RPC-News: a real-time, personalized, Chinese news system

Da-Wei Chang
Ing-Chou Chen
Hao-Ren Ke and
Ruei-Chuan Chang

The authors

Da-Wei Chang, Ing-Chou Chen, Hao-Ren Ke, and **Ruei-Chuan Chang** are in the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, ROC.

Abstract

With the speedy growth of information quantity, people need a mechanism to discover automatically the information that interests them. Such a mechanism is called selective dissemination of information (SDI). Describes the design and implementation of an SDI system with the ability of delivering real-time, personalized news articles. In addition to delivering English news, it delivers Chinese articles also. Focuses on the problems that other researches seldom address. First, discusses how to store and delete news articles efficiently, then describes the user model to let users specify their interests. Finally, presents an efficient method to embed the ability to deliver Chinese as well as English news articles in the system.

Internet Research: Electronic Networking Applications and Policy
Volume 7 · Number 4 · 1997 · pp. 320–328
© MCB University Press · ISSN 1066-2243

Introduction

With the rapid development of internet technology, more and more digital information is available for users. However, facing such large volumes of information makes users confused about how to find what they want. They get lost in this "information jungle." Some internet tools were developed to help users solve this problem, e.g. the Electronic Bulletin Board System, Newsgroups, and Gopher group information in a hierarchical form so that users can quickly find information that interests them. WWW search engines such as Yahoo and Lycos provide a keyword-matching mechanism for users to search information that they are interested in. However, these tools cannot solve the problem completely. Although providing a way for users to look for existing information, they do not have the ability to deliver new information to users. Thus, users still have to look for information by these tools periodically so as not to miss something interesting.

Selective dissemination of information (SDI) systems as described in Malone *et al.* (1987) were then developed to solve this problem. They filter information automatically and continuously deliver it to users[1]. In most of these systems, a user can describe his interests in a list of queries, called a user profile. And the system will then deliver information to him according to his profile. Typically, a query will contain some keywords that users are interested in, and the information that contains these keywords will be delivered to users.

Although there is much nonEnglish information, most of the SDI systems support English only. Thus, multi-language filtering tools are needed. In this paper, we describe an SDI system that supports not only English but also Chinese. It follows the client-server model. Clients contact the server and express their interests. The server receives real-time news articles periodically and prepares relevant articles for clients. Clients can then browse information that has been prepared for them.

In this paper we address the problem of the design and implementation of a real SDI system, especially user-interest modeling and Chinese filtering.

The remainder of this paper is organized as follows. The next section briefly describes the research efforts on information filtering and several SDI systems. We then present our system architecture and discuss several design issues. The following section shows the implementation details of our system, and finally, we discuss the future works and provide our conclusions.

Related work

It has been shown that many information filtering issues are similar to information retrieval (IR) problems (Belkin and Croft, 1992) and therefore many IR technologies apply to the field of information filtering. There are many research topics in IR. Some researchers address the problem of how to specify user interests (i.e. user models) (Allen, 1990; Goldberg *et al.*, 1992). Some focus on the filtering models and filtering precision (Lewis *et al.*, 1989). Some compare the performance of different filtering approaches (Foltz and Dumais, 1992; Turtle and Croft, 1992). And others focus on the design and implementation of information filtering or retrieval systems. In the following we describe some of these systems briefly.

The Information Lens system (Malone *et al.*, 1987) is an intelligent information-sharing system based on e-mail. Instead of letting users express their interests by natural language, it employs semi-structured templates for users to express their interests. Users can specify a few rules based on these templates to filter messages in their mailboxes. A major difference between the Information Lens system and ours is the location of the filter. The former provides only local filtering. That is, it only filters the articles that come to the client side and does not discover relevant articles at any other places. Thus, users still have to look for all the interesting topics and subscribe to them on their own.

Tapestry (Goldberg *et al.*,1992) is an experimental e-mail system. Its most interesting feature is that users are involved in the filtering process. Articles are read by some users first, and they give comments to the system. The comments can then be used for filtering. However, it is not feasible to use Tapestry as a real-time, news-delivery system. There are two reasons for this. One is that it stores articles in an article store but does not provide a mechanism to delete articles from it. Thus, with the coming of large volumes of real-time news, the article store will become full soon. The other reason is that it uses a “little box” on the server side for each user to store articles prepared for him. Thus, for instance, if there are one thousand users interested in a specific article, one thousand copies of this article should be maintained on the server side. This would overload the server.

