algorithm work on the vector space model. The TFIDF weighting scheme is chosen to capture the context of terms in a document. In order to score the document, content-based filtering technology is used to compare measure similarities between the document and the user profile.

Users play an important role in a personalized news agent that learns user interests from user feedback. If the users provide relevant information as feedback according to their interests, the learning process will be effective. This implies that it is important for the user to understand how the system works. Implicit feedback from user actions reduces this burden, although implicit feedback is less accurate than explicit feedback, and results in a minimal adjustment to user interests. Hence, both explicit and implicit feedback are considered in the system.

In the personalized news agent, we add an additional module to maintain system parameters and monitor changes to user profiles. This module is used to validate

the performance of the system. Since system parameters influence results in learning user interests and predicting ratings for articles, the administrator can adjust these parameters while the system is running.

7.2 Future work

In this thesis, a personalized news agent is developed to provide articles from the Canada.com web site to users. The lesson learned from this work is that the development of a recommendation system can be considered as an integral application using several proposed methods. This work can be extended in several directions in the future to address other issues in information gathering and filtering.

First, we only use the TFIDF method to analyze articles when we build the agent, so problems of synonymy and homonymy still exist. We could use the LSI method introduced in Chapter 2 to improve the retrieval module. Furthermore, we only gathered formatted information from Canada.com in this thesis, making it easy to learn the category of the article. However, most online information may not be organized by the same rules. Methods of gathering and analyzing information from different websites with our agent could be considered in future work.

Second, the filtering scheme employed in this thesis is based solely on the content of the document being evaluated. A collaborative approach could also be taken into account in the design of a personalized news agent. Both content-based and collaborative filtering have some advantages and disadvantages. In order to improve the performance of the system, we could find a way to integrate them and use both approaches simultaneously to incorporate the advantages of both filtering schemes.

Finally, since our agent collects implicit feedback from users by observing their

behavior, protection of privacy and security issues should be considered.

Bibliography

[1] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman, *Indexing by latent semantic analysis*, Journal of the American Society of Information Science **41** (1990), no. 6, 391–407.

[2] Christopher Fox, *Lexical analysis and stoplists*, Information retrieval: data structures and algorithms (1992), 102–130.

[3] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry, *Using collaborative filtering to weave an information tapestry*, Commun. ACM **35** (1992), no. 12, 61–70.

[4] M. J. Hannah, *Html reference guide*, Tech. report, Sandia National Laboraties, 1996.

[5] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl, *Framework for predicting collaborative filtering*, In Proceedings of the 1999 Conference on Research and Development in Information Retrieval, 1999.

[6] Ying Huang, *An intelligent adaptive news filtering system*, http://citeseer.ist.psu.edu/huang01intelligent.html.

[7] Sinead Hughes and Colm O'Riordan, *Collaborative filtering as means to reduce information overload*, Tech. report, Dept. of IT, NUI, Galway, Ireland, http://citeseer.ist.psu.edu/494054.html.

[8] Nicholas R. Jennings, *An agent-based approach for building complex software systems*, Commun. ACM **44** (2001), no. 4, 35–41.

[9] Thorsten Joachims, *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization*, Proceedings of ICML-97, 14th International Conference on Machine Learning (Nashville, US) (Douglas H. Fisher, ed.), Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 143–151.

[10] George Karypis, *Evaluation of item-based top-n recommendation algorithms*, CIKM, 2001, pp. 247–254.

[11] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Pragnesh Jay Modi Naveen Ashish, Ion Muslea, Andrew G. Philpot, and Sheila Tejada, *Modeling web sources for information integration*, Proc. Fifteenth National Conference on Artificial Intelligence, 1998.

[12] G. Kowalski, *Information Retrieval System: Theory and Implementation*, Kluwer Academic Publishers, 1997.

[13] Hung-Jen Lai, Ting-Peng Liang, and Y. C. Ku, *Customized internet news services based on customer profiles*, ICEC '03: Proceedings of the 5th international conference on Electronic commerce, ACM Press, 2003, pp. 225–229.

