University of Alberta

Goal-Directed Complete-Web Recommendation

by

Tingshao Zhu

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Spring 2006

Your file Votre référence
ISBN: 0-494-14074-7
Our file Notre référence
ISBN: 0-494-14074-7

To my parents

Abstract

While the World Wide Web (WWW) contains a vast quantity of information, it is often difficult for Web users to find the information they seek. There are many recommender systems that are designed to help users find relevant information on the Web; however, as many of these systems are server-side, they can only provide information about one specific Web site and they are typically based only on correlations amongst the pages that the various users visit. Unfortunately, there is no reason to believe that these correlated pages will necessarily contain useful information.

Here, a passive Goal-Directed Complete-Web (*GCW*) recommender system, which recommends relevant pages from anywhere on the Web to satisfy the user's current information need without any explicit additional input, has been developed. After identifying the search strategy that is employed by actual users while they browse the Web, the model attempts to locate the pages that satisfy the user's information need based on the content of the pages the user has visited, and the actions the user has applied to these pages.

To build such models, I develop a number of *browsing features* — browsing properties of the words, in the context of the current session — to capture the actions of the Web user. Because the method is based on how the words are used (while training on these browsing feature values), it can be applied to make predictions about pages that have never been visited. This model is therefore independent of users, specific words and specific Web pages, and so it can be used to identify relevant pages in any new Web environment.

To evaluate the predictive models, we have conducted two user studies, each involving over one hundred participants. Data from the user studies demonstrate that the models can effectively identify the information needs of new users, leading them to previously unseen, but relevant pages.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People are spending increasing amounts of time on the World Wide Web (WWW) searching for information, shopping, playing on-line games, etc. The WWW has become the primary information source for many people. While the WWW contains a vast quantity of information, it is often difficult for web users to find the information they are seeking. One way to access information from the WWW is to simply type in URLs (Uniform Resource Locator), but this requires the user to explicitly know the specific URL that is needed. Another approach is to follow links from one web page to another, but here again, the user must know which links to follow. In addition, most users employ information retrieval techniques in the form of popular search engines to find useful pages. Search engines, however, can only work if the users can accurately identify the keywords appropriate for the search. Since the user may not know how to find specific needs — e.g., which hyperlink to follow or which keywords to feed into the search engine — it is critical to help her, by suggesting relevant pages that address her current information need.

1.1 Goal-Directed Complete-Web Recommendation

Many recommendation systems have been developed to enable Web users to browse the Internet efficiently [38, 53]. Most Web recommendation systems employ patterns that are based on the frequency and co-occurrence of the visited page, such as association rules [2] and sequential patterns [3]. Unfortunately, such systems require a non-trivial support (i.e., many visits to each relevant page) to make each inference. This is adequate when searching a single site with hundreds or even thousands of pages, but if the goal is to search the entire web with billions of pages, new methods must be developed since these co-occurence based approaches can only make recommendations over a much smaller number of pages.

Another shortcoming of such recommendation systems is that they can only point users to pages that other users have visited. Unfortunately, these "visit-correlated" pages do not necessarily contain information useful to the current user. Indeed, these suggested pages may correspond simply to irrelevant pages on the paths that others have taken toward their various goals, or worse, they may be standard dead-ends that everyone seems to hit. For example, even though many Web users often visit a company homepages, these pages typically do not provide the specific information the user is seeking.

Obviously, it is critical for a recommender system to generate useful recommendations, as irrelevant recommendations will discourage or even annoy users. This motivates us to develop computational techniques to assist the user in finding "*Information Content*" pages (i.e., IC-pages for short). An IC-page is a page the user must examine to complete her search task, e.g., the page containing the article that the user wants to download, the page

includes the product that the user wants to purchase, etc.

The goal of my research is a passive Goal-Directed Complete-Web (GCW) recommender system:

Passive: The system will not require any explicit input from the user. Instead it will gather the information it needs by merely observing the user's browsing behavior.

Goal-Directed: It will locate pages that satisfy the user's current information need and not just pages that others have found.

Complete-Web: It will suggest pages from *anywhere on the Web* rather than a single site. This means the standard techniques (e.g., association rule [2] and sequential pattern [3]) are inapplicable.

The recommender system must be *dynamic* as well. As a user works on various tasks and subtasks, her needs often change dramatically from day to day and even in the course of a single browsing session. Indeed, a typical user will often require information on various unrelated topics, including topics the user has never investigated before. These fundamental observations suggest that a recommender system needs to predict the user's information need dynamically based on the current browsing session, not just long-term general interests. Moreover, a dynamic system can also adapt to different user communities and different domains.

1.2 Thesis Statement

I propose to develop Web browsing behavior models that infer a user's information need based on her actions with respect to the viewed information, and moreover, show that these modles can be learned from previous annotated data.

My research demonstrates that browsing features of words are sufficient to recommend relevant pages from anywhere on the Web, satisfying the user's current information need, without requiring the user to provide any explicit input.

The browsing behavior models use a novel source of information to provide recommendations when other paradigms cannot.

1.3 Outline

The remaining chapters are organized as follows. Chapter 2 summarizes other related work to date, and explains how my work has different objectives from the recommender systems currently in use.

Chapter 3 describes *browsing features* and a *browsing behavior model*, and demonstrates how to extract browsing features from labeled sessions and train behavior models. The next chapters demonstrate my method for predicting IC-pages from *anywhere on the Web*, through its application in two user studies: Travel Planning (in Chapter 4) is a laboratory study on limited domains, and focuses on identifying relevant *words* based on browsing features. LILAC (in Chapter 5) is a field study of unrestricted browsing, introducing more browsing features and a new method of locating relevant pages from the Web. Both user studies indicate that the browsing behavior model can predict previously unseen relevant pages effectively. Finally, Chapter 6 concludes the present study and indicates future directions for research.

Chapter 2

Related Work

The Goal-Directed Complete-Web recommender system (*GCW*) is designed to suggest relevant pages to satisfy the user's information need. To do so, it trains Web user browsing behavior models that infers the user's information need based on how users search for information on the Web. This chapter presents a number of other methods (Co-occurrence Based Methods, Collaborative Filtering, Heuristic-Based Methods, and Content Based Methods) developed for Web recommender system, and also some research on modeling Web browsing behavior (General User Models and Customer Behavior Models), and discusses how they differ from my task.

2.1 Co-occurrence Based Methods

There are several co-occurrence based methods that can be used to recommend Web pages. Most such systems generate recommendations based on correlations amongst the pages on a specific Web site that various users have visited. Figure 2.1 shows some pages from a Web site, where each p_i represents a specific Web page (i.e., URL), and each arrow means there exists a hyperlink in the source page pointing to the target page. For example, p_1 has two links point to p_2 and p_5.

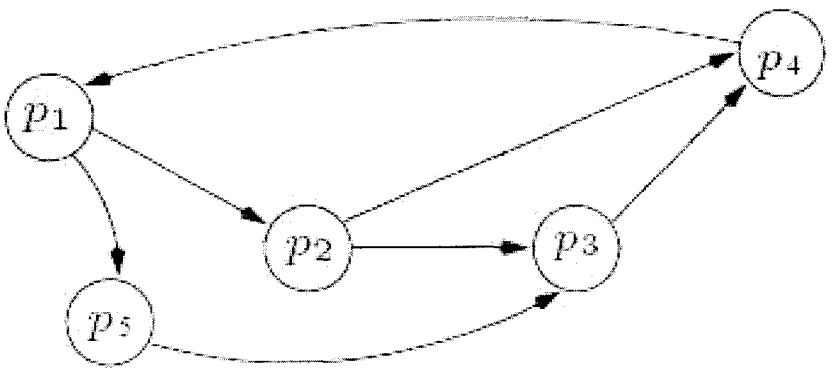

Figure 2.1: Part of a Web Site

Association Rules

Association rule mining has been well studied in Data Mining, especially for basket transaction data analysis. In our Web recommendation context, an *association rule* is a rule of the form "$p_1 \rightarrow p_3$" where the p_is are Web pages, with the intended

meaning that a user who has visited the pages on the left side (here, p_1), will typically also visit the pages on the right side (here p_3).

The algorithm that finds association rules, first need to identify all "large itemsets", which are sets of items that have transaction support above minimum support. The algorithm then generates all association rules from these large itemsets.

Many association rule mining algorithms have been proposed, such as Apriori [2, 70, 51, 34], Partition [63], and DHP [47]. Distributed and parallel association rule mining methods are also reported [14, 46]. Almost all of these research focus on finding association rules as fast as possible, rather than validating the usefulness of such rules in supermarket transaction or in Web application. For example, in Figure 2.1, p_1 is the home page of the Web site , p_2 and p_3 are intermediate pages leading to the pages containing useful information. Since people have to go through these intermediate pages to reach any content pages, the association rules that extracted from the Web server log might be "$p_1 \rightarrow p_2$" and "$p_1 \rightarrow p_3$". Obviously, the recommendations (e.g., p_2 or p_3) based on these association rules are not content pages.

Sequential Patterns

Given a collection of customer transactions, a sequential patterns is a maximal sequences among all sequences that have a certain user-specified minimum support [4, 37]. Several algorithms have been developed for efficient mining of sequential patterns, e.g., Web access pattern tree [52] and FreeSpan [22].

A sequential pattern is a list of URLs in the form of $\{p_1, p_2, p_3, p_1\}$, where the p_is are specific pages (see Figure 2.1). If the user has visited p_1 and p_2, then p_3 and p_1 could be suggested as a recommendation.

Page Clustering

Here, page clustering is the process of clustering pages according to the users' access to them, independent of page content and linkage [21]. The assumption is if a user who has visited p_1 in Figure 2.1 would be most likely to visit p_3, then p_1 and p_3 should be grouped into one cluster.

PageGather [54] assumes that for each Web site, only a subset of its Web pages are of importance to the visitors. It first computes the co-occurrence frequency between any pair of Web pages, then builds a graph where each node is a page and each arc connects two pages if their co-occurrence value is non-zero. The page clusters are the cliques in the graph.

Association Rule Hypergraph Partitioning (ARHP) [54, 39, 38] is a hypergraph partitioning techniques based on the large itemsets generated by the association rule mining. (A hypergraph is an extension of a graph that contains hyperedges, which each hyperedge connects two or more nodes.) In ARHP, each large itemset is represented by a hyperedge, and page clusters are these partitions identified by a hypergraph partitioning algorithm.

Each page cluster is a set of specific URLs. To generate recommendations, the recommender system identifies the page cluster that contains the current browsing session, then pick out URLs in the cluster as recommendations.

Hypertext Probabilistic Grammar (HPG)

HPG [9, 8] attempts to capture user web navigation patterns. The trails a user can follow when navigating through a Web site are assumed to be generated by a HPG — e.g., each node corresponds to a Web page and the arcs represent the links between pages. The weight of each arc is the probability that the user will follow the link, which can be computed from these browsing sessions.

Given the current user session, we can identify the most likely path in the HPG that can generate the session, and use it to recommend the link that the user might follow. HPG can also be used to improve the quality of Web service, and act as a personal assistant integrated with the Web browser.

Pattern Discovery over Aggregated Data

WUM [65, 66] merges the user browsing sessions into an Aggregated Tree, which is a tree constructed by merging trails with the same prefix page sequence. A tree node corresponds to a Web page in a session and is annotated with the number of visitors having reached this page across the same prefix. This value is the "support" of a node, computed in the context of the node's predecessors up to the aggregate tree's root. The traffic of a trail is then defined as the support of the tree leaf corresponding to the last node of the trail.

WUM extracts pattern descriptor from the Aggregated Tree. A pattern descriptor is a sequence of identifiers and wildcards, where an identifier refers to an occurrence of a Web page. For example a pattern descriptor extracted from Figure 2.1 is $\{p_1, *, p_3\}$, where $*$ is a wildcard character. If the user has visited $\{p_1, p_2\}$, then p_3 will be suggested to the user as a recommendation.

The above models are built to produce recommendations of specific URLs based on co-occurrence. Unfortunately, there is no reason to believe that these correlated pages will

contain information useful. These suggested pages may correspond simply to intermediate pages (e.g., p_2 and p_3 in Figure 2.1) on the paths that others have taken towards their various goals. Here, a GCW recommender system extends this, as it can suggest relevant pages from *anywhere on the Web*.

2.2 Collaborative Filtering

Collaborative Filtering [59, 50] produces recommendations by computing the similarity between the user's preference and the preference of other people, and it is the first attempt using Artificial Intelligence (AI) technology for personalization [41]. The basic mechanism behind collaborative filtering systems is to recommend based on a large group of people's preferences.

This requires the user to rank pages explored, and it is very difficult to get enough manually labeled Web pages in the real world. The Web is so diverse, and its users have such varied backgrounds and interests, it is impractical to identify relevant recommendations based on collaborative filtering only. GCW's task is different in that it is a passive recommender system, which recommends relevant pages without any explicit additional input (e.g., labelling Web pages).

2.3 Heuristic-Based Methods

Letizia [31] is an agent that tracks user browsing behavior and attempts to predict useful Web pages. The agent infers user interest (i.e., keywords) from browsing behavior based on a simple set of heuristics, and scouts from the user's current position to find pages that match the user's interest.

Watson [11] observes how users interact with everyday applications and anticipates their information needs using heuristics. It then automatically forms queries to information retrieval systems (e.g., search engines) to find the related information for the user.

The heuristic patterns used by Letizia and Watson are hand-coded. While they may represent the users behavior, we expect models learned from actual user data, will be more accurate. Therefore, GCW learns such patterns by training Web browsing behavior models using actual data.

2.4 Content Based Methods

Billsus and Pazzani [6] applied two models to recommend news stories to a user. These models were based on hand-selected words and Boolean feature word vectors. Unfortunately,

there do not use explicit feedback from the subject: instead, they only infer the level of interest in the news story based on the listener's actions, such as channel changes. It is also very difficult to guarantee that the selected words can cover all possible articles, and as a result, the trained model would be incapable of making any predictions in a new environment.

Jennings and Higuchi [25] trained one neural network for each user to represent a user's preferences for news articles. Since the Web user's interests may drift often and the content of the Internet changes from time to time, the neural network built on previous navigation history cannot adapt to rapid task changes.

Anderson and Horvitz [5] built a NaïveBayes model to predict the candidates (pages or topics) that the user will view next in a session, selected from the previous pages or topics in the same session. However, they provide no proof that the page the user has previously visited will be the page that she really wants.

Alternatively, GCW suggests Web pages by using the browsing features of the viewed information, independent of specific words, thus it can be used to recommend relevant pages in any novel Web environment.

2.5 General User Models

Blackmon et al. [7] proposed a Cognitive Walkthrough for the Web model for limited WWW interaction. The model uses Latent Semantic Analysis to compute the similarity between goal statements input by the user and heading/link texts on web pages, rather than the subjective estimation in the original Cognitive Walkthrough model. Unfortunately, people might not begin their search with a clear understanding of their goal, and they may even change their search task after visiting several pages. Also, users may be annoyed if forced to input a goal statement before every browsing session.

The Choo et al. [16, 17] experiment collected feedback from Web users by recording menu choices and web page information from the user's ordinary browsing. The goal of Choo et al.'s research is to build a model of web users' information seeking model, which separates web users into several groups. Users within each group share some specific browsing behaviors. Choo's results are very general in nature, thus it is not helpful to infer relevant pages based on any specific browsing session.

Information foraging theory [55] incorporates measures of semantic similarity in a model of user search behavior, and its key concept, information scent, characterizes how users evaluate the utility of hypermedia actions. Information scent can be used to characterize users' cost/benefit perceptions in making decisions, such as terminating the search of one

website in order to search for another site that contains more relevant information. Chi et al. [15] identifies "Information Need" based on the context of followed hyperlinks and information scent. Unfortunately, they provide no proof that the hyperlink context must correspond to the user's intention. For example, sometimes the context of the hyperlink only provides hints about the information need, rather than defining the need itself. e.g., "Publications" points to a page containing a list of papers. Pirolli and Fu [56] construct Information Need based on the SNIF-ACT model that is based on the concept of information scent. They use production rules to predict the user's information need and then enlarge it through a *spreading activation network* which is derived from Tipster corpus. These rules have been created manually and are hard to maintain. GCW employs machine learning algorithms to train browsing behavior models that can discover patterns that might not be found by human inspection.

2.6 Customer Behavior Models

Extracting a model of the behavior of Web user is a critical task for the E-commerce community, and much work has been done here. Lynch and Ariely [35] show that if customers can be presented with useful product-related information, their satisfaction with the merchandise they buy is increased. In other words, if we can infer the goal of web users, we can not only retrieve related information, but we can also help them effectively.

Bucklin and Sismeiro [10] developed a model to describe within-site browsing behavior: the visitor's decisions to stay or exit and the duration of each page view. But their research is at the individual level, and does not attempt to determine the kind of information each user wants. Park and Fader [49] incorporate observable outcomes and latent propensities across web sites to build a web browsing behavior model. Moe et al. [40] use a Bayesian Tree Model to present an online purchasing model. Johnson et al. [27] propose a model of the users search behavior, but only provide brief descriptions and no explicit information to infer what users want.

Customer Behavior models are therefore limited in that they do not take into account the content of pages viewed. The search behavior model is always proposed by experts after examining the recorded log data, and so some important aspects of the real Web user model may be lost.

GCW use a different source of information — i.e., browsing features — to train browsing behavior models, which can avoid the need to program complex systems by hand.

2.7 Evaluation

Although there is a great deal of research on generating recommendations for Web users, the evaluation of user models is an important, though often neglected area.

Kobsa and Fink [29] discuss the performance of a personalized web site by simulating the users' requests to test the performance and scalability of the components in user model (UM) servers. It is very helpful evaluate the workload of the UM server by simulating the real world applications, but it does not evaluate the actual recommendations that are generated (e.g., the likelihood that these recommendations are useful).

Weibelzahl and Weber [71] propose evaluating the accuracy of the predictive user model by comparing its assumptions to both an external test and the learners' true behavior. The limitation of this approach is that it can only be applied to simple user models, while excludes more complicated user models, such as models trained by machine learning.

Ortigosa and Carro [43, 44] describe how they acquire continuous evaluation in an adaptive-course system. The evaluation is inferred from the action of the Web users; however no evaluation was made on the suggested changes.

The recommended pages are expected to satisfy the user's current information need. The evaluation of the degree to which the suggested page can satisfy the user's information need should be acquired from actual users working on their day-to-day tasks, since the user is the best judge of the page's usefulness.

2.8 Summary

Table 2.1 compares the following types of recommender systems in terms of their key properties:

- COB: Co-occurrence Based [2, 3]
- CF: Collaborative Filtering [59]
- CB: Content-Based [6, 25, 5]
- HBM: Heuristic-Based Model [56, 31, 11]
- IC-Models: Browsing feature based Models.

The most common kinds of collaborative filtering are based on explicit feedback, like movie databases, where users explicitly rate the value of items. We would treat the explicit

	COB	CF	CB	HBM	IC-Models
Limited to Specific Site/Domain	Yes	Yes	Yes	No	No
Model Acquisition	Learning	Learning	Learning	Hand-coded	Learning
Annotation Required (training)	No	No	Yes	No	Yes
Annotation Required (performance time)	No	No	No	No	No
Training Pool	Population	Group	Individual	None	Population
Goal-Directed Recommendation	N/A	N/A	Yes	Yes	Yes
Using Sequential Information	No	No	No	Yes	Yes

Table 2.1: Techniques for Recommender Systems

recommendations as annotations. There are some systems that are based on weak informative choice signals from users (purchases at Amazon, views of a Web page, etc.). We view these systems as not requiring auxiliary user annotation.