O’Kane (1996) outlines an information retrieval system which can, given a pile of articles, generate HTML files that are used to index the original articles. Users can then browse the HTML files and select the articles that really interest them. However, similar to the Information Lens system, it does not provide a mechanism to find relevant information at any other places.

To increase the scalability of an SDI system, Yan and Garcia-Molina (1994) suggest that multiple SDI servers are needed. They then adapt existing protocols to distribute profiles and articles among these servers.

SIFT (Stanford Information Filtering Tool) (Yan and Garcia-Molina, 1995) is an efficient SDI system. It implements a novel indexing technique named profile indexing to make the system able to deal with large numbers of articles. Similar to Tapestry, SIFT does not address the problem of how to delete articles from the article store efficiently.

Our SDI system, the RPC-News, differs from others in that it delivers multi-language(i.e. Chinese as well as English) news articles, and it provides a mechanism to delete articles efficiently. With multiple-language document delivery, our system can prepare more interesting articles written not only in English but also in other languages for users. A news delivery system usually has to store a large volume of news articles in its storage system. However, as the time goes by, these articles will become out of date and useless. Therefore, our system implements a mechanism that will be described later to delete these out-of-date documents.

System architecture and design

In our system, a user can subscribe to the newspaper via a WWW browser. At that time, he specifies his interests[2], together with other information such as user name, password, etc. And the information, which is called user subscription information, will then be stored in the user profile database.

After a moment, he can read the news satisfying his interests by using a WWW browser. In addition to WWW, our system uses e-mail to transfer news articles.

Because our system receives real-time news articles, they may come to the server side at any time. If there are any new articles prepared for a user while he is reading the news, the system will notice that and show the new articles to him. Thus, the user need not explicitly issue another request to get the newest articles periodically.

In the following we briefly describe the components of our system and their functions. We then discuss some design issues.

System components

This system can be subdivided into two layers: the internal and the external layers. The internal layer is responsible for processing data (i.e. real-time news articles), filtering them , and putting them in the right place. The external layer acts as the interface between the system and front-end users. Figure 1 shows the system architecture. The following paragraphs describe the system components in the two layers.

The internal layer

There are six major components in this layer. They are a news grabber, a preprocessing logic, a filtering engine, a news store, a meta-data database for results of filtering, and a user subscription (i.e. user profile) database.

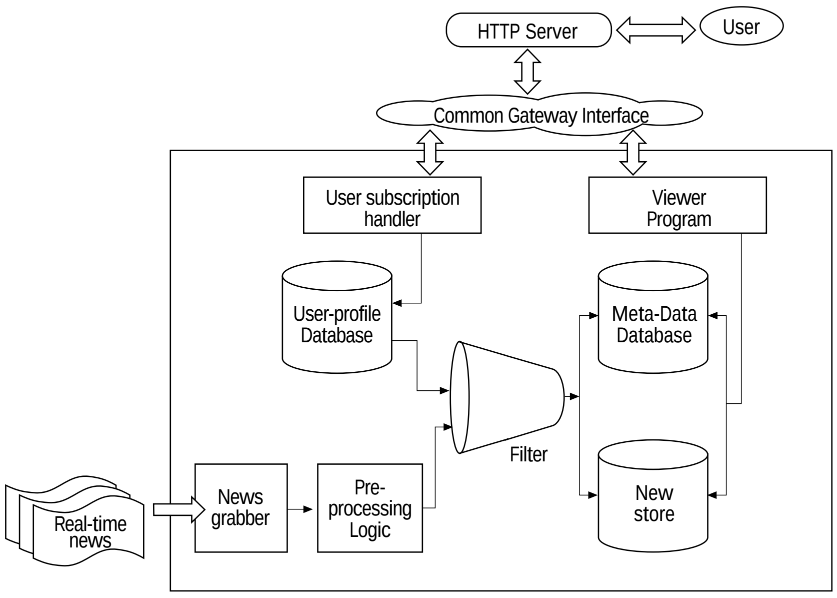
We have a news provider that feeds us real-time news periodically. And the news grabber is responsible for collecting new coming articles and hands them to the preprocessing logic and the filter. The preprocessing logic takes over housekeeping work so that the filter can concentrate on filtering articles without caring about other things. The filter matches the articles at hand with the key elements in the user profile database. If there are any key elements occurring in an article, the filter adds information about the article to the meta-data database. The information is then used to locate that article.

Each user is associated with a table that stores information about how to locate his personal news articles in the meta-data database. And we refer to the table as the “meta-data table” of that user. The last internal component, the news store, is the place where the news articles are stored after they have been filtered.

