Goal-Directed Site-Independent Recommendations from Passive Observations

Tingshao Zhu and **Russ Greiner**
Dept. of Computing Science
University of Alberta
Canada T6G 2E8

Gerald Häubl
School of Business
University of Alberta
Canada T6G 2R6

Kevin Jewell and **Bob Price**
Dept. of Computing Science
University of Alberta
Canada T6G 2E8

Abstract

This paper introduces a novel method to find Web pages that satisfy the user's current information need. The method infers the user's need from the content of the pages the user has visited and the actions the user has applied to these pages. Unlike content-based systems that attempt to learn a user's *long-term interests*, our system learns user-independent patterns of behavior that identify the user's *current* information need, based on his/her current browsing session, then uses this information to suggest specific pages intended to address this need. Our system learns these behavior patterns from labeled data collected during a five-week user study, involving over one hundred participants working on their day-to-day tasks. We tested this learned model in a second phase of this same study, and found that this model can effectively identify the information needs of new users as they browse previously unseen pages, and that we can use this information to help them find relevant pages.

1 Introduction

While the World Wide Web contains a vast quantity of information, it is often difficult for web users to find the information they are seeking. Presently, the most common way of accessing web resources is to use a web-browser to access an information portal or search engine. While such techniques have been quite helpful, they still require a user to provide explicit input: Navigation of a portal requires the user to interpret hyperlink anchor text and to actively follow links in the hope that they will lead to relevant pages. Similarly, search engines require users to explicitly enter a list of keywords that they hope will return pages with the information they are seeking and also to select one or more of the returned list of possible hits. Ubiquitous or embedded technologies would free the user from having to take the initiative. The user could simply concentrate on the task at hand, and passively observing agents would suggest relevant material when requested.

In order to make relevant suggestions, the observing agents must be able to infer the user's needs. Common approaches to learning a user's needs begin by analyzing the user's previous (inferred) interests, in order to discern his/her long term information needs. Of course, these systems are able to address the user's current information needs only if these needs correspond to long term trends. While this may be helpful for some tasks, such as filtering email [LK97] and news articles [BP99], it is problematic for our task, of helping the user browse the Web. Here, in particular, the user's needs will often change dramatically as s/he works on various tasks and subtasks over time. The needs associated with these tasks will often require information on diverse and unrelated topics, including topics the user has never investigated before. These observations motivate us to explore how a recommender could identify a user's current interests using only observations of his/her most recent activity.

Our earlier work [ZGH03b] provided a novel approach that extracts information needs based only the user's most recent activity. In that model, the most recent browsing activity is characterized by a set of "browsing features", which describe how the words are used in the current session. We were able to use the values of these browsing features to identify a set of words that characterize the user's *current* information need [ZGH03a].

We assume that the user's goal is to find web pages that address his/her current information need; we refer to each such page as an "information content" page or IC-page. Our earlier goal was simply to use browsing features, calculated from the user's current session, to predict the words that would appear within these IC-pages. This paper will focus on the more useful task of actually *finding the relevant pages themselves*. We do this by first training a model to identify a *search query* that can locate an IC-page that is relevant to the user's current session, and then passing that query to a search engine (here, Google).

The technique developed in this paper, like our earlier system, has two important properties:

1. It uses only passive observations of the user's current browsing activity to infer their information need. That is, the user is not required to provide any additional input, beyond his/her normal browsing actions.[1]
2. It can use any accessible site on the web to provide pages that satisfy this need. (This is different from site-specific

[1] This paper also describes a training phase, where paid subjects also provided auxiliary information. Note this will not be required of future users of this system.

recommendation systems, that can only refer to pages within this single website.)

The next section provides an overview of how our recommendation system will work, given a learned model. Section 3 then explains how we learned this model. Section 4 describes a large scale field evaluation of this system. Finally, we situate our work in the literature, mention some future directions and conclude.