While our IC-Models require the user's annotation to acquire training data, but the user does not need to annotate at performance time. However, the user may want to provide these annotations. Each of these user annotation requires only a few seconds. If this produces a significantly better model, which does a better job of identifying the user's interest, it can save the user considerably MORE time later, as it will direct the user to the IC-page more efficiently.

Chapter 3

Learning Web Browsing Behavior Models

To build a GCW recommender system for predicting relevant pages, I proposed a two-step method that first predicts the user's information need from current session, and then constructs an appropriate query to a search engine to return relevant pages. The first step, identifying information need, is the most important, yet difficult step.

The goal of identifying the user's current information need can be satisfied by learning the search strategy (*i.e.*, browsing behavior model) that is employed by actual users while they search for information on the Web. The model describes how users locate relevant information, which can be learned from previous Annotated Web Logs (AWLs) [1].

This chapter first introduces the concept of *Web Browsing Behavior*, and then demonstrates how browsing behavior can be used to identify "information need" with a set of words — words that may help identify a page that satisfy the information need.

3.1 Web Browsing Behavior

The concept Web browsing behavior model is driven by several observations concerning the nature of human-computer interaction while searching for information on the Web.

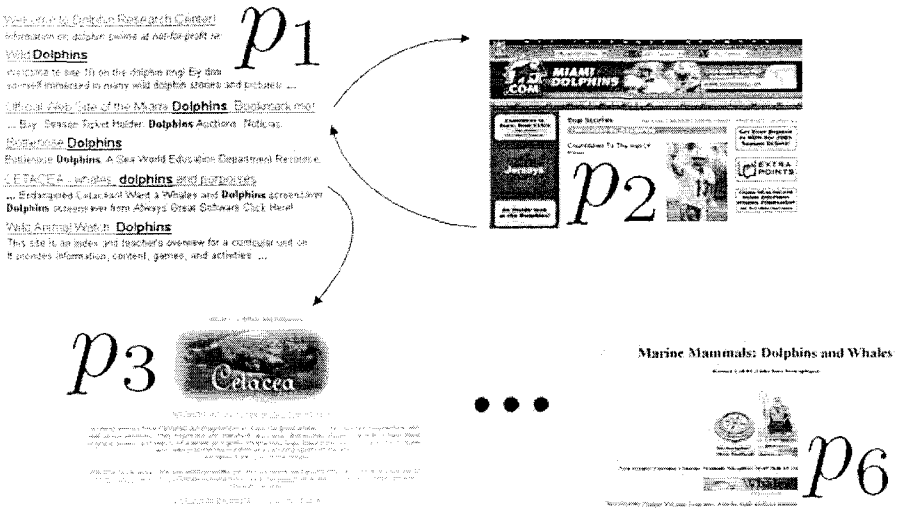

Figure 3.1: Web User's Browsing Behavior within a Session

Imagine that we are observing a user's browsing shown in Figure 3.1. The user starts from a page p_1 with links anchored with words like "Dolphins" and "Whales". Clicking on the "Dolphins" link takes the user to a page about the NFL Football team that goes by

[1] Each a sequence of webpages that a user visits, where each page is labelled with a bit that indicates whether this page is an IC-page.

this name. At this point the user employs the browser's "back up" button to return to the previous page p_1. Subsequently, the user follows another link on p_1 anchored with "dolphins" to reach a page p_3 on marine animals whose content includes words like "Whales" and "Dolphins". We observe that the user follows links anchored with "Whales" and "Dolphins" on p_3 to other pages with similar content.

What can we learn from this short example? We might conclude that the user's retreat from the page about the "NFL Football Dolphins" represents a negative judgment about the content of the page. Naturally, we do not expect single actions (e.g., *backup*) to unambiguously reveal the user's attitude toward page content, but by comparing the content of pages before and after user actions, we may be able to infer the specific content within pages that prompts the user's actions.

Following links anchored with words like "Whale" and "Ocean" suggests that these specific words might partially characterize the user's interests. The discarding of pages containing the word "football" suggests that "football" does not characterize the user's information needs.

Now imagine a second session in which the user is querying a search engine on "Belief Network". The user follows a link to reach a page on *"groups (networks) of people with a common religious (belief) views"*, but quickly discards this. We then observe the user is following links anchored with "Bayes nets" and "probabilistic inference". We might infer that the user wants to find information about "belief network". Although the keywords have changed from the marine animals example, the trajectory of web-pages followed by the user, the pattern of appearances of the keywords on this sequence of pages, and the actions the user chooses at each step are identical to the marine animal example.

The two examples above suggest that our intuitions about what the user was interested in are not based on specific words like "Dolphin" or "Belief Network", but rather on how these words appear in the browsing sequence and what actions the user applies to pages in this sequence, e.g., probably not interested in words appearing on pages "backed out of". Since we are interested in quickly determining a user's information need, we need a way of capturing these patterns. Once we identify these information-need-revealing patterns, we have no need to store the user's historical interest in various subjects such as "Dolphins" or "Belief Network". In fact, storing information about specific topics can be counter productive. For example, in a future session, this same user might be looking for the score of a recent Miami Dolphins game. Here the word "football" would be important, and of course his usage of the word "football" *in this session* would indicate this.

The key question, then, is which actions applied to what types of page sequences with

what types of patterns of recurring words reveal the content the user is really interested in? Intuitively, they will involve the contents of the pages, including the frequency of words followed up in subsequent pages, the applications of actions like the "back up" button, the font size of text, and many other features. But, we do not know the relative significance of these features, or even whether they lend positive or negative weight to the claim that a specific set of words being representative of the user's information need.

Some actions are relatively easy to interpret. For example, when a user clicks on a hyperlink, we assume this means there is a higher probability that some anchored words are relevant. Other actions require more subtle inferences. The key to my approach is explaining how the user's action choices signal the words that can be used to locate IC-pages. It can be viewed as a learning task, *i.e.*, training a model that can interpret the user's action on viewed information. The trained model that can be used to predict the pages that will satisfy a user's information need within the current browsing session.

We propose a system that can learn recommendation strategies from data to tune its internal parameters based on training examples. This has several advantages over non-learning approaches. First, learning algorithms can discover features in enormous data sets that might otherwise not be found by human inspection. Second, embedded machine learning algorithms can improve the performance of the programs by endowing them with adaptive, self-modifying behaviors. Finally, and most importantly, these methods avoid the need to program complex systems by hand, and thus allow us to develop working systems for tasks that are difficult, or impossible, to design and implement directly.

3.2 Learning Web Browsing Behavior

We define the pages in the current session except IC-page as *session pages* and the words that appear on them as *session words*. We use a slightly richer representation that allows us to express the degree to which a given session word represents the content of its respective page, determined by features (e.g., the frequency of the word on the page), and any special roles assigned to the word (e.g., appearances in titles, bold text or hyperlink anchors).

We also apply background knowledge to interpret user actions on a search page. We make use of the fact that links on the page are presented in a particular order. If the user skips over some items in an ordered list of options, we may conclude that she judged the content of these links to be less useful. If the user does not pursue links past a certain point in an ordered list, we might conclude that words appearing in these links are not useful, or that the information need has been either met or abandoned.

Together, the representativeness of a word on a session page, the session level features of

a word, the action signals associated with a word, and specific background knowledge can all be combined to provide evidence regarding a word's relevance. We can think of each of these sources of information for the word as a feature of the word. In order to convey the idea that these features depend partly on the user's actions, we call them *browsing features*.

To learn a browsing behavior model, we first extract training data from Annotated Web Logs (AWLs). Each of the web pages is first turned into bags of words. Then we compute "browsing features" of each word in the session, and label the word according to the model that we want to train. At this point we apply a machine learning algorithm to train the browsing behavior model that will later be used to recommend relevant pages.

It can be reasonably argued that my work is similar to filtering emails [30] or selecting interesting news articles [6]. However, while searching for information on WWW, information needs change quickly and are hard to anticipate. While GCW also needs the labelled web logs to learn the browsing behavior models, once this training has been completed, the user can use the system in any new Web environment without annotating any Web pages.

3.3 Browsing Features

On Web pages, much of the content of the page is communicated through the words on the page, thus the page content can be roughly approximated by examining the words that appear on it. This list, or as it is more commonly used, *bag of words*, is often sufficient for discriminating useful pages (*i.e.*, IC-pages) from unsuitable (non IC-pages).

Formally, we represent a browsing session $\mathcal{S}$ as a page-action sequence with length N

$$\mathcal{S} = [(p_1, a_1), (p_2, a_2), \ldots, (p_N, a_N)]$$

where p_i is page i in the session and a_i is the action applied to that page by the user. For example, the browsing session of Figure 3.1 is:

$$\begin{aligned}
&(p_1, \quad \text{"follow link"})\\
&(p_2, \quad \text{"back up"})\\
&(p_1, \quad \text{"follow link"})\\
&(p_3, \quad \text{"follow link"})\\
&\ldots\\
&(p_6, \quad \text{"exit"})
\end{aligned}$$

For each session word w, we define the role-action sequence

$$\mathcal{R}_w = [(R_1, a_1), \ldots, (R_i, a_i), \ldots, (R_{N-1}, a_{N-1})]$$

where R_i is a vector of roles played by word w in page i. For example, in Figure 3.1, "Dolphin" appears on page p_1 in the anchor text of hyperlinks and the main text, and also it appears in all the remaining pages in the session. The role-action sequence of "Dolphin" is thus:

$$\begin{aligned}
([hyperlink, plain]_{p_1}, &\quad \text{"follow link"})\\
([title, hyperlink, plain]_{p_2}, &\quad \text{"back up"})\\
([hyperlink, plain]_{p_1}, &\quad \text{"follow link"})\\
([plain]_{p_3}, &\quad \text{"follow link"})\\
&\ldots\\
([title, hyperlink, plain]_{p_6}, &\quad \text{"exit"})
\end{aligned}$$

Since "Whale" only appears in page p_1, p_3, and p_6, its role-action sequence is:

$$\begin{aligned}
([hyperlink, plain]_{p_1}, &\quad \text{"follow link"})\\
([hyperlink, plain]_{p_1}, &\quad \text{"follow link"})\\
([title, hyperlink, plain]_{p_3}, &\quad \text{"follow link"})\\
([hyperlink, plain]_{p_6}, &\quad \text{"exit"})
\end{aligned}$$

The browsing features of a word can be calculated directly from its role-action sequence. For instance we define a feature that gives the number of times a word w appears in a user's session. Let $|R_i|$ represent the number of roles a word plays on page i and $R_{w,i,j}$ be the j^{th} role of word w on page i. Then the session level feature *appears* is denoted as:

$$f_{\text{appear}}(\mathcal{R}_w) = \sum_{i=1}^{n-1} \sum_{j=1}^{|R_i|} \mathcal{R}_{w,i,j}$$

Additional functions include:

$$\begin{aligned}
f_0(\mathcal{R}_w) &\equiv \text{number of times } w \text{ appears in title}\\
f_1(\mathcal{R}_w) &\equiv \text{appearances of } w \text{ in a followed hyperlink}\\
&\ldots\\
f_k(\mathcal{R}_w) &\equiv \text{backed up from page with } w\\
&\ldots
\end{aligned}$$

We can think of the set of features as a vector function that produces the feature vector of a word for a given sequence, *i.e.*, browsing features of w:

$$F(\mathcal{R}_w) = [f_0(\mathcal{R}_w), f_1(\mathcal{R}_w), \ldots, f_k(\mathcal{R}_w)]$$

Let

$$\mathbb{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_i, \ldots\}$$

be the set of sessions collected for training. For each session $\mathcal{S}_i$, let W_i be the set of session words in the browsing session pages (*i.e.*, the union of all the words in the session pages minus the stop words). For each $w \in W_i$, we calculate a feature vector for w based on its role-action sequence in $\mathcal{S}_i$, and a label to show whether w is relevant to the current information need or not. An illustrative example appears in Table 3.1 based on the page sequence in Figure 3.1.

Word	#title	#hyperlink	...	#backup	...	relevant
dolphin	2	4		3		Yes
NFL	0	3		6		No
whale	1	4		0		Yes
football	0	4		5		No
⋮	⋮	⋮	⋮	⋮		

Table 3.1: Example Browsing Features

From Table 3.1, we might infer "browsing behavior" patterns in Figure 3.2. These patterns are the description of how the user searches for information, no matter which Web sites she visits. It is expected that once we can acquire such patterns we will be able to use them to predict what the user really wants based on her observable click stream.

IF w is in TITLE,
THEN w is relevant to the current information need

IF w is in a page's content, , (3.1)
and not in the title
and that page is "backwarded"
THEN w is a Non-relevant word

Figure 3.2: Sample Behavior Patterns

We estimate the user's information need from her current click-stream based on properties (*i.e.*, browsing features) of the *words* that appear there and the labels. Given the browsing features for a set of words in context of current session, we can use a browsing model (*i.e.*, classifier) to predict the probability (based on its browsing features) that any of these words will be *relevant*. We can then use these predicted relevant words to locate the relevant pages to satisfy the current (new) search task. The model is independent of users, specific words and specific Web pages, thus it can be used to identify relevant pages on any new Web environments.

3.4 Training Web Browsing Behavior Models

The first step for training Web browsing behavior models is to collect training data. To do so, we recruit a group of subjects to search for a wide variety of topics on the web and have these subjects explicitly annotated IC-pages. Figure 3.3 shows the learning process.

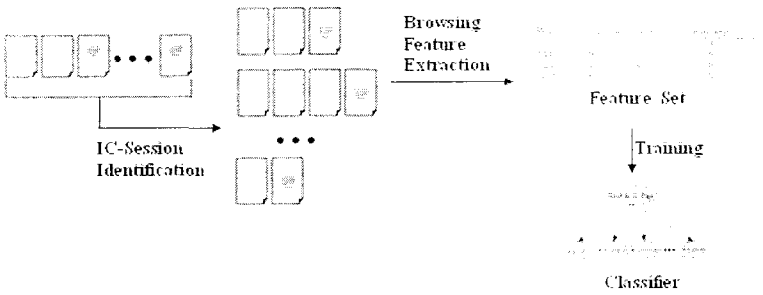

Figure 3.3: Learning Browsing Behavior Models from AWLs

First, we segment each user's complete click stream into a set of so-called browsing sessions — each of which concentrates on one search task and terminates in an IC-page. The technique for session identification and data cleaning will be discussed in Chapter 4.6.1.

Within each session, we extract all the words from the non-IC-pages, and compute various "browsing features" of each word. Then, we label each session word according to the model that we want to train. Next, we run a learning algorithm (e.g., C4.5) that uses these browsing features to learn a classifier (e.g., decision tree) that we can use to predict whether the session word is relevant or not.

The trained model can be applied to new browsing sessions to identify the user's current information need, which could then be used to recommend IC-pages for the user to satisfy her information need, and hopefully shorten her search.

Here, the model learned will not involve any specific words like "Dolphin" or "Belief Network", but instead, use the combinations of browsing features, roughly of the form shown in Figure 3.2. Once we have pattern rules that reveal the user's information needs, we no longer need the training data nor help from the users in marking pages that contain useful content. We can apply these generic patterns to any browsing session to reveal the user's information needs.

At performance time, the trained classifier takes, as input, a sequence of visited web pages $\vec{S} = \langle p_1, p_2, \ldots, p_n \rangle$ without annotation and returns a list of words that are likely to identify relevant pages (*i.e., relevant words*). In particular, it considers every word w that

appears in any p_i, then computes "browsing features" of this w as above. The system then classifies each w, determining the chance that a word with these browsing features will be relevant. Notice that this classifier bases its decisions on the browsing properties of a word rather than on the word itself. The performance system does NOT require that the user annotate the web pages: this was just done in the training phase. However, *if* the user is willing to do this labelling, we could hone the system to the nuances of the current user, and thereby obtain superior results compared to a generic system based on only the base population of users.

3.5 Evaluation

GCW is designed to find useful pages to help the user find useful information, so the best evaluation method is to ask for user feedback directly.

It is critical that we provide evaluation results to convince people that the browsing behavior models are truly useful. As one approach, we can collect the *AWLs* from the user studies, and test our model on these empirical data. Alternatively, we can acquire the evaluation by deploying these models in a real application, and ask for feedback from actual people using our tool. In our research, we followed both approaches for evaluation: in Chapter 4, we followed the first method, and in Chapter 5, we conducted user studies to evaluate the browsing behavior models by people in a real world environment. The results show the browsing behavior model can predict relevant pages very well, even though the predicted pages are previously unseen.

Chapter 4

User Study 1: Travel Planning

4.1 Introduction

The purpose of a GCW recommender system is to help users locate the information they want, by recommending IC-pages based on an observed page sequence. To this end we initially use a simple method to identify IC-pages using properties of URLs in the current annotated browsing session. Our ICPageWord model is used to identify the user's current information need, which can then be used to locate relevant pages. We have conducted user studies to collect AWLs, which are then used to train browsing behavior models, with promising results. This chapter first introduces the models that we want to train, then describes the user studies that we conducted, and finally summaries our empirical results.

4.1.1 IC URL Prediction as a Classification Task

Assume at time t a user has visited

$$\langle \langle p_1, \pm_1 \rangle, \langle p_2, \pm_2 \rangle, \langle p_3, \pm_3 \rangle, \ldots, \langle p_{t-1}, \pm_{t-1} \rangle, p_t \rangle$$

where each $\pm_i$ is "+" if this page p_i is explicitly labeled as an IC-page, and by default a page that is not labeled as IC-page is "−". We apply *Sequence Recognition* [67] on this information (augmented with other data, such as the domain type of each page) to determine whether p_t is an IC-page or not.

To address the challenge here, we consider an IC-page prediction as a classification task, and train a "IC-page classifier" that can take an annotated page sequence as input, and determine whether the final page is an IC-page or not.

The classifier we train is not based on each individual URL. That is, we are not attempting to predict whether one specific URL would be an IC-page all the time, but rather are predicting under what circumstance a page would be an IC-page. As an example of a possible rule, an URL will be an IC-page if 80% of the URLs with the same domain name in the current session have been identified as IC-pages. The model is expected to be more general than that built on specific URLs (*e.g.*, Association Rule [2]) since the latter can only be used on the Web site from which the training data are derived.

4.1.2 ICPageWord Prediction

Although IC URL prediction sounds promising, a shortcoming is that it requires the annotation information even at performance time, which is impractical in real application.

Similar to IC URL prediction, here we also apply machine learning algorithms to train such a browsing behavior model for identifying the current information need (i.e., ICPageWords) which can be used to locate IC-pages. This model is not based on a specific set

of pre-defined words, but rather on the user's observable behavior in response to the information within the pages visited. While there are important inter-individual differences in users' search behavior, we argue that our model — due to its high level of abstraction (that is, the way the word is used in the context of the session) — is able to capture user independent behavioral models that apply to broad classes of users and environments.

To train such a classifier, we first gather all the words from each annotated page sequence, and then compute browsing features for each word based on how that word appears within the sequence. The label for each word is whether it appears in the IC-page or not.

There are some systems that define the user's *information need* as a learned combination of a set of (possibly pre-defined) words. For example, the model [19] trains a NaïveBayes classifier to label each Web page as relevant or not, using the occurrences of a set of predefined words as the features [6]. In our work, however, we attempt to label each individual word with a measure of its likelihood to be relevant based on browsing features. (Fig. 4.1 contrasts these two approaches.) We can use this information to form a query to a search engine as a list of the high-scored words. We therefore view this set of word-score pairs as encoding the user's current information need.