The external layer

Two major components of the external layer are the user subscription handler and the viewer program. Both of them are implemented as CGI (Common Gateway Interface) (W3C, 1996) programs. Thus, they reside on the server side and communicate with WWW users through a HTTP server. The user subscription handler is responsible for receiving user subscription

Figure 1 The system architecture



information and storing it into the user profile database. And the viewer program shows users the up-to-date information that interests them.

Design issues

In the following we discuss some problems and issues in the design of a personalized, Chinese, real-time news system. We also present some alternative approaches to deal with these problems.

User model

Most of the current SDI systems treat all the news articles as the same type. In these systems, users can only express their interests by providing a set of key elements. The systems then normally apply a key-element-matching mechanism to all of the articles. The drawback of these systems is that a user may receive too much irrelevant information, if the key elements he specifies are meaningful in many fields but the user is interested only in one or some of these fields.

Given that our news provider associates each article with a specific field, we can solve this problem by having users provide a set of (field name, key element) pairs as their interests. For example, if a user is interested in the Java programming language, he can specify ("Programming Language," "Java") as his interest. Therefore, he will not receive any information about "Java coffee."

A side-effect benefit of this model is that it is more efficient. For example, if a new coming news article belongs to a field that no users are interested in, we can just discard the article without filtering it.

Deletion from the news store

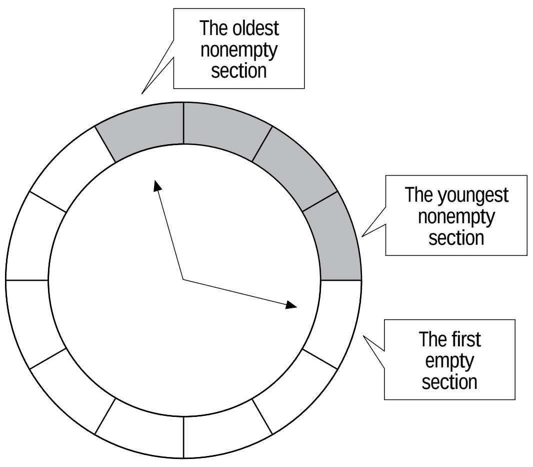
There may be thousands of articles from our news provider a day. Thus, the news store will become full quickly if we cannot delete articles that have been inserted into it. What kinds of articles should be deleted? A simple rule is to delete articles that are old enough. But how to achieve this? It is not a good idea to scan all of the articles and delete the oldest ones. An alternative way is to divide the news store into several sections. Each time we receive new articles, we place them into the newest (i.e. youngest) section. Thus, the age of an article can be viewed as the age of the section containing it. When we want to delete some articles from the store, we can just select the oldest sections and delete all the articles contained in them.

The section name should be assigned in a way so that we can easily figure out what the oldest sections are. For example, we can use a counter to record the section names. Each time we create a new section, we can use the current value of the counter as the name of the newly-created section and then increase the value of the counter. Thus, the oldest section has the "smallest" name. Therefore, we can identify the oldest sections in the news store by simply keeping track of the "smallest" section name in the store. Figure 2 shows the abstract structure of our news store. We can view it as a clock with two indicators. One indicator points to the oldest nonempty section and is used by the deletion logic of the news store. The other points to the first empty section and is used by the insertion logic of the news store.

Ways to transfer news

There are two common ways for news transmission: by e-mail and by WWW browsers. Our system implements both of them. The advantage to transmit news by e-mail is that it is easy to implement since we can eliminate the cost of maintaining the news store. When the news articles come in, we can just do the necessary processing (i.e. preprocessing and filtering) and then forward the articles to users interested in them. We need not store articles in the server side since we have given users all they want. Therefore, we do not need a news store and the time and space cost of maintaining it can be saved. However, there are at least three disadvantages in

Figure 2 The abstract structure of the news store



e-mail delivery. First, by using e-mail to transmit news, it requires users to check and clean their mailboxes frequently, or they will become full soon. Second, it causes a lot of network traffic. For each news article, the server would send it to all the users that are interested in it without considering if the users will "really" read them or not. Third, it is easy for your enemies or competitors to explode your mailbox by using your e-mail address to subscribe newspapers. This is known as an "e-mail bomb." To avoid it, our server sends a message to each subscriber. This message contains information which is only known by the server (i.e. a random number). Each subscriber must confirm the message by replying to it. The server will then check the reply and, if the reply information is correct, start preparing news articles for him. This e-mail validation approach prevents others using our system as a e-mail bomber. In addition, it gives the subscriber a second chance to consider seriously if they want to receive news articles by e-mail.