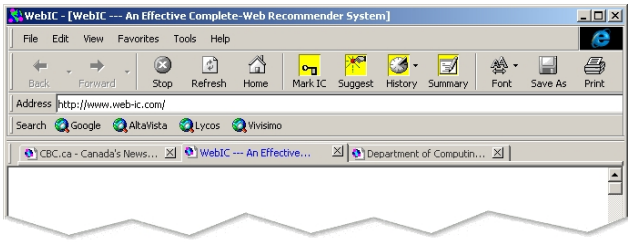

Figure 1: *W*ebIC — An Effective Complete-Web Recommender System

2 WebIC and the Recommendation Model

*W*ebIC, whose interface is shown in Figure 1, can be viewed as a web-browser, which includes a client-side Web recommender system. As the user is browsing, s/he has the option of clicking the "Suggest" button to ask *W*ebIC to recommend a Web page. *W*ebIC will then use information from the current session to recommend a page, from anywhere on the Web, that it predicts the user will find useful.

This section presents an overview of how *W*ebIC decides which pages to recommend. We assume that the user's current browsing activity is driven by a single information need; we divide his/her browsing activities into *sessions*, each of which is assumed to be related to a single need. We also assume that the user's information need can be satisfied by the content provided by a single page, which we call an *information content page* or IC-page. We attempt to relate the user's information need to a small number of abstract features, which correspond to how the user reacted to the words s/he saw in the current session. Our goal is to recommend web pages with relevant content, using only the information available in these browsing features. The remainder of this section explains these steps in more detail; see Figures 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 are used to isolate the tail end of the sequence of pages and actions that specifically relate to the user's current information need. This forms the "browsing session" of Figure 2A.

Step B: Assigning Features

We next convert the browsing session into abstract browsing features. This step has two dimensions. The first dimension

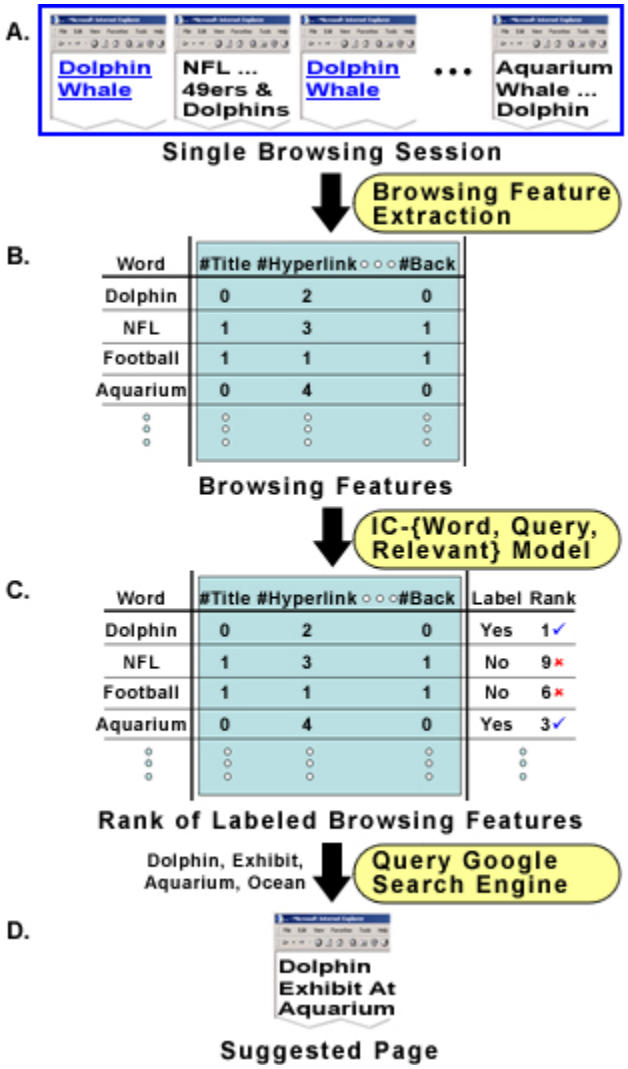

Figure 2: WebIC System at Performance Time

is to represent the content of the pages in the browsing sequence. We assume that this content can be coarsely approximated by the set of independent words found on the pages in the current session. To simplify processing and improve generalization, we remove stopwords and stem the remaining words [Por80]. We refine our representation of content by annotating each of the words in the session with features that describe the role the word played in the session pages. For instance, a word can be annotated with features such as "appears in title" or a "relative frequency of appearance in this session". These features allow us to represent the fact that some words are more representative of the content of pages in the session than others.