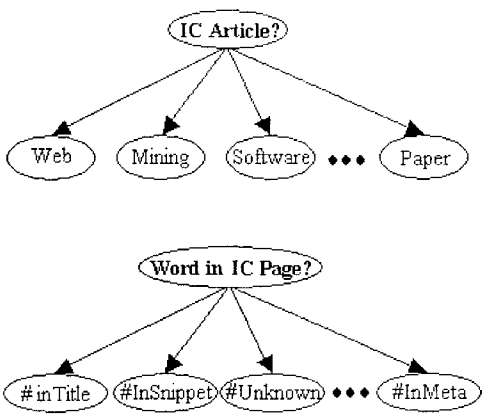

Figure 4.1: NaïveBayes Models for (a) IC-page Identification based on Presence of Specific Words; (b) ICPageWord Identification

The ICPageWord classifier depends on characteristics of words, not on the specific words themselves. For example, it might be the case that any word that appears once in a snippet that is followed, and twice in a page title, will probably be an "ICPageWord" (see Fig. 3.1). This means we can train our "ICPageWord classifier" in one set of sessions, and then use it to classify a completely different set of words associated with a completely different session.

4.2 Travel Planning Task

We conducted a series of laboratory studies to generate sets of data (i.e., *A*WLs), which we used both to train our models and then to test the performance of the trained models, to determine their ability to predict a user's current information need.

A total of 144 undergraduate business students participated in the Travel study for partial course credit and a lottery incentive [1]. Study sessions were administered in a supervised computer laboratory in groups of approximately 25 subjects.

The overall task consisted of selecting a new vacation spot (i.e., one that the subject had not visited in the past) to which the subject would like to travel. Each participant was asked to perform the following specific tasks:

1. *Identify 3 novel vacation destinations (i.e., places you have never visited)*
2. *Plan a* detailed *vacation to each destination, specifying specific travel dates, flight numbers, accommodation (hotels, campsite, ...), activities, etc.*

They were allotted approximately 45 minutes, and given access to our augmented browsing tool — Annotation Internet Explorer (AIE; see Chapter 4.3), which recorded their specific web-logs, and required them to provide the "importance" (i.e., IC-page) annotation. The participants also had to produce a short report summarizing their vacation plans, and citing the specific important webpages that were involved in these decisions. Here AIE made it easy to remember and insert these citations (see Figures 4.2 and 4.4).

We chose this specific task as

- It represents a fairly standard way of using the web
- It is goal directed (in contrast to simply asking the participants to "meander about the web")
- A diverse set of pages may be relevant, *e.g.*, plane schedules, travel brochures, recent news (such as terrorist attacks). The contents of many of the travel Web sites do not change frequently.
- It is easy to motivate students to do this task.
- The task is fairly well-defined and delimited: the references in the report help identify which pages qualify and which do not.

[1] This included both male and female students; our use of "she" in referring to a participant is merely for notational simplicity.

Throughout their search, subjects were asked to identify any IC-pages that they considered to be of use with respect to their task of making a vacation choice by clicking on a special button in their Web browser. Once they had selected their preferred vacation, participants recorded specific aspects of their choice on a form. The subjects' Web navigation (URLs and time stamps) during the search task were tracked electronically.

Usable data were obtained from 129 participants. Collectively the 129 participants in the study requested 15,105 pages, and labeled 1,887 IC-pages, which corresponds to 14.63 IC-pages per participant. This involved 5,995 distinct URLs, meaning each URL was requested 2.52 times on average. Table 4.1 is a histograph showing how often each page was visited.

Table 4.1: Number of Requests vs Percentage of the URLs

Number of Request(s)	Percentage of the URLs
1	58.93%
2	23.46%
3	7.63%
4	4.08%
5	1.85%
6	1.16%
7	0.88%
8	0.40%
9	0.40%
10	0.18%
...	...

Note that 82.39% of the URLs were visited only one or two times. Clearly very few URLs had strong support in this dataset, which would make it very difficult to build a recommendation system based on page correlation and frequency.

4.3 AIE: Annotation Internet Explorer

I developed an enhanced version of INTERNET EXPLORER, called AIE (shown in Figure 4.2) to collect the relevant information for the travel planning user study.

AIE incorporates several relevant extensions to IE, as seen in the toolbar across the top of Figure 4.2. First, the user can declare the current page to be an IC-page by clicking the *Important* button on the top bar. When doing this, AIE will pop-up a new window that shows this URL, and two edit fields that allow user input: a mandatory field requiring the user to enter an alias for this page (*e.g.*, "AirCanada Edmonton-Beijing ticket prices") and an optional field for writing a short description of why this page was important (*e.g.*, "provides the cost of the plane tickets"). See Figure 4.3.

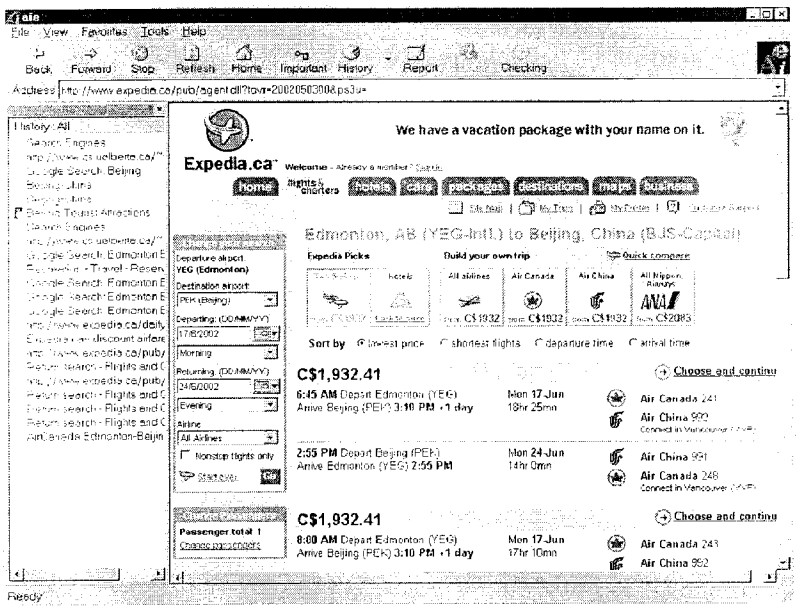

Figure 4.2: AIE Browser

The *History* button on the toolbar brings up the side-panel (Figure 4.2), which shows the set of all pages seen so far, with a flag showing which pages the user tagged as an IC-page.[2] The user can click on one of these to return to that important page; she can also reset its "importance" designation.

The *Report* button will switch the browse view to the report editor, which she can use to enter her report (Figure 4.4). Here, each subject has access to the IC-pages labeled during browsing, which she can use in producing her report. In fact, the participant can only use such pages in her report. She does have the option of returning to the first "browse" mode, and adding new pages to the list of IC-pages. She can also examine *all* pages visited earlier, and re-assign the pages (i.e., take a page previously considered unimportant, re-declare it to be important, and then use that page in her report.).

After completing the report, the user can then submit the entire session using the *Submit* button. This records the entire sequence of web-sites visited, together with the user's IC-page annotations, as well as other information, such as the time-stamp for each page. Figure 4.5 shows part of the annotated web logs. The URL in italic is the IC-page annotated by the subject, followed by the IC-page label.

In addition to collecting these sessions, we downloaded copy of every page visited by

[2]The user can view only the IC-pages.

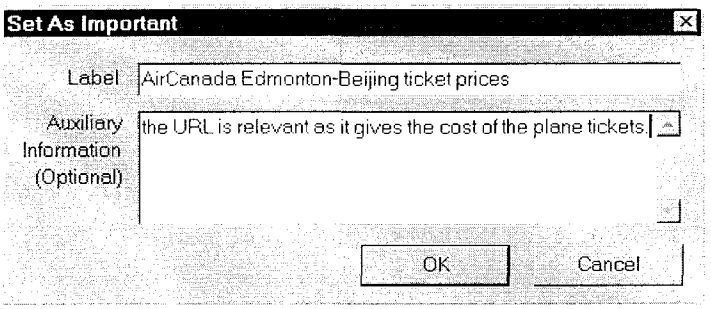

Figure 4.3: AIE PopUp Window to Declare an "Important" Page

any of the participants. Furthermore, we drill down the Web site from the visited page by breadth first search up to 5 levels. The 9.1GB collection of Web pages enable us to later determine something about the structure of the Web sites at the time they were visited.

4.4 Data Preprocessing

We use our own web log format, which records the URL and time stamp of each page, along with the one-bit IC tag. Unfortunately some of the pages in AWLs are just advertisements. As these ads do not contribute to the participant's information needs, leaving them in the training data might confuse the learner. We therefore assembled a list of advertisement domain names, such as: `ads.orbitz.com`, `ads.realcities.com`, etc. We compare each URL's domain name with the ad server list and ignore a URL if it is in the list.

After filtering out the advertisements, we build "page views". This is challenging because of the widely used frame pages. When a frame page is being loaded, all of its child pages will be requested by the browser automatically; and thus, instead of recording only the frame page in the log file, all of its child pages will be recorded as well. This is problematic when the participant browses within a frame page. For example, in Figure 3.1, page p_2 is a frame page, containing pages (p_2^1, p_2^2, p_2^3), thus in the log file, the page sequence is $p_1 \rightarrow p_2 \rightarrow p_2^1 \rightarrow p_2^2 \rightarrow p_2^3$. However, the second page view should contain p_2^1, p_2^2, p_2^3, and not 3 different pages. We therefore construct the real page views that the subject has seen when she was browsing the Internet. Here it should be $p_1 \rightarrow p_2^{(p_2^1, p_2^2, p_2^3)}$, as shown in Figure 4.6.

As only 10% of the pages are important, there is a trivial way to obtain high accuracy: just return "not important" to each instance. Of course, this will not serve our needs as it is critical to know which pages are important. To address this problem of "imbalanced data" [23], we have tried both down-sampling [33] and over-sampling [24] to build training

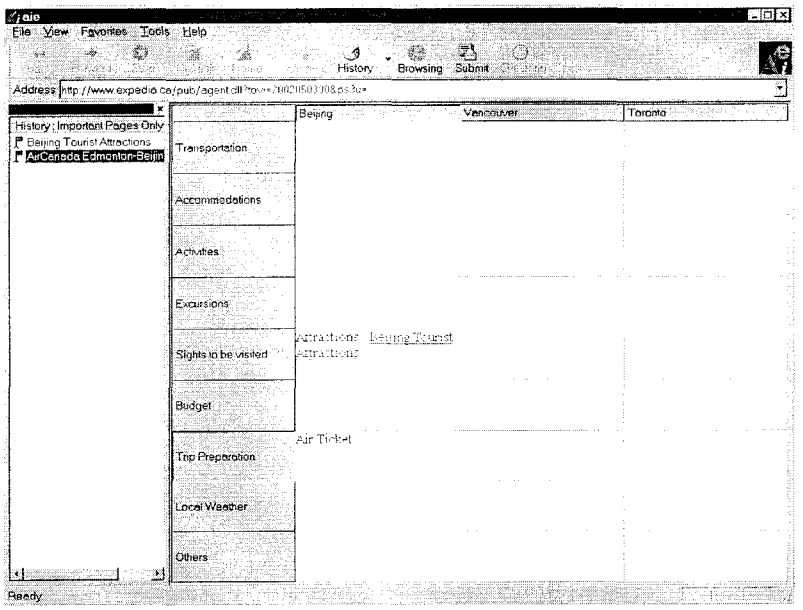

Figure 4.4: AIE Report

```
....
2001 11 26 13 17 1 0
http://www.google.com/search?q=weather+vancouver

2001 11 26 13 17 15 1
http://www.canoe.ca/Weather/CityVancouverBC.html
weather in Vancouver

2001 11 26 13 21 5 0
http://www.google.com/search?q=weather+vancouver

2001 11 26 13 21 21 0
http://www.google.com/search?hl=en&q=hongkong

...
```

Figure 4.5: Sample Annotated Web Log

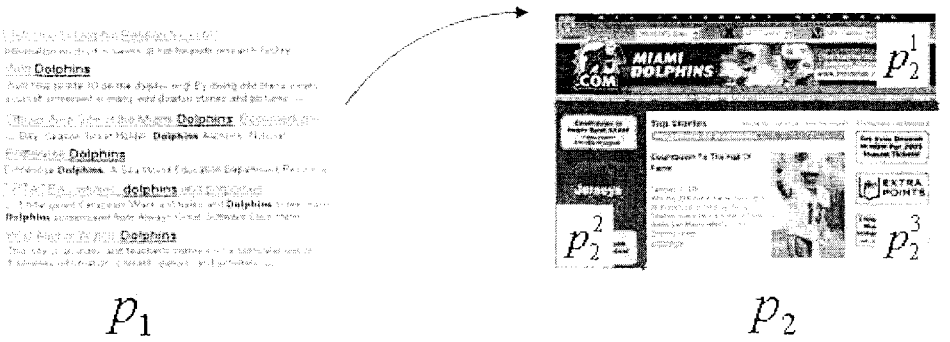

Figure 4.6: Page View with Frames

data as follows, in which S_{IC} is the number of all ICPageWords and S_{Non-IC} is the number of all Non-ICPageWords.

Down-Sampling Form a $2 \times S_{IC}$ training set by using all ICPageWords and randomly selecting (without replacement) S_{IC} Non-ICPageWords.

Over-Sampling Form a $2 \times S_{Non-IC}$ training set by using all Non-ICPageWords and randomly selecting (with replacement) S_{Non-IC} ICPageWords. (Notice some ICPageWords may appear several times in a single training sample.).

While we wanted to include "time" information (i.e., how long the user spent on each page), and we did record this information, but we were unable to use it. Before we ran the experiment, we incorrectly assumed the subjects would browse the Internet and mark important pages before switching to report writing. In the lab, however we found that many users switched modes (to "Report mode") upon finding each important page. This means that much of the time between requesting an important page and the next page, was spent in report writing, which skews the time statistics.

Moreover, Kelly et al. [28] report that they cannot find any direct relationship between the time spending on each page and its usefulness; this also support the policy for ignoring time information in our research.

4.5 IC URL Identification

After data preprocessing, we extract features for each URL, such as how often the URLs from the same domain have been identified as IC-pages. Section 4.5.1 gives more details of these attributes. Each URL was annotated as "Important" or not, thus we assign class "+"

to the "Important", while "-" for non-important.

While training, we do not give the specific URLs to the learner. Thus the learner is expected to acquire the patterns that determine what characteristics URL that are "Important".

4.5.1 URL Attributes Used

We extract the following features of URLs from the click stream, and define a site-session as the click stream within a single Web domain (i.e., if the subject enters a new web site, a new site-session begins.).

1. URL Properties

   URL Type

   wrong (*e.g.*, "404"), search (*e.g.*, `GOOGLE`), dynamic (*e.g.*, produced by CGI script), static, (*e.g.*, typical *.html page), misc (*e.g.*, pointer to jpg, mpg, or mov file, or whatever)

   DomainType

   wrong (404), edu, com, net, org, gov, misc

   Depth

   Number of "/"s in the URL

2. User Click Stream

   FollowSearchEngine

   Does this page follow immediately from some search engine (*e.g.*, `GOOGLE`)?

   isLastEntry

   Whether this page is the last one in the site-session.

   inTotalNumberofPage

   The number of pages that have been visited within this site-session, until now.

   inTotalNumberofImportantPage

   The number of pages, within this site-session, that have been labeled as important.

   inLastImportant

   The number of pages that have been visited since last important page within this site-session. If no previous important pages, just use the number of pages visited in this site-session until now.

   TotalNumberOfPages

   The number of pages have been visited so far.

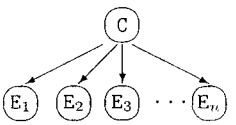

Figure 4.7: NaïveBayes Structure

TotalNumberOfImportantPages

The number of important pages that have been labeled so far.

LastImportant

The number of pages that have been visited since last important page. If no such previous important page, just use the number of pages visited until now.

PercentageDomain

The percentage of the pages that have the same domain as the current page since the user started.

PercentageImportant

The percentage of the important pages that have the same domain as the current page so far.

PercentageSameDomainImportant

For all the URLs under the same domain, the percentage of the important pages.

4.5.2 Empirical Results

After the data preparation, we run several classification algorithms on the data set after over-sampling, producing the following [79].

- Decision tree, using C4.5 (see [58], `http://www.cse.unsw.edu.au/~quinlan/`)
- NaïveBayes — a simple belief net structure of the form shown in Figure 4.7 (where C is the class, here "Importance", and the E_is are the various attributes shown in Section 4.5.1) which claims that the attributes are independent of one another, conditioned on the class label [19].
- Boosted NaïveBayes: In general, "boosting" is an approach to improve the result of a learning algorithm A, by using A to learn a set of classifiers over slightly different datasamples (which differ by reweighting the elements in training set) [64]. Here, we boosted the NaïveBayes learner.

Note that we were able to run the above classification algorithms using the WEKA system, which is a large collection of learning algorithms. (See [72], `http://www.cs.waikato.ac.nz/~ml/weka`.)

Table 4.2: Results for One Run

		Actual label	
		Important	Unimportant
Predicted label	Important	81	48
	UnImportant	19	52

In all cases, we used the default setting, and ran 10-fold cross validation. Table 4.2 gives the confusion matrix, based on one split, using NaïveBayes. Here, we define "Precision" and "Recall" for Important pages prediction as follows.

$$Precision_{Important} = \frac{ActualImportantPredicted}{PredictedAsImportant} = \frac{81}{81+48} = 62.8\%$$

$$Recall_{Important} = \frac{ActualImportantPredicted}{AllActualImportant} = \frac{81}{81+19} = 81.0\%$$

$ActualImportantPredicted$ are those important pages that are predicted as important as well (shown in Fig. 4.8).

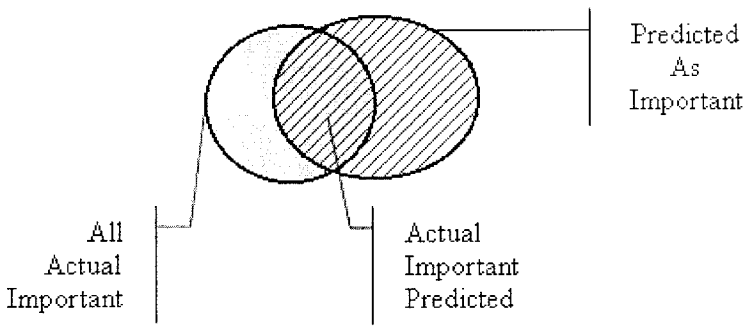

Figure 4.8: Precision and Recall for Important Page Prediction

We can similarly define Precision and Recall for Unimportant pages:

$$Precision_{Unimportant} = \frac{ActualUnimportantPredicted}{PredictedAsUnimportant}$$

$$Recall_{Unimportant} = \frac{ActualUnimportantPredicted}{AllActualUnimportant}$$

The results, over all 10 CV folds appear in Table 4.3 in the form mean±standard-deviation.

Notice that Boosted NaïveBayes has the best "worst-case" over these 4 values, averaging around 65%. Although we can get fairly high accuracy for the IC URL prediction, the problem is that the predictions require annotation information even at performance time, which is not applicable.