On the other hand, users can read news by WWW browsers. This way differs from the former because it gives users the ability to control the transmission of news articles. If users really want to read news, they can issue a request to the server; if they do not, none of the client-side resources would be consumed. Thus, it causes less network traffic and the "e-mail bomb" problem does not happen. Of course, we have to pay to maintain a news store on the server side.

Personal news location

If users read news by using WWW browsers, how can they (precisely speaking, the viewer programs) locate news articles that have been already prepared for them? A simple way is to build a personal output-data store for each user and store all articles prepared for a user in his output-data store like the "little box" in Goldberg *et al.* (1992). This method is space-consuming since one news article may have as many copies stored on the server side as the number of users. We hence use an alternative approach.

Instead of building an output-data store for each user, we construct a meta-data database that contains all the information needed to get the output data of all users. As we have mentioned earlier, the database itself is a set of tables (i.e. meta-data tables) where each table corresponds to a user. When a user wants to read news, the viewer program grabs the whole meta-data table that corresponds to him. He can then browse the table to see the articles prepared for him and select what he wants to read. The format of a meta-data table will be described in the next section.

Chinese filtering

One of the design goals of our system is to make the filter capable of filtering Chinese as well as English articles. However, the differences between Chinese and English make it hard to design such a filter. Before explaining the difficulties on the design, we first look at the differences between Chinese and English.

There are two major differences between Chinese and English. First, an English character can be represented by an ASCII number less than 128. But a Chinese character must be represented by a pair of ASCII numbers with the first number greater than 128. Second, an English sentence is made up by several words that are separated by one or more blanks from each other. And, in many cases, a key element may just be a word like "Java," "web," "browser," etc. However, a Chinese sentence is a concatenation of Chinese characters. That is, there are no blanks between two adjacent characters. And, the worst of all, we cannot use one character to be a key element since only one character itself may mean nothing. In general, the smallest meaningful unit in Chinese semantics consists of more than one character.

The following is an example to explain the difficulty in the design of Chinese-article filters. "abcdef" is a Chinese sentence where a, b, c, d, e, f are all Chinese characters. Though we know that "cd" is a substring of this sentence, we cannot determine whether it is a meaningful phrase in this sentence without knowing the semantics of the whole sentence. However, there is no efficient way for computers to know the semantics of a given sentence.

How do we cope with this difficulty? One approach is to ignore the situation that we describe above. That is, instead of determining whether a key element is a meaningful phrase in a given sentence, we just consider if the key element is in this sentence or not. This makes implementation easier but degrades the accuracy of a filter. The other approach is to maintain a database to keep the most commonly used phrases to help us extract the most likely meaningful phrases in a sentence so we can then match them with key elements. This approach improves the accuracy of a filter but makes the implementation more expensive. In our current implementation, we prefer using the former.

Implementation

In this section we propose our user model, followed by a detailed description of our implementation, including the tasks accomplished by each system component and the format of each database. Finally, we show the run time environment of our system.

User-interest modeling

As we described earlier, each user can provide a set of (field name, key element) pairs as his interests. To do so, our news provider defines a variety of field names and associates each article with one of them. Furthermore, we construct a field tree based on these pre-defined field names. Figure 3 illustrates a part of our field tree. All leaf nodes are pre-defined names. All nonleaf nodes are constructed by us to group the leaves in a hierarchy so that users can specify their interests more easily. For example, in Figure 3, if user X is interested in articles about key-element-A that belongs to either "Domestic Politics" or "International Politics," he can simply specify ("Politics," key-element-A) instead of specifying a couple of pairs ("Domestic Politics," key-element-A) and ("International Politics," key-element-A) as his interests.

Implementation details

We describe our implementation details in this section.

The user subscription handler

In our system, a user subscribes the newspaper by going to a URL that embeds our user subscription handler (i.e. a CGI program) via a WWW browser. He then fills in his information and sends it to the handler. The information consists of his name, password, e-mail address, the way to transfer news (i.e. by e-mail or by WWW browsers), and his interests, which are represented by the user-interest modeling described in earlier in this section. After the handler receives the information, it stores the information in the user profile database which will be loaded into main memory by the filter.

The news grabber and the preprocessing logic

Our news provider sends real-time news articles to us via FTP. This simplifies the work of our news grabber since it does not have to retrieve news articles from the news provider periodically. The only task of the news grabber is to put these articles into the right place where our preprocessing logic can access them. Because many of the articles are written in Chinese, and a single Chinese phrase may be separated by newline characters[3], it is difficult for a filter to get a line from an article each time and match it with key elements in the user profile database. Thus, our preprocessing logic erases the newline characters if they occur in a Chinese sentence so that our filter can get a line each time without worrying that it may contain any partial key elements.