Second, we use the abstract browsing features to represent those aspects of a user's actions that help us understand their information needs. We interpret the user's actions as signals communicating the user's attitude towards the content of the pages encountered. Since we represent this content as individual words, these actions will tell us about the degree to which users feel specific words are representative of their information needs. For example, following a link on a page implies that the user interpreted the words in the hyperlink as predictive of useful content. We therefore also annotate words with action features such as "number of times this word appeared in a followed link anchor".

We used 35 distinct browsing features; see `www.web-ic.com/feature.txt`. While most of these features are straight-forward, a few of these features merit further discussion. We implemented some features specifically for processing the web pages that are results pages returned from search engines. Since these pages list links ostensibly related to the user's expressed interests in top down linear order, we can infer additional information from the user's behavior in this case. When the user skips over items in this linear list, we might infer that the user believed this anchor text was not representative of the user's information need. For words appearing on search result pages, we therefore include features such as "in a skipped-over hyperlink anchor".

As shown in Figure 2B, this step assigns a value for each browsing feature to every word in the session.

Step C: Interpreting Features

The words in the browsing sessions and their associated feature values abstractly represent the content of the session and the users actions on this content, but do not give us a succinct representation of the user's information need. For instance, many of these words may be common English words that are not particularly relevant to the user's need. A classifier is used to decide, for each annotated word, whether the word is likely to represent information content; the next section describes how we learn this classifier. This identifies a small subset of words likely to be relevant; see Figure 2C.

Step D: Recommending Pages

The final step is to recommend pages. In principle, a web crawler could be used to crawl forward from the user's current location to find pages with relevant content (*i.e.,* containing IC-words). In our current *W*ebIC system, however, a query formulator is used to turn the set of IC-words into a much shorter ordered list of words which can be sent as a query to a commercial search engine. The query formulator also makes use of weights whose source will be explained in the next section. As shown in Figure 2D, the top ranked page returned in the list of results from the search engine is then presented to the user as a recommendation.

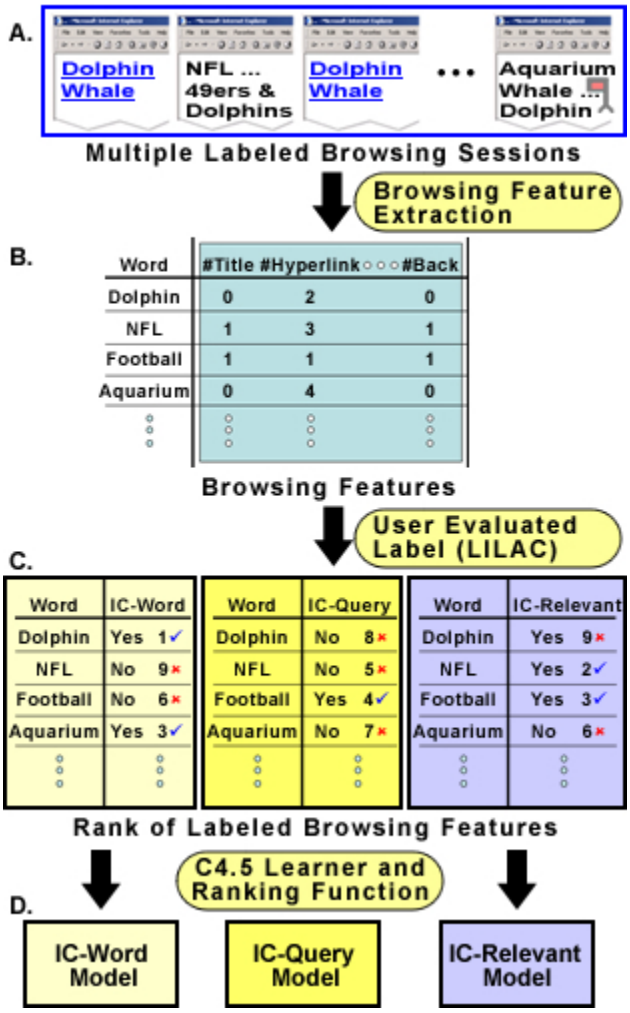

Figure 3: Training the IC-Models

3 Learning a Browsing Behavior Model

The previous section discussed how *W*ebIC identifies pages to suggest to the user. Step C of that process requires a classifier that returns the subset of words to send to a search engine. This classifier has two parts: a binary filter, to remove most of the words, and a ranker, to rank the remaining words. The classifier then returns the 4 highest-ranked words that pass the filter. This section describes how we learn both the filter and the ranker from training data. Figure 3A–D summarizes the process.