Table 4.3: Empirical Results

	Important		UnImportant	
	Precision	Recall	Precision	Recall
C4.5	0.712 ±0.063	0.27 ±0.05	0.5486 ±0.02	0.89 ±0.02
NaïveBayes	0.594 ±0.035	0.82 ±0.03	0.7075 ±0.06	0.46 ±0.12
Boosted NaïveBayes	0.669 ±0.048	0.70 ±0.04	0.6861 ±0.041	0.65 ±0.07

4.6 ICPageWord Identification

For training the ICPageWord model, we first identify the IC-session from the pre-processed log data, which differs from those introduced in [18].

4.6.1 IC-session Identification

Each user might pursue several different information needs as she is browsing. To identify and distinguish these needs, we separate the pages into a sequence of "IC-sessions", where each such IC-session pertains to a single information need. In general, each IC-session is a consecutive sequence of pages that ends with an IC-page.

Chen et al. [12, 13] terminated each session on reaching a Maximum Forward Reference (MFR) when the user does not follow any outlinks from a page. Of course, these final MRF pages need not correspond to IC-pages. Cooley et al. [18] used time-outs to identify sessions: if the time between consecutive page requests is greater than a threshold, they assume that a new session has started. While the fact that a user remained at a single page may suggest that it could be an IC-page, this is not the only explanation for remaining on a page. Note that neither set of authors claims that these final pages addressed the user's information need, and so they provide no evidence that these pages were IC-pages.

In our case, since we focus on goal-directed browsing, we terminate a session on reaching an IC-page. However, it is not clear that the next session should begin on the subsequent page. For example, imagine in Figure 3.1 reaching page p_1 after visiting a sequence of pages $p_a \rightarrow p_b \rightarrow p_c \rightarrow p_1$, and assume both p_2 and p_6 are IC-pages. Here, each IC-session should contain the sequence before p_1 since it also contributes to locating each of the IC-pages. In other words, in this instance we would produce two IC-sessions :

$$p_a \rightarrow p_b \rightarrow p_c \rightarrow p_1 \rightarrow p_2$$

$$p_a \rightarrow p_b \rightarrow p_c \rightarrow p_1 \rightarrow p_3 \rightarrow \ldots \rightarrow p_6$$

To identify meaningful IC-sessions, we used some heuristics. For example if the page after an IC-page is a new search query, then we assume that starts a new session, since it is very common that upon completion of one task, users go to a search engine to begin the next task. Figure 4.9 summarizes our IC-session Identification algorithm.

Algorithm ICSI:(page sequence $S = \langle p_1, p_2, \ldots, p_N \rangle$):
outputs Sequence of IC-sessions
F: Boolean: *% true iff current page is immediately after an IC-page*
L: Queue: *% stores the current session*
BEGIN
Set L := empty; F := false
For i=1..N do
If p_i is an IC-page then
If L is not empty, Output L
F := true;
Else
If(F) then
If p_i is a search query page then Empty(L);
If p_i is in L then Pop off L every page after this first p_i;
F := false
Append p_i to L
If L is not empty, Output L.
END

Figure 4.9: ICSI Algorithm

4.6.2 Browsing Feature Extraction

We consider all words that appear in all pages in a browsing session, removing stop words and stemming, using standard algorithms [57], and then we compute 25 browsing features for each word within the IC-session. In all cases, if the URL refers to a frame page, we calculate all the following measures based on the page view.

Search Query Category

As our data set includes many requests to search engines, we include several attributes that relate to the words in the search result pages.

Each search engine will generate a list of results according to the query, but the content of each result may differ for different search engines. We consider only information produced by *every* major search engine: *viz.*, the title and the snippet. For example, in Figure 4.10, the title of the first result is "Welcome to Dolphin Research Center!", and its snippet is "Information on dolphin swims at not-for-profit research facility:".

We tag each title-snippet pair in each search result page as one of: *Skipped*, *Chosen*, or *Untouched*. If the user follows a link, the words in its title and snippet will be considered "*Chosen*". The words that appear around the links that the user did not follow, but before

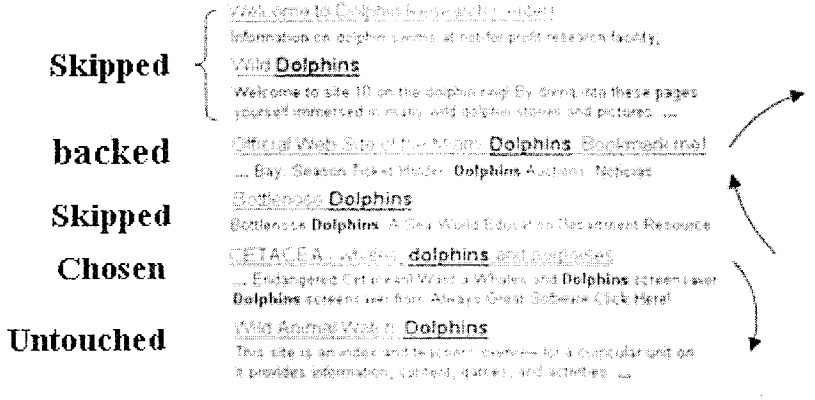

Figure 4.10: Title-Snippet State in the Search Result Page

the last chosen link, will be deemed "*Skipped*",[3] and all results after the last chosen link in the list will be "*Untouched*". Figure 4.10 shows 3 "Skipped", 2 "Chosen", and 1 "Untouched" results. The third entry "Miami Dolphins" is the first one followed; the user later clicks back to the search page, and chooses the fifth entry, "CETACEA - whales, dolphins and porpoises". Also, for pages in general, we say a hyperlink is *backed* if the user followed that link to another page, but went back later. A page is *backward* if that page has been visited before; otherwise we say a page is *forward*.

The actual features used for each word w appear below. Each is with respect to a single IC-session. Notice that most have numeric scores and many are simple integers — *e.g.*, how many times w is in some specified category. Please refer to Appendix B for further details.

isKeywordCnt
skippedTitleCnt
skippedSnippetCnt
chosenTitleCnt
chosenSnippetCnt
untouchedTitleCnt
untouchedSnippetCnt
unknownCnt
bkTitleCnt
bkSnippetCnt

Sequential Attributes

Each of the sequential attributes is extracted from set of the pages in an IC-session, excluding only the search result pages and the last IC-page.

[3]These are the links that the user probably saw, but actively chose not to follow.

The tf-idf scheme has been widely used in information retrieval, and it is also used for calculating our sequential features. However, Web pages have peculiar features such as markup tags and hyperlink structures. Thus, we compute the weight of each word in a page based on its occurrences contained in some informative HTML tags. We pick out the following HTML tags, as they appeared quite often in our AWLs, and use them to compute the "weight" of each word w in each page.

$$weight(w) = \sum_i Cnt_i(w) \times v_i$$

where $Cnt_i(w)$ denotes the number of occurrences of w contained in the i^{th} "HTML Tag" [69]; and v_i is the weight associated with this context, shown as

h1	10	h6	5	strong	15	b	15
h2	9	a	50	big	20	u	10
h3	8	title	20	em	15	blink	20
h4	7	cite	10	i	15	s	5
h5	6						

(4.1)

For each word w in an IC-session, we compute each of the following attributes as described in Appendix B. At training time, we also indicate whether w appears in the IC-page or not.

ratioWordAppearance	*avWeight*
varWeight	*trendWeight*
ratioLinkFollow	*ratioFollow*
ratioLinkBack	*ratioBackward*
avWeightBackward	*varWeightBackward*
ratioForward	*avWeightForward*
varWeightForward	*ratioInTitle*
ratioInvisible	*bkSnippetCnt*
bkTitleCnt	

We compute the browsing features of all the words along all pages of the entire IC-session, with the goal of anticipating what the user is seeking. (Hence, this differs from simply summarizing a single page [76].) Recall that when we train the classifier, we do not use the words themselves, but instead we use these attribute values, and use a label whether the word appears in the IC-page (i.e., the ICPageWord tag).

4.6.3 Empirical Results

To investigate the effectiveness of ICPageWord approach, we conducted a user study. The material in this chapter extends [77, 79, 78].

After preparing the data, we first use Weka [72] to produce a NaïveBayes (NB) classifier [19]. For each IC-session, let w_{seq} denotes all the words in the sequence except the last

page (which is an IC-page), and w_{dest} denotes the words in that final IC-page. We focus on only those sessions with $w_{dest} \subseteq w_{seq}$. We ran our test on $(8 \times 20) + 1$ subject groups, where each group involved $U \in \{1, 2, \ldots, 8\}$ subjects. For $U = 1$, we took each individual as a 1-user group; and for the other $U \in \{2, 3, \ldots, 8\}$, we randomly selected 20 different groups from the set of participants. Note that we allowed overlap among these 20 groups. For each group, we built 10-fold training/testing datasets, and computed the median value of these 10 results as the final score for this group. (We report medians because they are less sensitive to outliers than means.) To generate the training and testing data, we used the oversampling technique described earlier.

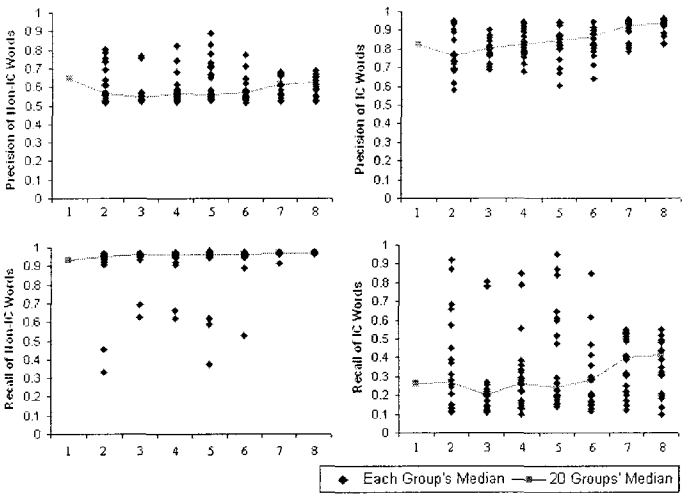

Figure 4.11: ICPageWord Prediction: Testing Result

Here, we found an average accuracy of around 78.3%; the precision and recall results appear in Figure 4.11. Even though the average recall of ICPageWords is only about 45%, we anticipate this will be sufficient to to find IC-pages, which of course is our ultimate goal. Given the high precision of our ICPageWord prediction, even with recall around 45%, we can anticipate finding tens of words that will surely be in the IC-page. Since the predicted ICPageWords are exclusive of stop words, we anticipate they will be quite relevant to the IC-page's content. We therefore suspect that a query with these relevant words will help retrieve the relevant IC-page, see Chapter 5.

In addition to NaïveBayes(NB) classifier, we also train a decision tree, Ripper [62], Support Vector Machines (SVM) [68], and Bayesian Network for identifying ICPageWords. To deal with those continuous attributes, we use estimation [20] instead of discretization [26] since the former typically produces higher accuracy. We find down-sampling can acquire

better accuracy than over-sampling, thus we use down-sampling in our testing experiments. The results appear in Table 4.4.

Table 4.4: Accuracy of Prediction

	Accuracy
C4.5	87.4%
Ripper	73.7%
SVM	75.4%
Bayesian Network	72.3%

The accuracy of C4.5, at about 87.4%, is much better than that of other classifiers. We conjecture two possible reasons for C4.5's superior performance. First, C4.5 uses *local* discretization of integer attributes, whereas NaïveBayes uses a global approach. Second, C4.5 does implicit selection of relevant features through its tree splitting and pruning operations.

Single-User Case

The single-user case ($U = 1$) is perhaps the most relevant, as it shows how well our system, trained on a single user, will perform for that user. Here, for each user, we run 10-fold training/testing.

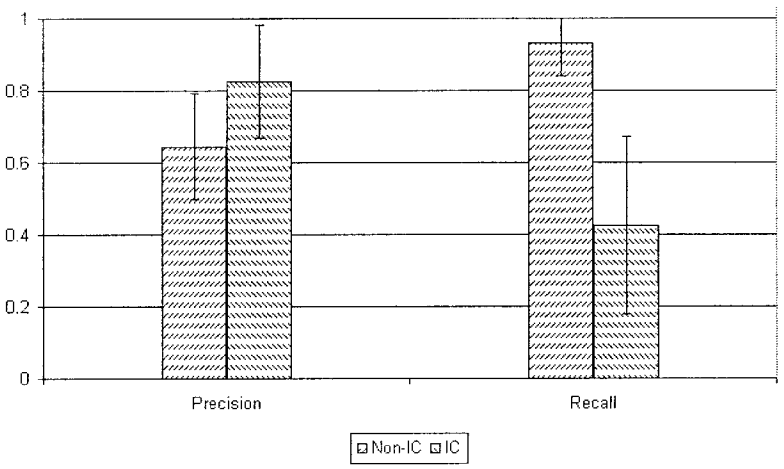

Figure 4.12: Prediction Result for Individual Users

Figure 4.12 shows the average values for all 4 cases (Non-IC vs IC; and Precision vs Recall) on all testing sessions. The precision of ICPageWord prediction is higher than that of Non-ICPageWord. This is encouraging, since ICPageWords will be used for producing IC-pages in our recommender system. The recall of Non-ICPageWord is higher than that

of ICPageWord, possibly as the majority of all the words in the observed page sequence are Non-ICPageWords. However, the recall of Non-ICPageWord is less useful for IC-page prediction.

The precision of ICPageWord prediction is above 80% for 78% users, indicating that many individual users are very predictable. This suggests that there should exist a general model of web user's information search — one not based on a particular website or a specific set of words, but a general model that describes how users find useful information on the Web.

Leave-One-Out Testing

We are also interested in the performance of the trained model for individual users. Here, we are interested in the recall of ICPageWord prediction, since the more ICPageWords that can be recalled, the more accurately can *information need* be identified. In addition to the single user training/testing in Chapter 4.6.3 (Self-Testing), we run another two types of leave-one-out testing as follows:

Leave-One-Out Testing: We use all data of one user as testing data, and the remaining users' data for training. We then compute the average recall of the prediction. This is the case when training a population model, then testing on a new user.

Group-Based Leave-One-Out Testing: We suspect that for each specific user u, there might exist a user group G_u that shares the same browsing behavior model. The purpose of Group-Based Leave-One-Out testing is to acquire the highest performance for each user, but it is very time-consuming to try all subsets of the users to find such a group for each user. In the group-based leave-one-out testing, for each user u, we randomly select M groups from the remaining users, where each group contains U ($U = 10$) users. For each group, we use all involved users' data as training data, for training NaïveBayes. Then, for each of u's sessions, we choose the highest recall from the Ms as the final result. Here, we assume that the user group that results in the highest recall is the group that the user should belong to. Fig. 4.13 shows the average recall of ICPageWord prediction for each single user with $M = 20, U = 10$.

Figure 4.13 shows the average recall of leave-one-out testing. We run the Wilcoxon test on each pair of the three methods, and the results are presented in Table 4.5.

From Table 4.5, we conclude that both Group-Based testing and Self-Testing work better than *Leave-One-Out* testing, but there is no significant difference between *Group-Based* and *Self-Testing*. We can see that Group-Based testing gets the best predicting results on

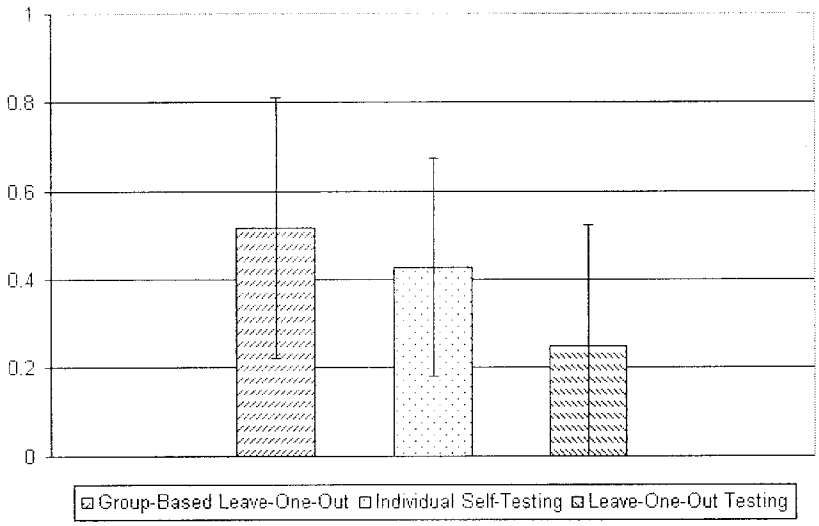

Figure 4.13: Average Leave-One-Out Testing Result for Individual User

	p
Group-Based $\geq$ Leave-One-Out	0.0186
Self-Testing $\geq$ Leave-One-Out	0.0029
Group-Based $\geq$ Self-Testing	0.097

Table 4.5: Wilcoxon's Test on Three Individual-Oriented Testing Methods

average. This is what we expected: the generic model learned from all populations cannot compete with the model learned from a specific user group. Therefore, we can produce more effective recommendations for any user if we can identify the user group that she belongs to.

We also observe that the single-user training works better than the group-based method. This also motivates us to learn personalized models in our future work.

4.6.4 General Evaluation Method

Clearly a good recommendation system should predict all-and-only the IC-pages. It would also be useful to predict these pages early. In Fig. 3.1 it is better to recommend an IC-page after the user has traversed only $\langle p_1, p_2 \rangle$, rather than wait until the user has visited all of $\langle p_1, p_2, \ldots, p_6 \rangle$. This would save the user visiting the 4 intervening pages. We therefore define an evaluation method based on these two objectives.

For each IC-session $S = \langle p_1, p_2, \ldots, p_N \rangle$ of length N (where p_N is an IC-page), there are $N-1$ subsessions, where each subsession $S_\ell = \langle p_1, p_2, \ldots, p_\ell \rangle$ is the first consecutive

ℓ pages. for $1 \leq \ell < N$. We will call the recommendation model on each subsession S_ℓ to generate a proposed set of ICPageWords. $WP(S_\ell)$. $W(p_i)$ denotes all the non-stop words on page p_i.

We compute precision and recall for the predicted ICPageWords, and then calculate the F-Measure [61].

$$\text{ICprecision}(S, \ell) = \frac{|WP(S_\ell) \bigcap W(p_N)|}{|WP(S_\ell)|}$$

$$\text{ICrecall}(S, \ell) = \frac{|WP(S_\ell) \bigcap W(p_N)|}{|W(p_N)|}$$

$$\text{F}(S, \ell) = \frac{2 \times \text{ICprecision}(S, \ell) \times \text{ICrecall}(S, \ell)}{\text{ICprecision}(S, \ell) + \text{ICrecall}(S, \ell)}$$

In order to get a better idea of how far ahead the model is predicting, we define h to be the *horizon* of prediction, which is the number of pages from the prefix to the actual IC-page (i.e., $h = N - \ell$, see Figure 4.14.). Using *horizon*, we finally define

$$score(S, \ell) = \begin{cases} penalty & \text{if } WP(S_\ell) \bigcap W(p_N) = \emptyset \\ \text{F}(S, \ell) \times h & \text{otherwise.} \end{cases}$$

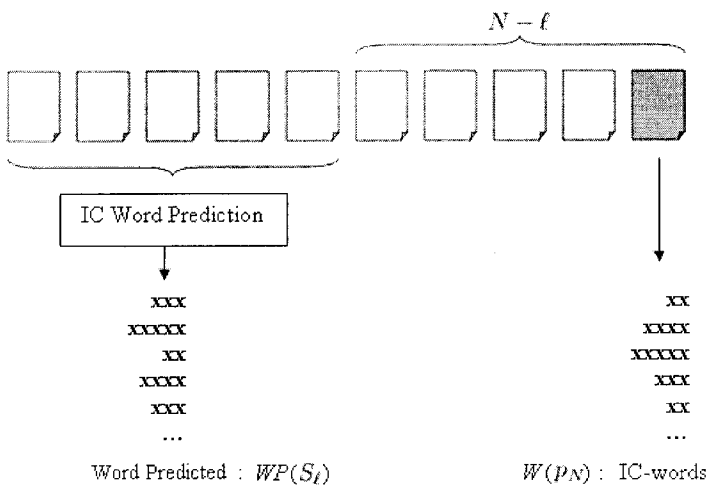

Figure 4.14: Evaluation Specification

The *penalty* $\in \Re^-$ term penalizes the system for being non-responsive. Here we use -0.05. Notice this $score(\cdot, \cdot)$ increases the earlier the system can make a prediction, provided that prediction is accurate (based on the F-measure). We divide by the number of predicted words $|WP(S_\ell)|$ to discourage the system from simply suggesting everything.