The filter

After an article is preprocessed, the preprocessing logic hands it to the filter. The filter then determines which users are interested in it. To achieve this, the filter constructs an index structure[4] from the user profile database in advance. A snapshot of our *index structure* is shown in Figure 4. There are three kinds of links in it. First, we have an array of *hash table links.* Each hash table link is associated with a pre-defined document field and points to a hash table. In other words, each document field has a unique hash table associated with it. Second, each element in a hash table chains a list of key elements via the *key element links.* Third, each key element chains a list of users via the *user links.*

We explain how to construct the index structure now. Table I shows an example of our profile database. To add information about the first record, (userX, ("Politics," key-element-A), other-attributes-of-X), the following steps are executed. First, the filter traverses the field tree

Figure 3 A part of our field tree

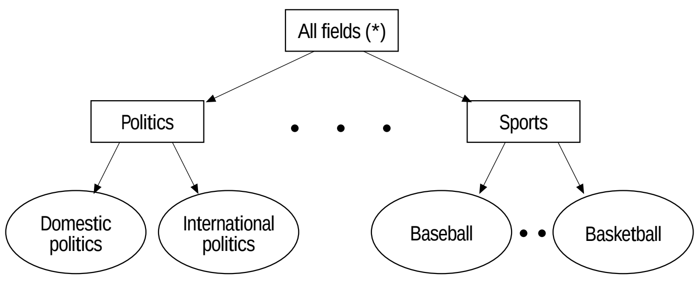
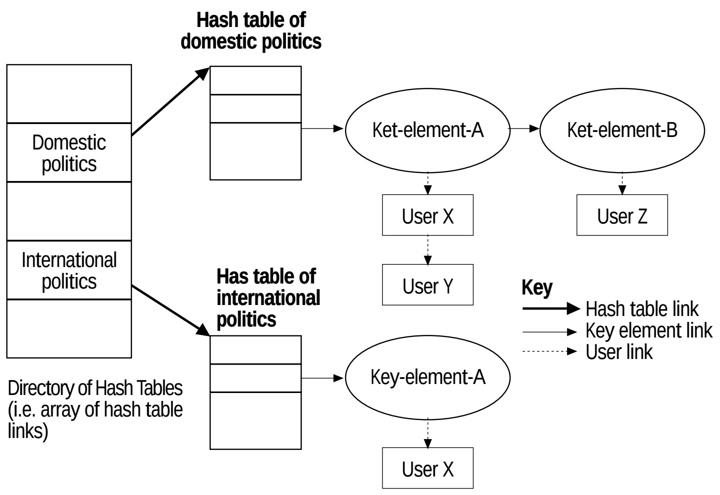


Figure 4 The index structure



described earlier in this section and finds out the child-and-leaf nodes of the node "Politics": the "Domestic Politics" and the "International Politics." For each of the two document types, the filter locates the corresponding hash table via the hash table link, uses the first byte of the key-element-A as input of our global hash function[5] to index the hash table and gets the key element list. Then, the filter inserts the key-element-A into the key element lists, and finally inserts the userX into the user list of each key-element-A.

After the information about all of the records in our profile database has been added, the construction process of the index structure is completed. Figure 4 shows the result of our index structure after the information on all the records of Table I has been added.

We now explain how the filter determines which users are interested in an article after it has constructed the index structure. With a given article at hand, the filter finds out the field of it and identifies the hash table corresponding to that field. For each paragraph in that article, the filter fetches a Chinese character (or an English word) each time and, as with the construction of our index structure, uses the first byte of that character (or word) as input of the hash function to index the hash table and hence locates a key element list. It then scans the list to see if there are any key elements contained in that line[6]. If there are, it increases the score numbers[7] of all the users in the user lists of those key elements.

After the whole article has been processed, the filter checks the scores of all users. If a score is larger than a pre-determined threshold, the filter considers that the user is interested in this article and therefore adds information about the article to the meta-data table corresponding to him.

In this paragraph, we discuss the reason for using only the first byte as input of the hash function. If we use the whole key element instead, the only way to locate it via the hash function would be to know exactly what it is. However, as we describe above, it is difficult to extract any key elements (i.e. meaningful phrases) from a Chinese sentence without knowing its semantics. In contrast, our approach allows the filter to locate any key elements by just getting a Chinese character each time, calling the hash function, and then doing some string-matching. It realizes the combination of Chinese and English filtering.