[14] Henry Lieberman, *Autonomous Interface Agents*, Proceedings of the ACM Conference on Computers and Human Interface, CHI-97 (Atlanta, Georgia), 1997.

[15] H. P. Luhn, *The automatic creation of literature abstracts*, IBM Journal of Research and Development, 2, 159–165.

[16] T. Mitchell, *Machine learning*, McGraw-Hill, 1997.

[17] K. J. Mock, *Hybrid hill-climbing and knowledge-based methods for intelligent news filtering*, In Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96), AAAI Press, 1996, pp. 48–53.

[18] Kenrick J. Mock and V. Rao Vemuri, *Information filtering via hill climbing, wordnet and index patterns*, Inf. Process. Manage. **33** (1997), no. 5, 633–644.

[19] Alexandros Moukas and Giorgos Zacharia, *Evolving a multiagent information filtering solution in Amalthaea*, Proceedings of the First International Conference on Autonomous Agents (Agents'97) (New York) (W. Lewis Johnson and Barbara Hayes-Roth, eds.), ACM Press, 5–8, 1997, pp. 394–403.

[20] H. S. Nwana, *Software agents: An overview*, Knowledge Engineering Review **11** (1995), no. 2, 205–244.

[21] Nick Papadopoulos and Dimitris Plexousakis, *The role of semantic relevance in dynamic user community management and the formulation of recommendations*, http://citeseer.ist.psu.edu/papadopoulos02role.html.

[22] P. Resnik, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, *Grouplens: An open architecture for collaborative filtering of netnews*, In Proceedings of the ACM, Conference on Computer Supported Cooperative Work, 1994, pp. 175–186.

[23] Colm O' Riordan and Humphrey Sorensen, *Information filtering and retrieval: An overview*, http://citeseer.ist.psu.edu/483228.html.

[24] Norvig P Russell S, *Artifical intelligence a modern approach*, Prentice-Hall, 1995.

[25] Hidekazu Sakagami and Tomonari Kamba, *Learning personal preferences on online newspaper articles from user behaviors*, Selected papers from the sixth international conference on World Wide Web, Elsevier Science Publishers Ltd., 1997, pp. 1447–1455.

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

[27] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl, *Item-based collaborative filtering recommendation algorithms*, World Wide Web, 2001, pp. 285–295.

[28] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jonathan L. Herlocker, Bradley N. Miller, and John Riedl, *Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system*, Computer Supported Cooperative Work, 1998, pp. 345–354.

[29] B.D. Sheth, *A learning approach to personalized information filtering.*, Thesis,Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

[30] Yoav Shoham, *An overview of agent-oriented programming*, Software agents, MIT Press, 1997, pp. 271–290.

[31] A. Tan and C. Teo, *Learning user profiles for personalized information dissemination*, In Proceedings, 1998 IEEE International Joint Conference on Neural Networks, Alaska, 1998, pp. 183–188.

[32] M. Weiss, *Electronic Commerce Technologies course notes*, 2003.

[33] Dwi H. Widyantoro, Thomas R. Ioerger, and John Yen, *An adaptive algorithm for learning changes in user interests*, CIKM, 1999, pp. 405–412.

[34] Dwi Hendratmo Widyantoro, *Dynamic modeling and learning user profile in personalized news agent*, Tech. report.

[35] Mike Wooldridge and P. Ciancarini, *Agent-Oriented Software Engineering: The State of the Art*, First Int. Workshop on Agent-Oriented Software Engineering (P. Ciancarini and M. Wooldridge, eds.), vol. 1957, Springer-Verlag, Berlin, 2000, pp. 1–28.

[36] T. Yan and H. Garcia-Molina, *SIFT—A tool for wide-area information dissemination*, Proc. 1995 USENIX Technical Conference (New Orleans), 1995, pp. 177–186.

[37] Y. Y. Yao, *Measuring retrieval effectiveness based on user preference of documents*, J. Am. Soc. Inf. Sci. **46** (1995), no. 2, 133–145.