For each IC-session S_ℓ, let

$$\text{coverage}(S_\ell) \quad = \quad \frac{|W(p_N) \bigcap W(S_\ell)|}{|W(p_N)|}$$

be the overlap between the words in the IC-page and the words in S_ℓ (i.e., $W(S_\ell)$). *coverage* = 1 corresponds to the $W(p_N) \subseteq W(S_\ell)$ condition which means all the ICPageWords can be found somewhere in the session.

We randomly select 90% of the IC-sessions as training data, and use the others for testing. For each testing IC-session, we calculate the average *score* over all subsessions, provided there were any recommendations. (That is, we provide no recommendation if our system finds no word qualifies as an ICPageWord.) We then compute the average score for all testing IC-sessions as the final score for this trial. Figure 4.15 graphs this information, as a function of the coverage. We find that in most cases, where the coverage increases, $|W(U_\ell)|$ grows very quickly, which is the main reason that the score worsens.

We compare our method with two simple methods: let the ICPageWords be (1) *All*: all words in S_ℓ, or (2) *Features-Only*: all feature words in S_ℓ, which are those words enclosed by some specific HTML tags, such as "a", "title", "b", "h1", etc. Figure 4.15 shows that our approach did significantly better than these two methods.

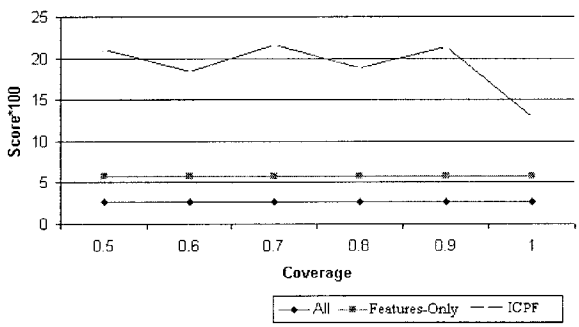

Figure 4.15: Evaluation Result

In a second analysis we evaluate the ability of the model to predict ICPageWords from a subset of the session pages. For all the above experiments, the models were trained on the session $S_{N-1} = \langle p_1, p_2, \ldots, p_{N-1} \rangle$. But here, the input to the model is a subsession $S_\ell = \langle p_1, p_2, \ldots, p_\ell \rangle$, where $\ell < N$. We suspect that the model learned from the largest subsession ($\ell = N - 1$) will perform better. The total length of user sessions varies considerably from session to session. Sessions range in length from under ten pages to well over twenty-five pages. In a session with ten pages, a model based on a prefix of length four

will be predicting six pages ahead. In a sequence of length twenty-five a model based on a prefix of length four will be predicting twenty-one pages ahead.

All results for our long-range prediction experiment are based on leave-one-subject-out testing. We present the results for NaïveBayes in Table 4.6 and C4.5 in Table 4.7.

Table 4.6: NaïveBayes Average F-Measure Measurement Matrix

	Subsession length ℓ				
h	1	2	3	4	≥ 5
1-10	26.50	24.51	21.21	20.06	15.15
11-15	18.99	20.21	19.09	20.94	16.66
16-20	26.57	26.27	26.08	21.79	14.57
21-25	23.33	25.13	19.99	20.21	11.82
>25	10.32	10.04	6.61	6.49	6.54

Table 4.7: C4.5 Average F-Measure Measurement Matrix

	Subsession length ℓ				
h	1	2	3	4	≥ 5
1-10	37.80	45.92	54.15	49.58	29.51
11-15	40.20	47.43	38.24	30.96	22.67
16-20	46.72	39.67	21.72	14.23	18.73
21-25	23.41	27.04	26.11	24.49	13.04
>25	11.22	11.22	11.22	10.18	6.98

When we compare the results for NaïveBayes in Table 4.6 to the results for C4.5 in Table 4.7, we see that C4.5 is again clearly superior. Within each table, we see that the F-score decreases as ℓ increases (i.e., as more pages are used to define the context). We suspect this is due to problems in our use of IC-session. It introduces more noise to the learner as we increase the number of pages that have been observed, since the user may have several different information needs during a long browsing session. When one considers that many of the ICPageWords are not even present in the limited prefix sessions, the performance is quite impressive.

4.7 Summary

When we compare the training data with the associated testing data, we find that only 30.77% of IC-pages in the testing data appear anywhere in the training data, and that the average support for these in-training IC-pages is only 0.269. So, for those co-occurrence based recommendation systems that could only use page frequency, about 70% of the IC-pages would never be recommended. There is only a small chance that the remaining 30% would ever be selected as a recommendation.

As we are not imposing any restriction on Web users' access to recommend relevant pages while no co-occurrence based methods would work. I propose to learn the ICPageWord prediction as the first step of the IC-page recommendation. The above testing and general evaluation results indicate that this is very promising and that it can produce a fairly accurate prediction.

Chapter 5

User Study 2: Evaluating Browsing Behavior Models

The results of ICPageWord prediction in Chapter 4.1.2 encourage us to move forward, that is, to predict relevant Web *pages* (i.e., IC-pages) that satisfy the user's current information need. We extend our ICPageWord model to this task. We also developed two new methods for training browsing behavior models, and provide algorithm to learn how to predict which words will retrieve relevant pages by querying a search engine.

5.1 Browsing Behavior Models

Chapter 4.1.2 describes a technique for determining which words are relevant. Here, we develop two other methods to label each session word w, and consequently we can train three kinds of models: ICPageWord, ICRelevantWord[1] , and ICQueryWord[2] . Each model produces a classifier, that takes a descriptions of how a word is used as input, and returns a bit indicating whether this word is relevant or not. Each of these models use training data to train their respective classifier.

In this way, we obtain 3 different datasets based on the same raw data (browsing features), but with (possibly) different labels for each word. For each dataset, we then run C4.5 [58] to produce a decision tree, which will later be used to predict which words are *significant* for each of these three models.

5.1.1 Browsing Feature Extraction

We consider all words that appear in all pages viewed, and compute 35 browsing features for each word in each IC-session including all "Search Query" features in Chapter 4.6.2 as well as a few new sequential attributes below, mainly based on the *Jewell weight* (Chapter 5.1.4). Refer to Appendix B for details.

latestAppearance	*relativeFreq*
ratioOccurences	*seqTFIDFWeight*
avTFIDFWeight	*varTFIDFWeight*
avJewellWeight	*varJewellWeight*
trendTFIDFWeight	*trendJewellWeight*
avTFIDFWeightBackward	*varTFIDFWeightBackward*
avJewellWeightBackward	*varJewellWeightBackward*
avTFIDFWeightForward	*varTFIDFWeightForward*
avJewellWeightForward	*varJewellWeightForward*

Note that all sequential features are extracted from the pages visited in an IC-session except for the search result pages and the last IC-page.

[1] also denoted as IC-Relevant in [78, 80]
[2] also denoted as IC-Query in [78, 80]

5.1.2 ICPageWord

The label here simply indicates whether this word, w, is an ICPageWord or not. In the training examples we know which pages are IC-pages, and hence we can simply "look up" this label.

5.1.3 ICRelevantWord

While we collect data in the user study, besides annotating IC-pages, we also ask the user to pick out only those words she thinks are relevant to her current search task from the current IC-session. These "relevance" bits are then used to label each word.

5.1.4 ICQueryWord

Here, I use a new mechanism to compute the weights of each word in an IC-session, and several new browsing features have been developed based on this new mechanism.

In general, let $Google(W)$ be the top page returned by Google when given the list of keywords W. For any page p, $Google^{-1}(p)$ is defined as a list of words in p that would cause *Google* to return p as the top page. This function can be approximated by finding ways to get *Google* to return sample pages from the OpenDirectory `http://dmoz.org`.

Here, a linear function is defined to compute a predictive score for each stemmed non-stopword w in p:

$$score(w) = \sum_{i=1}^{19} \beta_i \times PF_i(w)$$

where $PF_i(w)$ is the i-th page feature value of w. There are 19 page features for each w in p, including: number of occurrences of w; normalized TF/IDF of w; number of occurrences of w in the following "HTML context" [69]: "h1", "h2", "h3", "h4", "h5", "h6", "a", "title", "cite", "strong", "big", "em", "i", "b", "u", "blink", and "s".

To train these β_i parameters, we randomly select N pages $P = \{p_1, p_2, \ldots, p_N\}$ from OpenDirectory, and set these $\beta = \langle \beta_i \rangle$ values on the sample Web pages as follows.

BEGIN
Initialize β_i as a random number in $[0, 1]$
For each page $p_j \in P$
    compute the scores of all its words (W_k)
    rank W_k based on their scores
    choose top 4 words as keywords for querying search engine
    G is the top returned page
Compute $accuracy = \frac{|\{p_j \in P | p_j \equiv G\}|}{|P|}$ on P
Tune β_i to acquire the highest $accuracy$
END

After we obtain the tuned β, for any page, we compute the predictive score of each of its words, and choose the top m words as $Google^{-1}(p)$. We have tried $m \in \{2,3,4,5,6\}$, and found that $m = 4$ can acquire the highest accuracy, thus we choose the top $m = 4$ words as $Google^{-1}(p)$.

The predictive score of each word w in a Web page p can also be considered as one special kind of weight of w, which describes how likely w is to contribute to locating p by querying the search engine. Here, we denote the predictive score of w as its *"Jewell weight"*.

Given this notation, our goal is simply to identify $Google^{-1}$(IC-page). We define IC-QueryWord words as the subset of words in the session that belong to $Google^{-1}$(IC-page).

This $Google^{-1}$(IC-page) is similar to *Lexical signature [48]*, but here we extract more features from Web page, to acquire more accurate description of the role of each word in the page.

5.1.5 Identifying Appropriate Search Query

Whenever we need to predict a page, we first extract the browsing features of each of these words in the current session, and then run one of these classifiers to select a subset of these words. Unfortunately, each of these classifiers may identify hundreds of such significant words — far too many to submit to any search engine. Furthermore, we need to send a LIST of words, rather than a set. For these reasons, we also learn three sets of weights (one for each model) to rank these predicted words as follows:

$$score_\alpha(w) = \sum_{i=1}^{35} \alpha_i \times BF_i(w)$$

where $BF_i(w)$ is the i-th browsing feature value of the word w in the current session. We use the training data to set these $\alpha = \langle \alpha_i \rangle$ values as well. Let M be the number of words in the training set that were labeled as significant. For any value $\alpha = \langle \alpha_i \rangle$, let W_α^M be the M words with the highest $score_\alpha(\cdot)$ values, and for model γ (γ = ICPageWord, ICRelevantWord, or ICQueryWord), let

$$precision_\alpha^\gamma = \frac{|\{w \in W_\alpha^M \mid IC - \gamma(w) = 1\}|}{M}$$

be the fraction of W_α^M that are labelled as significant. For each γ , we set α to optimize this $precision_\alpha$ score. Note that we will find different sets of α weights for each IC-model.

At performance time, we first use the decision-tree classifier to filter away most of the words, and then run this linear function to rank the remaining words, and finally send the top $m = 4$ words to the search engine, in the order of rank.

5.2 *W*ebIC — A Goal-Directed Complete-Web Recommender System

*W*ebIC, whose interface is shown in Figure 5.1, is a client-side Web recommender system that employs the three browsing behavior models (described in Chapter 5.1) to suggest IC-pages.

As the user is browsing, she has the option of clicking the "Suggest" button to ask *W*ebIC to recommend a Web page. *W*ebIC uses information from the current session to recommend a page, from anywhere on the Web, that it predicts the user will find useful. Refer to Appendix C for a sample ICPageWord model, and page sequence that *W*ebIC produced a relevant page as recommendation.

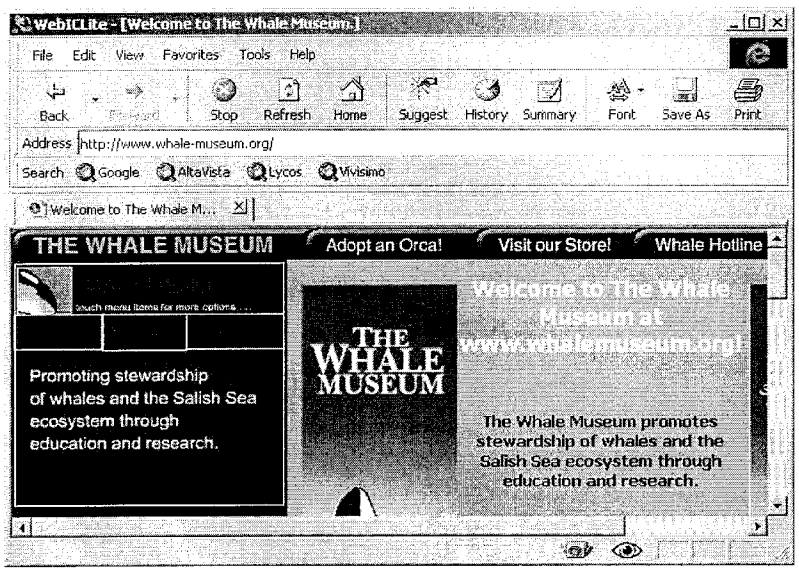

Figure 5.1: *W*ebIC — A Goal-Directed Complete-Web Recommender System

We assume that the user's current browsing activity is driven by a single information need. *W*ebIC first divides her browsing activities into IC-sessions, and then attempts to recommend web pages with relevant content, using only the information available in the browsing features of words in current session. The remainder of this section explains these steps in more detail (Figures 5.2A–D).

Step A: Identifying Browsing Sessions The process begins with a record of the pages the user has visited and the actions the user has applied to the pages. Heuristics (in Chapter 4.6.1) are used to identify IC-session; see Figure 5.2A.

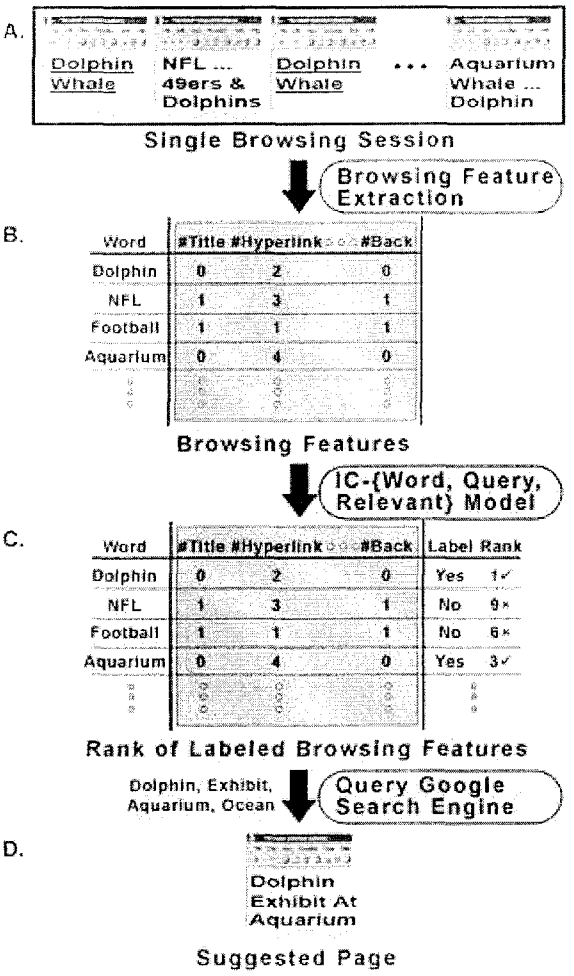

Figure 5.2: WebIC System at Performance Time

Step B: Assigning Features *W*ebIC next converts the browsing session into browsing features. This step has two substeps. The first substep is to represent the content of the pages in the browsing sequence by the set of words found on the pages in the current session. Second, *W*ebIC extracts browsing features to represent those aspects of a user's actions that can be used to infer her information needs. As shown in Figure 5.2B, this step assigns a value for each browsing feature to every word in the session.

Step C: Interpreting Features A browsing behavior model is used to decide, for each word in the session based on its browsing features, whether the word is likely to represent information content. This identifies a small subset of words likely to be relevant (Figure 5.2C).

Step D: Recommending Pages The final step is to recommend pages. A query formulator (Chapter 5.1.5) is used to turn the set of predicted words into a much shorter, ordered list of words that is then sent as a query to a commercial search engine. As shown in Figure 5.2D, the top ranked page returned in the list of results from the search engine is then presented to the user as a recommendation.

Obviously, browsing behavior models play a key role in *W*ebIC. To collect data for training these behavior models, we conduct a user study to obtain data for training and evaluating these models.

5.3 The LILAC Study

The study, LILAC (Learn from the Internet: Log, Annotation, Content) [82], gathers data from people working on common daily tasks. We use the data first to train our models, and then collect the evaluation of browsing behavior models.

LILAC study participants needed to install the *L*-WebIC system (Chapter 5.3.2) on their own computers, and browse their own choice of non-private, English language Web pages. *L*-WebIC kept track of all the interactions — storing a record of the pages the user visited, as well as the evaluation data for the pages that they considered to be relevant. Whenever the subject discovered an IC-page, she was instructed to label this page as an IC-page by clicking on the "MarkIC" button. [3] Here, *L*-WebIC then asked the subject to compare this page to an alternative page that it suggested, which was produced by running one of the models on the current user session. In the case that the subject could not find a page that

[3]Recall that labelling is not required for the end user.

addressed her information need, she had the option of clicking on the "Suggest" button, which retrieved a recommended page for her review. As before, the subject was asked to rate the usefulness of the suggested page with respect to her current information need. Note that rating process after both "MarkIC" and "Suggest" takes no more than a few seconds to complete.

There were five distinct steps involved in the LILAC study.

Pre-Test: The main purpose of the pre-test was to make *L*-WebIC more stable. As such, local colleagues were recruited to participate in the pre-test, using *L*-WebIC for 2 hours each week. They were asked to report any bugs and to make suggestions that would improve the usability of *L*-WebIC.

Pilot Study: After all participants have confirmed their participation in the study, we then randomly selected 10 subjects for a one-week pilot study. The purpose of the pilot study was to evaluate *L*-WebIC and to test the interface mechanisms before launching the regular study. The comments and feedback from the pilot study were very helpful in identifying potential problems and issues prior to the main study.

LILAC Study: The five-week main study was designed to quantitatively assess the different browsing behavior based models relative to a baseline model, and to gather data for additional further work. Ideally, we wanted to demonstrate that our browsing behavior models can work effectively on arbitrary pages taken from arbitrary sites on the Web. As such, our goal was to test our models when used by actual users working on real-life applications.