The viewer program and the meta-data database

After a user has subscribed to the newspapers, he can actively contact the viewer program via a WWW browser to read the news articles prepared for him, or passively receive news by e-mail. Each time the viewer program is contacted, it authenticates the user by asking for his user name and password. If he is a valid user, it sends his meta-data table to him. Thus, he can browse the table and select what he wants to read.

A meta-data table consists of a list of records and each record corresponds to an article prepared for that user. There are two major attributes in a record, the title and the location (i.e. the file name) of the article. The former is presented to the user to let him select articles for reading. If he does select an article title, the viewer program translates the title into its location, reads the article content from the file system by using the location, and sends the content back to the user.

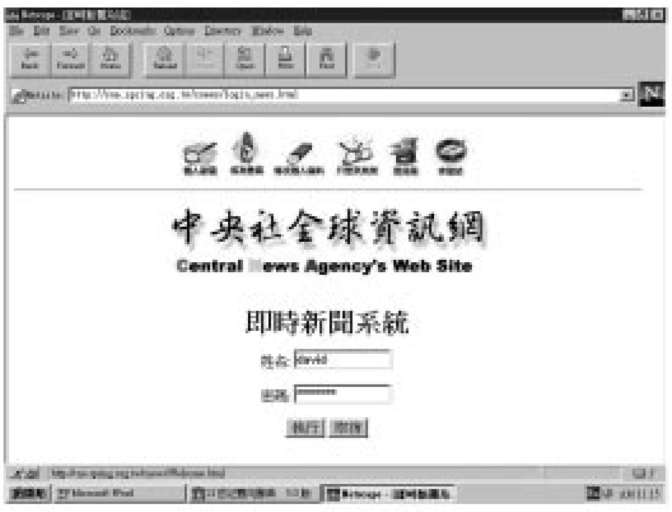
Table I An example of the user profile database

User	Interests (field name, key element)	Other attributes
UserX	("Politics", key-element-A)	Other-attributes-of-X
UserY	("Domestic Politics", key-element-A)	Other-attributes-of-Y
UserZ	("Domestic Politics", key-element-B)	Other-attributes-of-Z

The run-time environment

Our system is implemented in C and now runs on Solaris 2.5.1. The user run-time environment is shown in Figures 5 to 8, and the URL of our system is http://cna.spring.org.tw/cnews.

Figure 5 The user login screen



即時新聞系統

姓名: david

密碼: ********

執行 取消

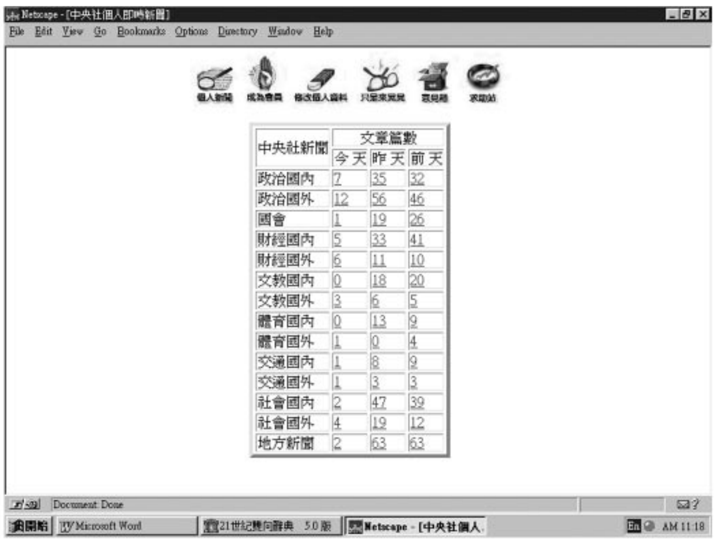
Note: In this screen, the user provides his name and password

Conclusions

SDI systems will become more valuable with the explosion of information. However, they seldom provide filtering on nonEnglish articles. In this paper, we discuss the design and implementation issues on a personalized, Chinese, real-time news system. We also present an approach to combine Chinese and English filtering neatly.

Some advanced issues that we do not discuss here will be addressed in future. For example, the deletion strategy in our current implementation is just to delete the oldest articles. However, the oldest articles may be the most popular ones. Thus, we should replace the current strategy with a more complex one. In addition, the performance and scalability of the system could be improved further, so that it can deal with larger volumes of news articles and more feasibly become a real, personalized, real-time news system.