[38] G. K. Zipf, *Human Behavior and the Principle of Least Effort*, Cambridge, Mass.: Addison-Wesley, 1949.

Appendix A

Example of Log File

This example shows the history change of profile for one user.

profile is updated at Wed Dec 22 14:56:42 EST 2004
Category Name: world
Positive Interest Keyword:
ikaho 0.0837
waters 0.0564
inns 0.0502
japan 0.0375
spas 0.0335
tap 0.0335
springs 0.067
scandal 0.0503
spa 0.0502
hot 0.0503

Negative Interest Keyword:
gaulle 0.0263
france 0.069

paris 0.0888
division 0.0296
liberated 0.0263
celebrations 0.0263
liberation 0.0525
chirac 0.0394
de 0.0394
french 0.0656

Long Term Interest Keyword:
gaulle 0.0282
france 0.0741
paris 0.0953
division 0.0318
liberated 0.0282
celebrations 0.0282
chirac 0.0423
liberation 0.0563
de 0.0423
french 0.0704

---

learning rate is changed at Wed Dec 22 14:57:30 EST 2004
Now learning rate is 0.5215

Appendix B

DTD for News Article

The DTD of XML format for representing news items in this thesis is shown as follows:

<! ELEMENT News(Category, Headline, ByLine, StoryPub, StoryDate, Content)>

<! ELEMENT Category (# PCDATA)>

<! ELEMENT Headline (#PCDATA)>

<! ELEMENT Byline (#PCDATA)>

<! ELEMENT StoryPub (#PCDATA)>

<! ELEMENT StoryDate (#PCDATA)>

<! ELEMENT Content (#PCDATA)>

Appendix C

Classes Definitions

The main classes in our personlized news agent include: NewsRetriever, DownloadNews, NewsReader, RecommendNews, UserProfileProcess. Following are the main classes definition.

```
public class NewsRetriever{
    public static void retrieveNews(String path){
        DownloadNews national;
        national=new DownloadNews("http://www.canada.com/news/national/",
                    path+"/tempNews/national",20);
        national.download();
        ...
    }
    public static boolean checkFile(String fileName) throws IOException{
        ...
    }
```

```
        public static Hashtable saveNews(String path) throws IOException {
            //check if the file is an article
            if checkfile(filename)
                        ...
            else
                        ...
        }
}
public class DownloadNews extends DirectorySpider {
        static URL firstURL;
        public String url;
        public String storePath;
        public int maxCount;
        //overwrite the method
        public List getNewLinks(HTMLPage page) {
        ...
        }
        protected void handleUCommandLineOption(String value) {
        ...
        }
        private String getDirectory(URL u) {
        ...
        }
        public void download(){
```

```
        ...
        }
}
public class NewsReader {
        public String path;
        public Hashtable newsSet;
        public int count;
        public NewsReader(String aPath){
                newsSet=new Hashtable();
                path=aPath+"/news";
        }
        public Vector getNewsByCategory(String category){
                return (Vector)newsSet.get(category);
        }
        public void createNewsSet(int amount,Hashtable newsList)throws Exception{
        ...
        }
        public void computeKeyWeight(Hashtable newsList)throws Exception{
        ...
        }
        ...
}
public class RecommendNews {
        public NewsReader newsReader;
```

```
        public UserProfileProcess profileProcess;
        public Vector newsVector;
        public RecommendNews(String path,String userId){
            ...
            profileProcess=new UserProfileProcess();
            profileProcess.setProfileFromDB(userId);
        }
        //based on the user profile,create the newslist
        public Vector getNewsByCategory(String category){
            ...
        }
        public NewsContent getNewsContent(int index){
            return ((ScoredNews)newsVector.get(index)).getNews();
        }
}
public class UserProfileProcess {
        public UserProfile profile;
        public void setProfileFromDB(String userId){
            ...
            profile=UserDBProcess.getUserProfile(userId);
        }
        public void learnUserFeedBack(String category,
                                HashMap newsKeyWeight, double learningRate) {
            ...
```