Follow-up Survey: We also selected 12 users for the follow-up survey. The goals of the follow-up survey were: 1) to improve our understanding of the hypothesis that a user's browsing actions provide information about her needs; 2) to gain an assessment of how significant this source of information is relative to other sources of information; 3) to test the usefulness of our assumptions about how a user's needs, page content and browsing actions can be represented and how relationships between these representations can be expressed; and 4) to evaluate how well our recommender system worked with real users during unrestricted browsing on the Web.

5.3.1 LILAC Subjects

A total of 104 subjects participated in the five-week LILAC study, of which 97 resided in Canada, and 7 resided in the United Stated. Table 5.1 gives the geographic locations of all LILAC subjects.

Canada		
	Edmonton	82
	Montreal	7
	Toronto	3
	Longueuil	2
	Calgary, St. Alberta, Beaumont	1 (each)
USA		
	PA, AL	2 (each)
	CA, NC, MA	1 (each)

Table 5.1: Location of LILAC Subjects

Of the 104 participants, 98 provided age/sex information. Among them, 47% were female and 53% were male. The age distribution of the subjects is shown in Table 5.2.

Range	Number of Subjects
$18-20$	23
$21-25$	41
$26-30$	20
$31-35$	11
over 35	3

Table 5.2: Age of LILAC Subjects

The subjects differed in the amount of priori experience they had using the Web. This helped us test our models in a more meaningful way, and make the results more promising.

5.3.2 *L*-WebIC: Enhanced *W*ebIC for LILAC

L-WebIC, whose interface is shown in Figure 5.3, is an enhanced *W*ebIC for the LILAC study. It has several features that differ from *W*ebIC.

In *L*-WebIC, we not only record the URL and time stamp of each browsing action, but also the HTML source of the web pages that the user has visited. For a frame Web page, we download all the involved frame pages. We want to record the exact page sequence that the user has seen while browsing. We download the exact page content that the user has visited, and we also record the user's action sequence in the browsing session.

Annotation

There are two purposes of the "Annotation" in *L*-WebIC:

1. Just as with AIE (Chapter 4.3), by distributing *L*-WebIC to people for their ordinary web browsing, we can collect *A*WLs to train a user-independent population model.

2. Log data for one specific user can be collected and used to build the personalized model.

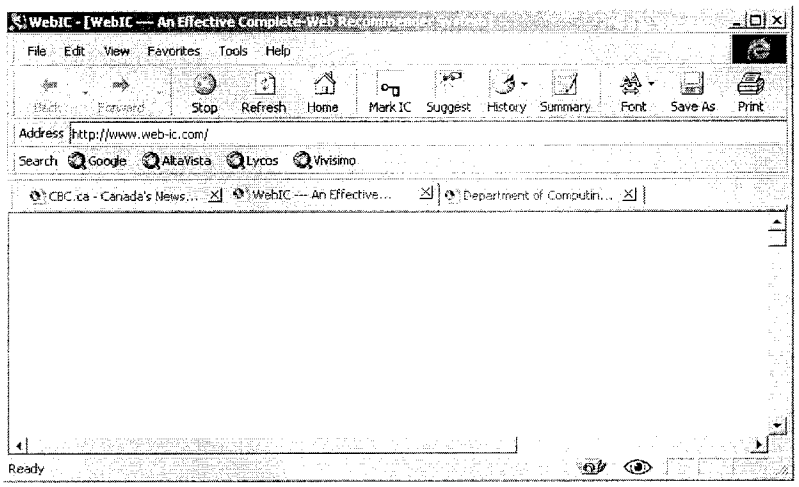

Figure 5.3: *L*-WebIC — An Enhanced *W*ebIC for LILAC

Whenever a visited page qualifies as an IC-page, the user is instructed to click the "MarkIC" button in *L*-WebIC. Having done so, she must give the IC-page a label and optionally specify why it qualifies, on a pop-up window. Fig. 5.4 shows the pop-up window that allows the user to input the label and description of the IC-page.

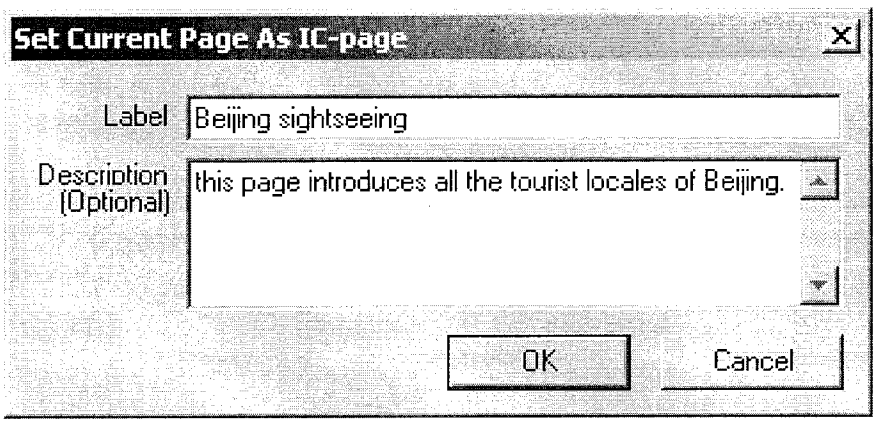

Figure 5.4: Annotation of *L*-WebIC

Training

To help us suggest useful Web pages, the *L*-WebIC system trains several general "browsing behavior models" to learn the relationship between users' page-action sequences and their

information needs based on browsing behavior.

Evaluation

L-WebIC provides a user interface for the subjects to evaluate the suggested IC-pages, which is the main purpose of the LILAC study (Chapter 5.3). Figure 5.5 shows the evaluation interface.

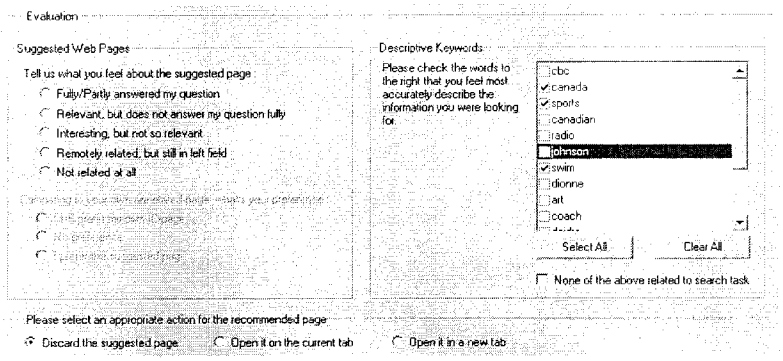

Figure 5.5: WebIC Evaluation Interface

Here, whenever the user requests a recommendation by clicking the "Suggest" button, *L*-WebIC presents a recommendation page, and asks the user to evaluate this proposed page.

Another goal of LILAC is to collect annotated Web logs for future research. We therefore instructed these paid participants to click "MarkIC" whenever they found a page they consider to be an IC-page. After marking an IC-page, *L*-WebIC will recommend an alternative Web page (different from the IC-page), just as if the user had clicked "Suggest". Once again, *L*-WebIC will then ask the user to evaluate this recommended page.

In order to evaluate the recommendation, *L*-WebIC will ask the user to provide feedback in several key areas. First, the user is instructed to "Tell us what you feel about the suggested page", in order to indicate whether the information provided on the page suggested by *L*-WebIC was relevant to her search goal. There are two categories of relevance evaluations: *related* and *not related at all*. We further divided the *related* category into four different levels, including, "Fully answered my question", "Somewhat relevant, but does not answer my question fully", "Interesting, but not so relevant", and "Remotely related, but still in left field".

As mentioned, after the user has explicitly identified an IC-pageusing "MarkIC", *L*-WebIC will suggest an alternative page, and ask the user to evaluate how the content of this page compares to the original IC-page. To accomplish this, the user needs to examine both

the suggested page and her own IC-page, while she is asked the question, "Comparing to your own IC-page, what's your preference?". The user must select from one of the following three options: "I still prefer my own IC-page", "No preference", and "I prefer the suggested page".

Third, the user will be asked to select informative "Descriptive Keywords" from a short list of words that L-WebIC predicted as relevant words. The information collected here will be used to train behavior models as described in Chapter 5.1.

Finally, the user will be asked to, "Please select an appropriate action for the recommended page", from one of three choices: "Discard the suggested page", "Open it on the current tab", and "Open it in a new tab". Analysis of these data will allow us to evaluate the user's impression of the suggested page.

5.4 Empirical Results

In LILAC, a total of 104 subjects participated in the 5-week study, visiting 93,443 Web pages. Over this period of time, the users marked 2977 IC-pages (i.e., clicking "MarkIC") and asked for recommendations by clicking the "Suggest" button 2531 times.

We used the collected data during the study period to retrain each of our IC-models. That is, the users initially used the ICPageWord_0, ICRelevantWord_0 and ICQueryWord_0 models, which were based on a model obtained prior to this study. At the 3rd week, they used the $\text{ICPageWord}_{(1+2)}$, $\text{ICRelevantWord}_{(1+2)}$ and $\text{ICQueryWord}_{(1+2)}$ models, based on the training data obtained from week 1 and 2, in addition to the prior model. And so on and so forth.

The Followed Hyperlink Word (FHW) model is used as a baseline model for this study. It is similar to the Inferring User Need by Information Scent (IUNIS) model presented in [15]. The basic idea is to collect those words found in the anchor text of the followed hyperlinks in the page sequence, rank based on their frequency, then pick out top 4 words as query keywords. As such, there is no training involved in this model.

5.4.1 Overall Results

At the conclusion of the LILAC study, we collected the evaluation results of these IC-models and the baseline model.

As these two conditions are significantly different, we analyzed the evaluation results for "Suggest" and "MarkIC" separately.

Figure 5.6 indicates how often the user considered the "Suggest"ed page to be "related" (Chapter 5.3.2), and Table 5.3 shows the results in 4 sub-categories. We see that each of

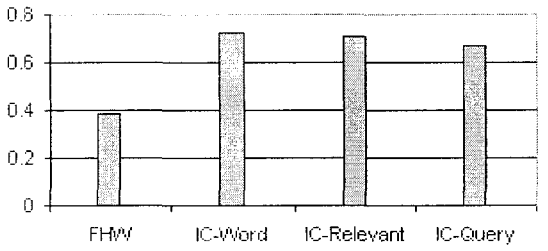

Figure 5.6: How often Users Rated a Recommended Page as "related", after "Suggest"

our 3 IC-models works much better than the baseline model as each returned page that was considered to be related over 65% of the time, versus the 38% for FHW.

Model	Fully	Somewhat	Interesting	Remotely
FHW	0.068	0.118	0.093	0.106
ICPageWord	0.253	0.237	0.116	0.121
ICRelevantWord	0.202	0.246	0.158	0.098
ICQueryWord	0.201	0.216	0.126	0.126

Table 5.3: Ratio of Relevant Recommended Pages after "Suggest"

Figure 5.7 shows the evaluation results for the recommended pages after the user clicked "MarkIC", which is the sum of the 4 "related" categories in Table 5.4.

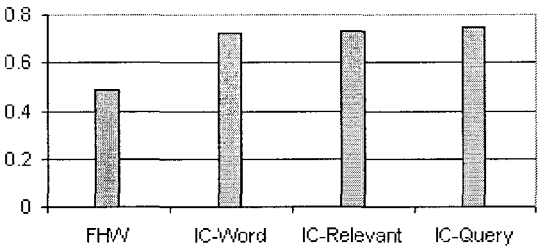

Figure 5.7: How Often User Rated a Recommended Page as "Related", after "MarkIC"

We again observe that our IC-models work better than FHW. The scores for ICPageWord and ICRelevantWord remained at around 70%, roughly the same values they had for the "Suggest" case, while the ICQueryWord model increased from 66% to 74%. We also observed that FHW increased by almost 10%. As an explanation, we speculate the following: If the subject is able to find an IC-page, then the links followed in the current session appear to provide a very strong hint of what constitutes the relevant information content; and FHW benefits significantly from this hint.

To test whether there is any significant difference among these models, we ran several

Model	Fully	Somewhat	Interesting	Remotely
FHW	0.114	0.134	0.134	0.106
ICPageWord	0.253	0.227	0.125	0.117
ICRelevantWord	0.276	0.235	0.136	0.085
ICQueryWord	0.291	0.217	0.127	0.111

Table 5.4: Ratio of Relevant Recommended Pages after "MarkIC"

statistical tests on the evaluation results. The Friedman ANOVA is a statistical measure of two-way analysis of variance by ranks, and as such, is a nonparametric test for use with k repeated (or correlated) measures. In LILAC, each subject was required to evaluate the recommended page generated by one of the models, thus we need to compute each subject's evaluation on each of these randomly selected models. This is accomplished by evaluating the Friedman test using $k = 4$. The null hypothesis states that there is no significant difference among the four models. The result of the Friedman statistic is 56.6395, which corresponds to $p < 0.0001$. Given the threshold for significance (p=0.05), these results demonstrate that there does exist significant statistical difference among these four different models used in the LILAC study.

The Wilcoxon signed-ranks test is another nonparametric test that can be used for 2 repeated (or correlated) measures. As such, we run the Wilcoxon test on each pair of the four models, and the results are presented in Table 5.5.

	p
FHW $\leq$ ICPageWord	<0.0001
FHW $\leq$ ICRelevantWord	<0.0001
FHW $\leq$ ICQueryWord	<0.0001
ICPageWord$\neq$ ICRelevantWord	0.6727
ICPageWord$\neq$ ICQueryWord	0.7796
ICRelevantWord$\neq$ ICQueryWord	0.7959

Table 5.5: Wilcoxon's Test on Overall Results

From Table 5.5, we conclude that each of the three different IC-models perform better than the baseline model (i.e., FHW). This result validates our basic assumption that we are able to provide useful recommendations by integrating the user's browsing behaviors into the prediction.

In addition to the aggregated results, we also compute how each of the three IC-models performed over the course of the LILAC study. Table 5.6 shows the evaluation of the various different IC-models for each week of the experiment.

From Table 5.6, we can see that performance does not increase linearly with the additional training data available over the study period. This is especially true for the percentage

Week	Fully	Somewhat	Interesting	Remotely	Irrelevant
ICPageWord					
1	0.1559	0.2271	0.1373	0.2254	0.2542
2	0.2674	0.2396	0.0972	0.1319	0.2639
3	0.1964	0.2054	0.1964	0.0804	0.3214
4	0.3056	0.2222	0.0833	0.1667	0.2222
5	0.2593	0.2593	0.1111	0.0741	0.2963
ICRelevantWord					
1	0.1270	0.2052	0.1652	0.2191	0.2835
2	0.2423	0.2500	0.1500	0.1000	0.2577
3	0.2062	0.1856	0.1340	0.1031	0.3711
4	0.3286	0.2571	0.1714	0.0429	0.2000
5	0.2143	0.2857	0.0714	0.0714	0.3571
ICQueryWord					
1	0.1456	0.2253	0.1577	0.2236	0.2478
2	0.2491	0.2058	0.1227	0.1480	0.2744
3	0.3173	0.1635	0.1346	0.0865	0.2981
4	0.2195	0.3049	0.1220	0.0854	0.2683

Table 5.6: Weekly Data for IC-Models

of these "Fully" suggested pages. The training data and the subjects' expectation might be the main reason for this observed fluctuation in performance. Indeed, the models for the first week are based on the pre-test data, which can be considered less reliable, and are thus likely contributed to the lower score for these models initially. Starting from the second week, each of the models were trained using all previous LILAC data. This resulted in an increase in the performance of the models, as expected, since the training data now included data submitted by the study participants. But while the subjects obtained greater experience, their expectations of the system could also be expected to change. For example, based on the improvement in the recommendations seen in the second week, the subjects would expect that the performance would continue to increase in the following weeks. However, if the improvement seen in the third week did not match these higher expectations, then the users may have interpreted the performance as lower even though the third week's models were actually more refined.

5.4.2 Simple Browsing Behavior Models

All the IC-models that we trained in LILAC were decision trees. After investigating these decision trees, we found several significant browsing features including, "ratio of the pages in the session that contain word w", and "latest relative location of the page that contained w", etc. We selected three such features and ranked all the words based on these features.

Then we selected the top $n = 4$ words as query keywords. The goal was to evaluate how much other browsing features contribute to the overall prediction. To examine this question, we compared these simplified models (described below) to the complete IC-models.

Most Frequent Word (MFW) In this model, the number of occurrences of a word w in the session is multiplied by the fraction of the pages that contain w.

Simple TFIDF (STFIDF) In this model, rather than computing w's TF/IDF on each page, we just treat all the pages in the session as one page, and calculate w's TFIDF weight in this virtual single page.

Both the MFW and STFIDF models were tested only once during the study, week 1 for MFW and week 3 for STFIDF. Figure 5.8 shows the evaluations of MFW and the three different IC-Models in Week 1 of the study.

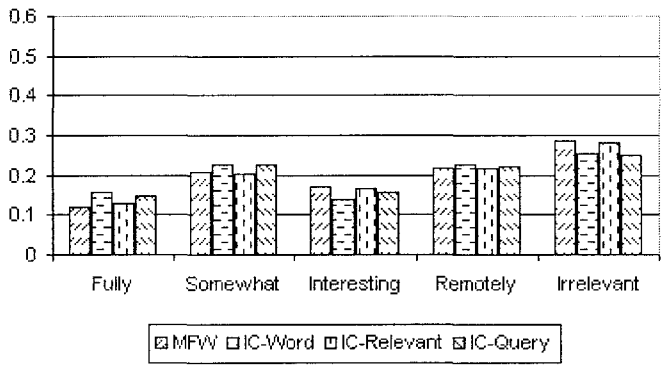

Figure 5.8: Comparison of MFW model to IC-models Data from Week 1

	p
MFW $\leq$ ICPageWord	0.0704
MFW $\leq$ ICRelevantWord	0.5408
MFW $\leq$ ICQueryWord	0.2023

Table 5.7: Wilcoxon's Test on MFW and IC-models Data from Week 1

We then ran the Wilcoxon test on these data from Week 1 to see whether there exists significant difference between MFW and any of our IC-Models. As shown in Table 5.7, we cannot detect any significant improvement by these IC-Models. Note that for the first week the IC-models were trained on the data from the pre-test phase of the study, and since the main purpose of this phase was to evaluate the stability of *L*-WebIC, it is not suitable to use these data for training. From the second week of the study, all the IC-Models were trained on

LILAC data, thus we compare MFW to the average results of these IC-Models in the study (excluding the first week of data). Figure 5.9 shows the results of the comparison between MFW and the three different IC-models. From these results (Table 5.8), we conclude that there exists a significant difference between MFW and ICRelevantWord.

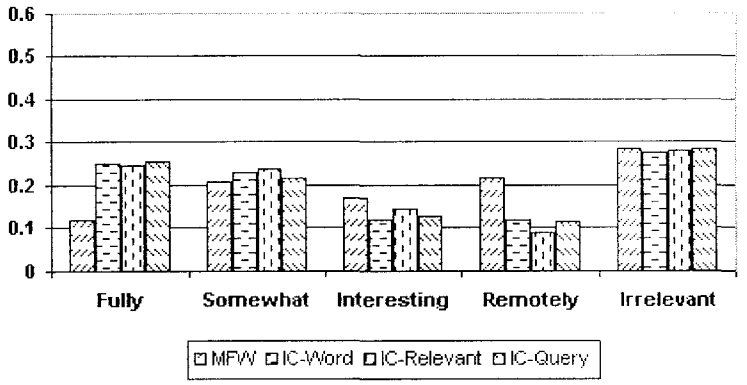

Figure 5.9: Comparison of MFW vs. IC-models on Average from Week 2

	p
MFW $\leq$ ICPageWord	0.1443
MFW $\leq$ ICRelevantWord	0.0168
MFW $\leq$ ICQueryWord	0.0826

Table 5.8: Wilcoxon's Test on MFW and IC-Models on Average From Week 2.