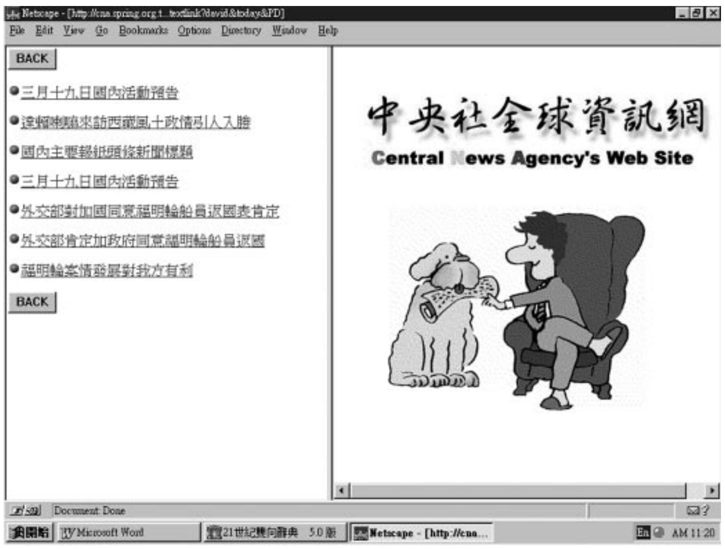
Figure 6 Number of articles in all fields



中央社新聞	文章篇數		
	今天	昨天	前天
政治國內	7	35	32
政治國外	12	56	46
國會	1	19	26
財經國內	5	33	41
財經國外	6	11	10
文教國內	0	18	20
文教國外	3	6	5
體育國內	0	13	9
體育國外	1	0	4
交通國內	1	8	9
交通國外	1	3	3
社會國內	2	47	39
社會國外	4	19	12
地方新聞	2	63	63

Note: The above table contains the number of articles in all fields that satisfied the user's profile during the recent three days. Each row represents a document field. The first column lists the document field names and the second to fourth columns list the numbers of selected articles from today, yesterday and the day before yesterday, respectively. The user can read documents of a specific field and a specific day (i.e. today, yersterday or the day before yesterday) by selecting the corresponding number in the table

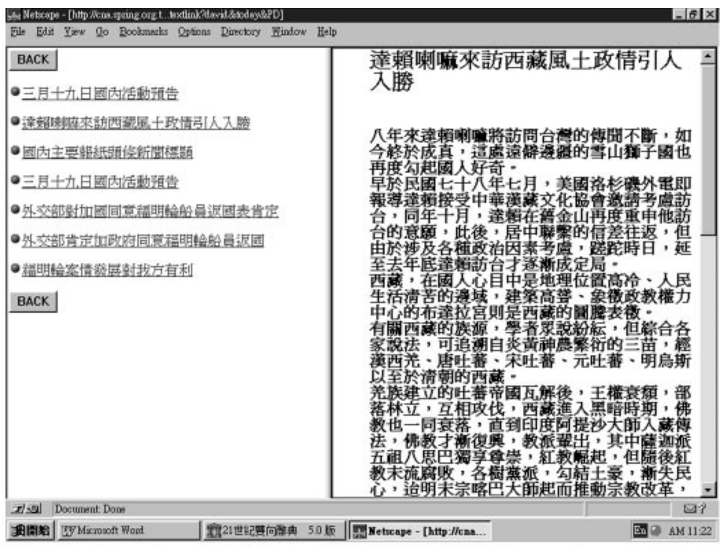
Figure 7 Document titles



- 三月十九日國內活動預告
- 達賴喇嘛來訪西藏風土政情引人入勝
- 國內主要報紙頭條新聞標題
- 三月十九日國內活動預告
- 外交部對加國同意福明輪船員返國表肯定
- 外交部肯定加政府同意福明輪船員返國
- 福明輪案情發展對我方有利

Note: After the user has selected a specific number the left frame shows the document titles (i.e. a portion of the user's meta-data table) that correspond to a document field. The user can now select one of the titles to read the full text.