We also ran the Wilcoxon test on these data in Week 3 to compare STFIDF to each of the three different IC-Models. Figure 5.10 shows the evaluation of STFIDF and IC-Models in Week 3, and the Wilcoxon test results are presented in Table 5.9.

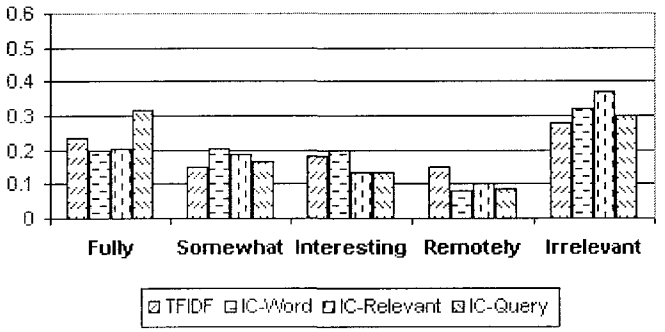

Figure 5.10: Comparison of Simple TFIDF Model to IC-models Data at Week 3

	p
STFIDF $\leq$ ICPageWord	0.69083
STFIDF $\leq$ ICRelevantWord	0.3965
STFIDF $\leq$ ICQueryWord	0.0438

Table 5.9: Comparison of STFIDF vs. IC-Models at Week 3

We conclude from the results in Figure 5.9 that ICQueryWord is able to generate better recommendations than STFIDF. In fact, following careful investigation of the log data, we noticed that STFIDF only generates good recommendations under certain conditions. Specifically, STFIDF was discovered to work well only when all of the sessions contain highly correlated pages such that the user never browses more than one web site without either returning to the search results page or terminating the session. For these highly correlated pages, especially where the user returns to the search results page multiple times within the same session, the appearance of some words are significantly increased, which will allow the STFIDF model to artificially find relevant words. Indeed, for those cross-site or cross-topic sessions, the performance of STFIDF is found to drop dramatically.

5.4.3 Alternate Training for ICQueryWord Model

In order to train our IC-models, the study participants must actively label IC-pages while browsing the Web; this is so inconvenient for the user that is it unrealistic in a production version of the system. To solve this data collection problem, we propose to *passively* train the ICQueryWord model based on previous evaluation results. Recall that every time a user requests a recommendation, we generate a search query using one of the models, and send the query to search engine to produce pages, then return a page to the user, which she must then evaluate. If we assume that the search engine (e.g., Google) remains relatively consistent over time, we can label each query by using the actual evaluation of the recommended page. For example, the user may evaluate the top returned page as "Fully", which was based on the query q ="data mining software public". We can then label q as "Fully". From these results, we can extract only the queries that are evaluated as "Fully" as belonging to the ICQueryWord model.

In LILAC, the ICQueryWord models derived in the first four weeks of the study were trained based on annotations (refer to Chapter 5.1.4). In the fifth week we changed the experimental protocol to train the ICQueryWord model based on only those queries that resulted in a "Fully" evaluation in the previous weeks. The evaluation results of the two training methods are presented in Figure 5.11.

We conclude from Figure 5.11 that by using the previous evaluation results we can obtain

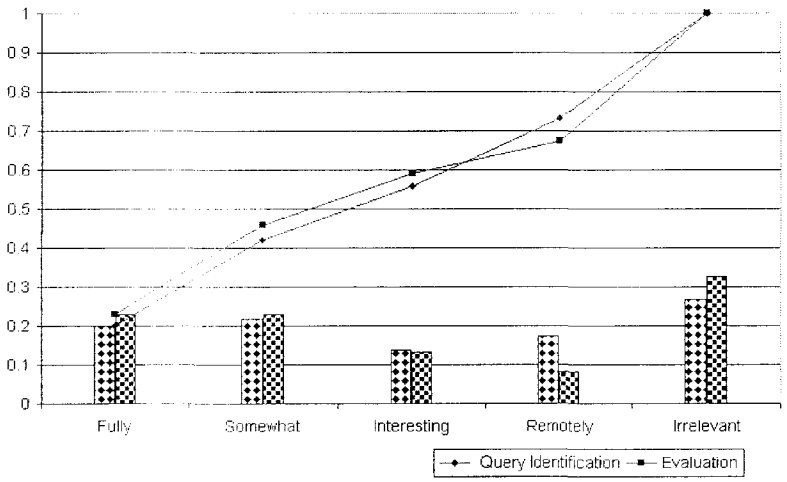

Figure 5.11: Training ICQueryWord Models

a similar performance as compared to when the ICQueryWord model was trained directly on the original IC-pages. This result is significant in that it will allow us to continuously refine the ICQueryWord model without requiring user input to label IC-pages while browsing the Internet. Importantly, this alternate training method will make the recommender system more realistic in real world situations, as opposed to mainly research environments.

5.5 Off-line Evaluation of Web User Models using LILAC Data

We can conclude that these browsing behavior models work better than the baseline model based on the evaluation results collected in LILAC, but it is time consuming and costly financially to conduct such a user study whenever a new model has been developed. In this section, I propose a novel method to assess the performance of Web user models off-line (i.e., infer the evaluation by an off-line computation). This chapter extends the work presented in [81].

We assume that the user's evaluation of the suggested page is based on the similarity between her own IC-page and the suggested page. For each MarkIC session in LILAC, the user annotated a page as an IC-page whenever it satisfied her current information need. In such cases, the IC-page can be considered as the page that "Fully answered my question". If the suggested page is evaluated as "Fully", it must contain very similar content to the IC-page. Alternatively, a page that is evaluated as "Irrelevant" contains unrelated information.

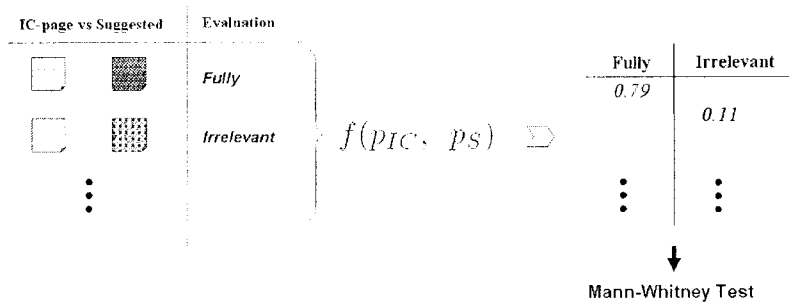

Figure 5.12: Similarity Function

Here, we only consider the two extreme evaluation options: "Fully" and "Irrelevant". Our goal is a "similarity function", $f(A, B)$, that takes as input a pair of web pages, and returns a large number iff they are similar. We could then evaluate $f(p_{IC}, p_S)$ on the IC-page (p_{IC}), and the suggested page (p_S). Ideally, we would want this function to return a large number if p_S was a "Fully" page, and a small number if p_S was an "Irrelevant" page.

5.5.1 Similarity Function

Of the multitude of functions that can be used to calculate the similarity between a given pair of pages, it is very important to select a function that matches our understanding of the evaluations: the values of $f(p_{IC}, p_F)$ over the span of "Fully" pages p_F must be significantly different from the values of $f(p_{IC}, p_I)$ over the "Irrelevant" pages p_I.

For each MarkIC session in LILAC, we collected a pair of pages: 1) an annotated IC-page (p_{IC}); 2) a suggested page generated by L-WebIC (p_S). We then compute the similarity $f(p_{IC}, p_S)$ together with the evaluation label of the suggested page. In doing so, we can construct an independent sample table consisting of the similarities of "Fully" and "Irrelevant" pages compared with the IC-page. We can then analyze these data using a Mann-Whitney test to check whether there exists significant difference between "Fully" and "Irrelevant". Figure 5.12 describes the whole process to collect the data to verify any purported similarity function.

Below we propose four different similarity functions. These use $W(p_S)$ and $W(p_{IC})$, which respectively denote the bags of words in the Suggested page and IC-page, after removing stop words and stemming.

ITM: Information Theoretic Measure This function is a simplified version of the measure that was proposed in [32].

$f_{ITM}(p_{IC}, p_S) = \frac{|W(p_{IC}) \cap W(p_S)|}{|W(p_{IC}) \cup W(p_S)|}$

Recall: ICPageWord Recall This function shows how likely the suggested page is to be able to recall the words in the IC-page.

$f_{Rec}(p_{IC}, p_S) = \frac{|W(p_{IC}) \cap W(p_S)|}{|W(p_{IC})|}$

avTFIDF: Average TF/IDF of Common Words This function returns the average TF/IDF weight of the words in an IC-page that also appear in the suggested page.

$f_{TFIDF}(p_{IC}, p_S) = \frac{\sum_{w \in W(p_{IC}) \cap W(p_S)} \text{TFIDF}(w \in W(p_{IC}))}{|W(p_{IC}) \cap W(p_S)|}$

avRankTFIDF: Mean of Ranks of the Common Words' TFIDF This function ranks all the words in IC-page based on TFIDF weights. from the highest to the lowest, and returns the mean of ranks of the words in $W(p_{IC}) \cap W(p_S)$.

$f_{Rank}(p_{IC}, p_S) = \frac{\sum_{w \in W(p_{IC}) \cap W(p_S)} \text{TFIDFRank}(w \in W(p_{IC}))}{|W(p_{IC}) \cap W(p_S)|}$

For each similarity function, we collect the testing samples by following the procedure in Figure 5.12, and then perform the directional Mann-Whitney test (Appendix A) to determine whether there is a significant difference between the "Fully" and "Irrelevant" cases. The results, shown on Table 5.10, indicate that each of the four functions can detect the significant difference between "Fully" and "Irrelevant" pages as compared with IC-pages.

	Hypothesis	Confidence Intervals	p
ITM	Fully>Irrelevant	>0.027	<0.0001
Recall	Fully>Irrelevant	>0.045	<0.0001
avTFIDF	Fully>Irrelevant	>0.007	<0.0001
avRankTFIDF	Fully<Irrelevant	<-4.554	0.0055

Table 5.10: Mann-Whitney Test on Different Similarity Functions

5.5.2 Validating Similarity Functions on LILAC Data

In LILAC, every time the user annotates an IC-page, *L*-WebIC randomly chooses one of four models to generate a suggested page for evaluation. Section 5.4.1 showed that the three IC-models work better than the baseline model (i.e., FHW), based on the subjects' evaluations.

To validate the functions that we proposed in Section 5.5.1, we performed an analysis of these functions on LILAC data, to determine whether the results are consistent with the conclusions that we have made based directly on evaluation results.

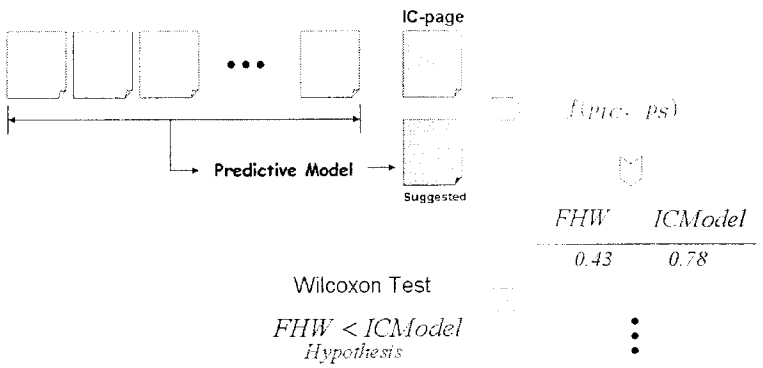

Figure 5.13: Validation on LILAC Data

For one-quarter of the MarkIC sessions within the LILAC study data, *L*-WebIC selected the baseline FHW model. We can now compute the similarity between the user's IC-page, p_{IC}, and this proposed p_{FHW} page, $f(p_{IC}, p_{FHW})$, using each of the 4 functions, $f \in \{f_{ITM}, f_{Rec}, f_{TFIDF}, f_{Rank}\}$.

In each such case, we will now generate 3 other proposed pages based on that user session, one using each of the three other IC-models (ICPageWord, ICRelevantWord, and ICQueryWord) and call them p_{Word}, $p_{Relevant}$ and p_{Query}. We can then compute the similarity between IC-page p_{IC} and each of these pages: $f(p_{IC}, p_{Word})$, $f(p_{IC}, p_{Relevant})$ and $f(p_{IC}, p_{Query})$.

Similarly, if any of the above IC-Models is chosen during the user study, we can again compute $f(p_{IC}, p_{\chi})$ for the p_{χ} suggested. We can also run the FHW model on the same session to produce p_{FHW}, and then compute $f(p_{IC}, p_{FHW})$.

For each similarity function, we run the same process on all MarkIC sessions as shown in Figure 5.13. For each MarkIC session, we can obtain a pair of similarities, $f(p_{IC}, p_{FHW})$ and $f(p_{IC}, p_{\chi})$. To test the hypothesis that each IC-Model is better than FHW, we perform a statistical test (i.e., Wilcoxon; Appendix A) on the correlated samples. If the p value is less than 0.05, then we can conclude that the ICModel is better than the FHW model.

	FHW<ICPageWord	FHW<ICRelevantWord	FHW<ICQueryWord
ITM	<0.0001	0.0002	<0.0001
Recall	0.087	0.0213	0.003
avTFIDF	0.1104	0.0011	0.0002
avRankTFIDF	0.0057	<0.0001	<0.0001

Table 5.11: Wilcoxon Test on LILAC Data

Table 5.11 presents all the p values of each similarity function on each hypothesis. It

clearly demonstrates that the ITM and Rank (avRankTFIDF) functions can detect a significant difference between FHW and any of ICModels, which is consistent with the overall results in Table 5.5.

Chapter 6

Future Work and Contribution

6.1 Future Work

Despite the promising success of the GCW recommender system in the user studies, there are still lots of work that should be done to make it more useful in real-life applications.

We are currently investigating more efficient ways to predict IC-pages, and ways to further increase the recall for the positive prediction. We also plan to collect more AWLs, and explore the potential of content and structure mining, as well as tools for learning from imbalanced datasets to aid us in this endeavor.

We have extracted some browsing features for each word in the observed click stream, and we think that these features represent how the user treats the observed information. But, there may be other attributes that are also very useful in capturing the behavior of Web users. We intend to incorporate other browsing features.

For each word, we only compute its own attributes, but ignore its context in the page content, such as the relationship among the words in the same sentence, same paragraph, or same page. The attributes we are now using are applied to distinct words, but actually, it is quite possible that several words in the IC-Session may have the same meaning. We therefore plan to explore Natural Language processing systems to extend the range of our predicted relevant words, such as Word Sense Disambiguation (WSD) [1, 73, 42, 75, 60].

For the learning part of the ICPageWord prediction, we have tried C4.5, which is very reliable. In future research, we intend to try different learning algorithms, such as support vector machines, or neural networks, to find more accurate predictors, and the emerging Web technology such as semantic Web to get a better understanding of the context of arbitrary pages.

To deal with the imbalanced data, we chose downsampling rather than over-sampling, as down-sampling seems work better. But while down-sampling may increase the recall of the prediction, it might decrease the precision. Are there any other methods that can perform better than down-sampling, or can we develop new learning algorithms to avoid the imbalance of the learning?

Each of our current models is basically user-independent. We are considering ways to personalize the generic model by assigning each user individual prediction weight for the generic one [36].

We are also exploring the best way to apply the browsing behavior models to other applications besides Web recommendation, such as prefetching [74], topic focused crawling [45], etc.

6.2 Contribution

The ultimate goal of my research is to help Web users by suggesting relevant pages from anywhere on the Web. To address this challenge, I have developed a Goal-Directed Complete-Web recommender system to recommend pages to satisfy the user's current information need, without requiring any explicit input. The recommender system uses browsing models that describe how a user locates useful information (IC-pages) on the Web. The models are not based on a specific set of words or URLs, but rather on a user's observable behavior in response to the information within the pages visited.

We conducted two user studies to collect annotated Web logs and evaluate the browsing behavior models by actual people. We considered several browsing behavior models to determine which words encountered in the current browsing session would be relevant to the user's search task, which are then used to locate the pages that satisfy their information needs. In particular, we investigated a general way to extract relevant information based only on the user's current web session: by finding browsing features of the words that appear, then using a classifier to determine which of these words are significant for locating IC-pages to satisfy the current information need. Three models have been developed within this framework: ICPageWord, ICRelevantWord, ICQueryWord. The results collected from LILAC show that all three approaches work effectively, finding relevant pages approximately 70% of the time, and that all three models are superior to a plausible alternative approach (FHW). We also provided a way to obtain comparable performance without requiring as additional annotation from the users, which will help us in producing an even more practical Web recommendation system.

Bibliography

[1] E. Agirre and G. Rigau. Word sense disambiguation using conceptual density. In *Proceedings of 15th International Conference on Computational Linguistics, COLING 96*, Copenhagen, Denmark, 1996.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conference on Very Large Databases (VLDB'94)*, Santiago, Chile, Sep 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the Int'l Conference on Data Engineering (ICDE)*, Taipei, Taiwan, Mar 1995.

[4] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the Int'l Conference on Data Engineering (ICDE)*, Taipei, Taiwan, Mar 1995.

[5] Corin Anderson and Eric Horvitz. Web montage: A dynamic personalized start page. In *Proceedings of the 11th World Wide Web Conference (WWW 2002)*, Hawaii, USA, 2002.

[6] D. Billsus and M. Pazzani. A hybrid user model for news story classification. In *Proceedings of the Seventh International Conference on User Modeling (UM '99)*, Banff, Canada, 1999.

[7] M. Blackmon, P. Polson, M. Kitajima, and C. Lewis. Cognitive walkthrough for the web. In *2002 ACM conference on human factors in computing systems (CHI'2002)*, pages 463–470, 2002.

[8] J. Borges and M. Levene. Data mining of user navigation patterns. In *Proceedings of Workshop on Web Usage Analysis and User Profiling (WEBKDD)*, San Diego, Ca., aug 1999.

[9] Jose Borges. *A Data Mining Model to Capture User Web Navigation Patterns.* PhD thesis, Department of Computer Science, University College London, London University, 2000.

[10] Randolph Bucklin and Catarina Sismeiro. A model of web site browsing behavior estimated on clickstream data.

[11] Jay Budzik and Kristian Hammond. Watson: Anticipating and contextualizing information needs. In *Proceedings of 62nd Annual Meeting of the American Society for Information Science*, Medford, NJ, 1999.

[12] M. Chen, J. Park, and P. Yu. Efficient data mining for path traversal patterns in distributed systems. In *Proc. of the 16th IEEE Intern'l Conf. on Distributed Computing Systems*, pages 385–392, May 1996.

[13] M. Chen, J. Park, and P. Yu. Efficient data mining for path traversal patterns. *IEEE Trans. on Knowledge and Data Engineering*, 10(2):209–221, Apr 1998.

[14] D. Cheung, J. Han, V. Ng, A. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *Proc. of 1996 Int'l Conf. on Parallel and Distributed Information Systems (PDIS'96)*, Florida, USA, Dec 1996.

[15] E. Chi, P. Pirolli, K. Chen, and J. Pitkow. Using information scent to model user information needs and actions on the web. In *ACM CHI 2001 Conference on Human Factors in Computing Systems*, pages 490–497, Seattle WA, 2001.

[16] Chun Wei Choo, Brian Detlor, and Don Turnbull. A behavioral model of information seeking on the web – preliminary results of a study of how managers and it specialists use the web. In Cecilia Preston, editor, *Proceedings of the 61st Annual Meeting of the American Society for Information Science*, pages 290–302, Pittsburgh, PA, Oct 1998.

[17] Chun Wei Choo, Brian Detlor, and Don Turnbull. Working the web: An empirical model of web use. In *HICSS 33 (Hawaii International Conference on Systems Science)*, Jan 2000.

[18] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1), 1999.

[19] Richard Duda and Peter Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[20] U. Fayyad and K. Irani. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence(IJCAI '93)*, pages 1022–1027, 1993.

[21] C.L. Giles G. Flake, S. Lawrence. Efficient identification of web communities. *ACM Conference on Knowledge and Data Discovery (KDD 2000)*, pages 150–160, 2000.

[22] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proc. 2000 Int. Conf. on Knowledge Discovery and Data Mining (KDD'00)*, Boston, MA, Aug 2000.

[23] Rob Holte, Nathalie Japkowicz, Charles Ling, and Stan Matwin. *AAAI'2000 Workshop on Learning from Imbalanced Data Sets*. AAAI Press, 2000.

[24] N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI '2000)*, 2000.

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

[26] George John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Francisco, 1995. Morgan Kaufmann Publishers.

[27] Eric Johnson, Wendy Moe, Peter Fader, Steven Bellman, and Jerry Lohse. On the depth and dynamics of world wide web shopping behavior. *Management Science*, 50(3):299–308, 2004.

[28] D. Kelly and N.J. Belkin. Display time as implicit feedback: Understanding task effects. In *Proceedings of the 27th Annual ACM International Conference on Research and Development in Information Retrieval (SIGIR '04)*, pages 377–384, Sheffield, UK.

[29] Alfred Kobsa and Josef Fink. Performance evaluation of user modeling servers under real-world workload conditions. In *Proceedings of User Modeling(2003)*, pages 143–153, Johnstown, USA, June 2003.

[30] David Lewis and Kimberly Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

[31] H. Lieberman. Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, Montreal, Canada, Aug 1995.

[32] DeKang Lin. An information-theoretic definition of similarity. In *Proceedings of International Conference on Machine Learning*, Madison, Wisconsin, July 1998.

[33] C. Ling and C. Li. Data mining for direct marketing problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, New York, NY, 1998. AAAI Press.

[34] Bing Liu, Wynne Hsu, and Yiming Ma. Pruning and summarizing the discovered associations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, CA, USA, Aug 1999.

[35] John Lynch and Dan Ariely. Wine online: Search costs and competition on price, quality, and distribution. *Marketing Science*, 19(1):83–103, 2000.

[36] E. Manavoglu, D. Pavlov, and C.L. Giles. Probabilistic user behavior models. In *Third IEEE International Conference on Data Mining (ICDM 2003)*, pages 203–210.

[37] H. Mannila, H. Toivonen, and A. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(2):259–289, Nov 1997.

[38] Bamshad Mobasher, R. Cooley, and J. Srivastava. Automatic personalization through web usage mining. Technical Report TR99-010, Department of Computer Science, Depaul University, 1999.

[39] Bamshad Mobasher, Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. Hypergraph based clustering in high-dimensional data sets: A summary of results. *IEEE Bulletin of the Technical Committee on Data Engineering*, 21(1), Mar 1998.

[40] Wendy Moe, Hugh Chipman, Edward George, and Robert McCulloch. A bayesian treed model of online purchasing behavior using in-store navigational clickstream.

[41] Maurice Mulvenna, Sarabjot Anand, and Alex Bchner. Personalization on the net using web mining: introduction. *Communications of ACM*, 43(8):122–125, Aug 2000.

[42] Hwee Tou Ng and Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 40–47, Santa Cruz, CA, USA, 1996.

[43] Alvaro Ortigosa and Rosa Carro. Agent-based support for continuous evaluation of e-courses. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics(2002)*, volume 2, pages 477–480, Orlando, Florida, 2002.

[44] Alvaro Ortigosa and Rosa Carro. The continuous empirical evaluation approach: Evaluating adaptive web-based courses. In *Proceedings of User Modeling(2003)*, pages 163–167, Johnstown, USA, June 2003.

[45] G. Pant, K. Tsioutsiouliklis, J. Johnson, and C.L Giles. Panorama: extending digital libraries with topical crawlers. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL 2004)*, pages 142–150.

[46] J. Park, M. Chen, and P. Yu. Efficient parallel data mining for association rules. In *Proc. of the ACM 4th Intern'l Conf. on Information and Knowledge Management*, pages 31–36, Nov 1995.

[47] J. Park, M. Chen, and P. Yu. Using a hash-based method with transaction trimming for mining association rules. *IEEE Trans. on Knowledge and Data Engineering*, 9(5):813–825, Oct 1999.

[48] S.-T. Park, D.M. Pennock, C.L. Giles, and R. Krovetz. Analysis of lexical signatures for improving information presence on the world wide web. *ACM Transactions on Information Systems*, 22(4):540–572, 2004.

[49] Young-Hoon Park and Peter Fader. Modeling browsing behavior at multiple websites.

[50] D. Pavlov, E. Manavoglu, C.L. Giles, and D.M. Pennock. Collaborative filtering with maximum entropy. *IEEE Intelligent Systems*, 19(6):40–48, 2004.

[51] J. Pei, J. Han, and L. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proc. 2001 Int. Conf. on Data Engineering (ICDE'01)*. Heidelberg, Germany, Apr 2001.

[52] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining access pattern efficiently from web logs. Kyoto, Japan, apr 2000. Proc. 2000 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'00).

[53] Mike Perkowitz and Oren Etzioni. Adaptive sites: Automatically learning from user access patterns. Technical Report UW-CSE-97-03-01, University of Washington, 1997.

[54] Mike Perkowitz and Oren Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence*, 118(1-2), 2000.

[55] P. Pirolli and K. Card. Information foraging. *Psychological Review*, 106(4):643–675, 1999.

[56] P. Pirolli and W. Fu. Snif-act: A model of information foraging on the world wide web. In *Ninth International Conference on User Modeling*, Johnstown, PA, 2003.

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

[58] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, 1992.

[59] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[60] Philip Resnik. Selectional preference and sense disambiguation. pages 189–196, Washington, D.C., USA, Apr 1997.

[61] C. Rijsbergen. *Information Retrieval*. 2nd edition, London, Butterworths, 1979.

[62] http://www.cs.cmu.edu/ wcohen/.

[63] E. Omiecinski Savasere and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21th Very Large Data Base Conference*, pages 432–443, Zurich, Switzerland, Sep 1995.

[64] Robert Schapire. Theoretical views of boosting and applications. In *Tenth International Conference on Algorithmic Learning Theory*, 1999.

[65] Myra Spiliopoulou, C. Lukas Faulstich, and Karsten Winkler. A data miner analyzing the navigational behaviour of web users. In *Proc. of the Workshop on Machine Learning in User Modelling of the ACAI'99 Int. Conf.*, Creta, Greece, jul 1999. Springer Verlag.

[66] Myra Spiliopoulou and Lukas Faulstich. Wum: A tool for web utilization analysis. Valencia, Spain, mar 1998. EDBT Workshop WebDB'98, Springer Verlag.

[67] R. Sun and C.L. Giles. Sequence learning: From recognition and prediction to sequential decision making. *IEEE Intelligent Systems*, 16(4):67–70, 2001.

[68] http://svmlight.joachims.org/.

[69] W3C. Html 4.01 specification.

[70] K. Wang, Y. He, and J. Han. Mining frequent itemsets using support constraints. In *Proc. 2000 Int. Conf. on on Very Large Data Bases (VLDB'00)*, Cairo, Egypt, Sep 2000.

[71] Stephan Weibelzahl and Gerhard Weber. Evaluating the inference mechanism of adaptive learning systems. In *Proceedings of User Modeling(2003)*, pages 154–162, Johnstown, USA, June 2003.

[72] Ian Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, Oct 1999.

[73] Li Xiaobin, Stan Matwin, and Stan Szpakowicz. A wordnet-based algorithm for word sense disambiguation. In *Proceedings of IJCAI-95*, Montral, Canada, 1995.

[74] Qiang Yang, Henry Hanning Zhang, and Ian Tianyi Li. Mining web logs for prediction models in www caching and prefetching. In *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'01(Industry Applications Track)*, San Francisco, California, USA, aug 2001.

[75] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196. Cambridge, MA, 1995.

[76] Mary Zajicek and Chris Powell. Building a conceptual model of the world wide web for visually impaired users. In *Proc. Ergonomics Society 1997 Annual Conference*, Grantham, 1997.

[77] Tingshao Zhu, Russ Greiner, and Gerald Häubl. Predicting web information content. In *IJCAI-03 Workshop on Intelligent Techniques for Web Personalization*, pages 1–7, Acapulco, Mexico.

[78] Tingshao Zhu, Russ Greiner, and Gerald Häubl. An effective complete-web recommender system. In *The Twelfth International World Wide Web Conference(WWW2003)*, Budapest, HUNGARY, May 2003.

[79] Tingshao Zhu, Russ Greiner, and Gerald Häubl. Learning a model of a web user's interests. In *The 9th International Conference on User Modeling(UM2003)*, Johnstown, USA, June 2003.

[80] Tingshao Zhu, Russ Greiner, Gerald Häubl, Kevin Jewell, and Bob Price. Goal-directed site-independent recommendations from passive observations. In *The Twentieth National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, Pennsylvania, July 2005.

[81] Tingshao Zhu, Russ Greiner, Gerald Häubl, Kevin Jewell, and Bob Price. Off-line evaluation of recommendation functions. In *The 10th International Conference on User Modeling(UM2005)*, Edinburgh, Scotland, July 2005.

[82] Tingshao Zhu, Russ Greiner, Gerald Häubl, Kevin Jewell, and Bob Price. Using learned browsing behavior models to recommend relevant web pages. In *Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, August 2005.

Appendix A

Statistical Tests

Non-parametric tests are often used when we want to determine statistically significant difference but with fewer and weaker underlying assumptions than those associated with parametric tests. We employed two non-parametric tests to evaluate the quality of our functions : Mann-Whitney and Wisconsin.

Mann-Whitney Test The Mann-Whitney test can be used to analyse data from a two-group independent design (Experimental vs Control). The null hypothesis H_0 assumes that the two sets are from the same population; and therefore, there will be no siginificant difference between them. The alternative hypothesis states that the two sets do differ significantly, and it can also specify the direction of the difference (*i.e.*, Experimental Group is systematically higher or lower than Control Group).

Wilcoxon Signed-Ranks Test The Wilcoxon signed-ranks test is another non-parametric test that can be used for 2 repeated measures. It takes into account both the magnitude and the direction of the difference.

Appendix B

Browsing Features

We consider all words that appear in a session $\mathcal{S}$, removing stop words and stemming (using standard algorithms [57]), then compute several browsing features for each word w_i. In all cases, if the URL refers to a frame page, we calculate all the following measures based on the page view. † denotes that the browsing feature has been used in travel study in Section 4.1.2, and ‡ shows that the feature used in Section 5.1.1.

isKeywordCnt†‡
Number of times that w appeared within the query's keyword list.

skippedTitleCnt†‡
Number of skipped titles containing w.

skippedSnippetCnt†‡
Number of skipped snippets that contain w.

chosenTitleCnt†‡
Number of chosen titles that include w.

chosenSnippetCnt†‡
Number of chosen snippets that include w.

untouchedTitleCnt†‡
Number of untouched titles that include w.

untouchedSnippetCnt†‡
Number of untouched snippets that include w.

unknownCnt†‡
Number of times that w appears in the anchor of a chosen link that is not one of the listed results — *e.g.*, when the user clicks the hyperlink in the advertisement area.

bkTitleCnt†‡
Number of chosen titles that include w, but where the user later goes back to the same search result page, presumably to try another entry there.

bkSnippetCnt†‡
Number of chosen snippets that include w but were later "backed".

latestAppearance‡
the relative position of the latest page that contains w.

relativeFreq‡
In $\mathcal{S}$, compute the ratio of the number of occurrences of w to the number of the occurrences of all the words in $\mathcal{S}$.

ratioOccurences‡
the ratio of the number of pages contain w to the length of $\mathcal{S}$.

seqTFIDFWeight‡
(the absolute number of occurrences of w) $\times$ w's IDF.

ratioWordAppearance†‡
Number of occurrences of w divided by the number of all words.

avWeight†
Average weight of w across the whole session.

varWeight†
w's weight variation across the whole sequence.

avTFIDFWeight‡
Average TF/IDF weight of w in $\mathcal{S}$.

varTFIDFWeight‡
w's TF/IDF weight variation in $\mathcal{S}$.

avJewellWeight‡
Average Jewell weight (Chapter 5.1.4) of w in $\mathcal{S}$.

varJewellWeight‡
w's Jewell weight variation in $\mathcal{S}$.

trendWeight†
The trend of the word's weight in the whole sequence: { *ascend, descend, unchanged* }. If the word's weight becomes higher along the session, it is expected to be ICPageWord with high probability.

trendTFIDFWeight‡
The trend of w's TFIDF weight in $\mathcal{S}$: { *ascend, descend, unchanged* }. If the word's weight becomes higher along $\mathcal{S}$, it is expected to be ICPageWord with high probability.

trendJewellWeight‡
The trend of w's Jewell weight in $\mathcal{S}$: { *ascend, descend, unchanged* }.

ratioLinkFollow†‡
For the hyperlinks whose anchor text contain w,
$\text{ratioLinkFollow}(w) = \frac{\text{followed hyperlinks whose anchor text contain } w}{\text{hyperlinks whose anchor text contain } w}$.

ratioFollow†‡
How often w appeared in the anchor text of hyperlinks that were followed : $\text{ratioFollow}(w) = \frac{\text{number of followed hyperlinks whose anchor text contain } w}{\text{length of } \mathcal{S} - 1}$.

ratioLinkBack†‡
For the clicked hyperlinks whose anchor text contain w:
$\text{ratioLinkBack}(w) = \frac{\text{number of hyperlinks that were backed later}}{\text{number of hyperlinks followed}}$.

ratioBackward†‡
For these pages that contain w, $\text{ratioBackward}(w) = \frac{\text{number of pages that are revisited}}{\text{number of pages}}$.

avWeightBackward†
The average weight of w in the backward pages.

varWeightBackward†
The variance of w's weight in the backward pages.

avTFIDFWeightBackward‡
The average TFIDF weight of w in the backward pages.

varTFIDFWeightBackward‡
The variance of w's TFIDF weight in the backward pages.

avJewellWeightBackward‡
The average Jewell weight of w in the backward pages.

varJewellWeightBackward‡
The variance of w's Jewell weight in the backward pages.

ratioForward†‡
For the pages that contain w, $\text{ratioForward}(w) = \frac{\text{number of pages that are forward}}{\text{number of pages}}$.

avWeightForward†
The average weight of w in the forward pages.

varWeightForward†
The variance of w's weight in the forward pages.

avTFIDFWeightForward‡
The average TFIDF weight of w in the forward pages.

varTFIDFWeightForward‡
The variance of w's TFIDF weight in the forward pages.

avJewellWeightForward‡
The average Jewell weight of w in the forward pages.

varJewellWeightForward‡
The variance of w's Jewell weight in the forward pages.

ratioInTitle†‡

For those pages that contain w,

$$\text{ratioInTitle}(w) = \frac{\text{number of pages that contain w in the title}}{\text{number of such pages}}.$$

ratioInvisible†‡

For these pages that contain w, $\text{ratioInvisible}(w) = \frac{\text{number of pages where w is invisible}}{\text{number of pages}}$.

We only count the words in META tags (keyword & description) as invisible.[1]

[1] We thought "*ratioInvisible*" might be very important, as many websites ensure that all the relevant words appear in the META elements of the page, as a way to help establish a good position in search engine results pages.

Appendix C

One Sample ICPageWord Model and Recommendation

Figure C.1 is part of the decision tree of ICPageWord model that has been used in LILAC study.

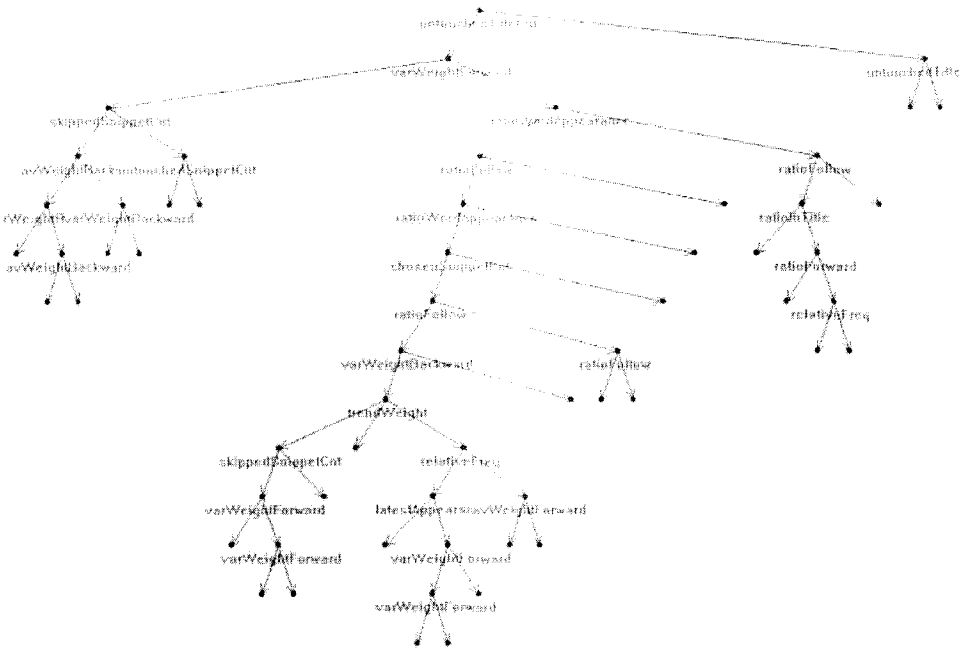

Figure C.1: One sample ICPageWord decision tree

By using the decision tree in Figure C.1, we recorded the page sequence of one subject's browsing as follows.

```
http://www.google.ca/search?q=analytical+coverage+regular+placement+optimal
http://www.it.cityu.edu.hk/~clliu/
http://www.google.ca/search?q=analytical+coverage+regular+placement+optimal
http://www.sheridanprinting.com/typedept/dac1.htm
http://www.google.ca/search?q=analytical+coverage+regular+placement+optimal
http://tcad.ece.orst.edu/list03.html
http://www.google.ca/search?q=analytical+coverage+regular+placement+optimal
http://bikmrdc.lm.fju.edu.tw/eee04/Long-accept-list.htm
```

After the subject visited the above page sequence, s/he clicked the "Suggest" button to request a recommendation. WebIC observed the above page sequence, and predicted the following query:

placement+optimal+regular+paper

It then sent the query to Google, and took the top returned page as the recommendation. The subject rated the suggested page as

"*Fully* answered my question".