Figure 8 The full text of a document



- 三月十九日國內活動預告
- 達賴喇嘛來訪西藏風土政情引人入勝
- 國內主要報紙頭條新聞標題
- 三月十九日國內活動預告
- 外交部對加國同意福明輪船員返國表肯定
- 外交部肯定加政府同意福明輪船員返國
- 福明輪案情發展對我方有利

達賴喇嘛來訪西藏風土政情引人入勝

八年來達賴喇嘛將訪問台灣的傳聞不斷，如今終於成真，這處遠僻邊疆的雪山獅子國也再度勾起國人好奇。
早於民國七十八年七月，美國洛杉磯外電即報導達賴接受中華漢藏文化協會邀請考慮訪台，同年十月，達賴在舊金山再度重申他訪台的意願，此後，居中聯繫的信差往返，但由於涉及各種政治因素考慮，蹉跎時日，延至去年底達賴訪台才逐漸成定局。
西藏，在國人心目中是地理位置高冷、人民生活清苦的邊域，建築高聳、象徵政教權力中心的布達拉宮則是西藏的圖騰表徵。
有關西藏的族源，學者眾說紛紜，但綜合各家說法，可追溯自炎黃神農繁衍的三苗，經漢西羌、唐吐蕃、宋吐蕃、元吐蕃、明烏斯以至於清朝的西藏。
羌族建立的吐蕃帝國瓦解後，王權衰頹，部落林立，互相攻伐，西藏進入黑暗時期，佛教也一同衰落，直到印度阿提沙大師入藏傳法，佛教才漸復興，教派輩出，其中薩迦派五祖八思巴獨享尊崇，紅教崛起，但隨後紅教末流腐敗，各樹黨派，勾結土豪，漸失民心，迨明末宗喀巴大師起而推動宗教改革，

Note: After the user has selected a document title, the full text of that document is shown in the right frame.

Notes

1 Some SDI systems do not deliver information to users, they just *prepare* (i.e. store) it and wait for users to retrieve their own information.

2 In general, user interest contains a set of keywords. However, since the articles are not only in English but also in Chinese, the keywords that he provides may be Chinese phrases. That is, a keyword may not just be a word. So, we prefer using "key elements" instead of "keywords" later in this paper.

3 That is, a line may contain a partial key element.

4 The index structure is an extension of the profile indexing used in SIFT (Yan and Garcia-Molina, 1995).

5 For simplicity, although we have multiple hash tables, we use a single hash function for all of them. And the reason for using only the first byte instead of the whole key element as input of it will be described later in this section.

6 Precisely speaking, the filter matches each key element in that list with the string that starts at that Chinese character (or English word) and stops at the end of the line.

7 Each user has a corresponding score number which is set to zero by the filter before filtering an article.

References and further reading

Allen, R.B. (1990), "User models: theory, method, and practice," *International Journal of the Man-Machine Studies,* Vol. 32 No. 5, May, pp. 511-43.

Belkin, N.J. and Croft, W.B. (1992), "Information filtering and information retrieval: two sides of the same coin?," *Communications of the ACM,* Vol. 35 No. 12, December, pp. 29-38.

Foltz, P.W. and Dumais, S.T. (1992), "Personalized information delivery: an analysis of information filtering methods," *Communications of the ACM*, Vol. 35 No. 12, December, pp. 51-60.

Goldberg, D., Nichols, D., Oki, B. M. and Terry, D. (1992), "Using collaborative filtering to weave an information tapestry," *Communications of the ACM,* Vol. 35 No. 12, December, pp. 61-70.

Lewis, D.D., Croft, W.B. and Bhandaru, N. (1989), "Language-oriented information retrieval," *International Journal of the Intelligent Systems*, Vol. 40 No. 3, Fall, pp. 285-318.

Malone, T.W., Grant, K.R., Turbak, F.A., Brobst, S.A. and Cohen, M.D. (1987), "Intelligent information-sharing systems," *Communications of the ACM*, Vol. 30 No. 5, May, pp. 390-402.

O'Kane, K.C. (1996), "World wide web-based information storage and retrieval," *ONLINE & CDROM REVIEW*, Vol. 20 No. 1, February, pp. 11-20.

Salton, G. (1989), *Automatic Text Processing*, Addison Wesley, Reading, MA.

Schulzrinne, H. (1996), "World wide web: whence, whither, what next?" *IEEE Network,* Vol. 10 No. 2, March, pp. 10-17.

Turtle, H. R. and Croft, W. B. (1992), "A comparison of text retrieval models," *The Computer Journal,* Vol. 35 No. 3, June 1992, pp. 279-90.

W3C (1996), "CGI: Common Gateway Interface," on line document, http://www.w3.org/pub/WWW/CGII, World Wide Web Consortium (W3C), last updated at May 1996.

Yan, T.W. and Garcia-Molina, H. (1994), "Distributed selective dissemination of information," *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*, September, pp. 89-98.

Yan, T.W. and Garcia-Molina, H. (1995), "SIFT – A tool for wide-area information dissemination," *Proceedings of the 1995 USENIX Technical Conference,* January, pp. 177-86.