Step A: Labeled Browsing Sessions

We begin with traces of how many different users used the Web; see Section 4. As before, we divided these pages into sessions. We also asked each user to explicitly identify which of these pages were IC-pages, and also to explicitly specify whether certain provided words were relevant to his/her information need or not. *(Recall this is just to collect relevant data; we will not require typical users to provide this input.)*

Step B: Browsing Features

Browsing features are calculated as in the previous section. As before, each row of the "matrix" again corresponds to a word and each column, to a browsing feature. Here, however, these words are from many different sessions, from a number of different users.

Step C: Providing Labels for the Words

Our overall challenge is identifying which words should serve as keywords for a search engine, to help it identify an IC-page. We will attempt to learn this based on the words we have collected. To do this, we need to provide a label for each of these words. Below are 3 possible ways to assign these labels, corresponding to three models:

1. IC-word: In Zhu et al.'s original work, IC-words were defined as those words that actually appeared on pages the user marked as IC-pages.
2. IC-Relevant: In this model, a word reflects a user's information need if the user has explicitly marked it as relevant, after a browsing session.
3. IC-Query: In this model, a word reflects a user's information need if it can be used as a query to retrieve a page the user marked as useful (i.e., an IC-page). The user will not know if a word would be useful in a query. We describe below the process we use to infer if a word is IC-Query indirectly from user labels below.

As shown in Figure 3C, each IC-x model specifies a label for each word. This produces 3 sets of labeled data — which have the (browsing) features, but different labels.

We use the "yes/no" value of one set of words to train a classifier, and a ranker as well. Thus each model includes a classifier and a ranker, they are both trained on the same data set.

Step D: Training the Actual Classifier

The predictive score of each word w in a Web page p_i can also be considered as a special kind of weight for w, which describes how likely w can contribute to the locating of p_i by querying a search engine.

Whenever the system needs to predict a new page, it will first extract the browsing features of each of these words in the current browsing session, then run one of these classifiers to select a subset of these words. Consequently, (1) IC-word and IC-Relevant may identify hundreds of such significant words; far too many to submit to any search engine, and (2) we need to send a sequence of words, not a set. For these reasons, we also learn three sets of weights (one for each model), which the system will use 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 N 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^N be the words with the N highest $score_\alpha(\cdot)$ values, and then let

$$precision_\alpha \;=\; \frac{|\{w \in W_N \,|\, \text{class}(w) = significant\}|}{N}$$

be the fraction of W_N that are labeled as significant. We set α to optimize this $precision_\alpha$ score. Note that we will find different sets of α weights for each IC-model.

When the user requests a recommendation in the release version, we will first use the decision-tree classifier to filter away most of the words, then run this linear function to rank the remaining words (we need both part, as the filter may remove some highly-ranked words). We will then send the top $m = 4$ words to the search engine, in that order.

3.1 Notes on Training IC-Query

The IC-Query model requires us to label each word based on whether it, as a search engine keyword, would retrieve the relevant page. Unfortunately, users cannot supply this label (as they did for IC-word and IC-Relevant). We could attempt to directly calculate which words make good query words by, perhaps, choosing all possible k-element subsets of the words on a page, submitting them to the search engine and seeing which subsets actually return the original page. Unfortunately, this process is extremely slow and not practical.

To produce the desired labels, we independently train a function to predict whether a word described by certain page features will make a good query word for retrieving this page from a search engine. Page features are a subset of our browsing features that can be calculated from a single page. We have developed 19 distinct page features for each word w in page p, including: number of occurrences of w, normalized TFIDF of w, number of occurrences of w embedded within the following "HTML context" tags [W3C]: "h1", "h2", "h3", "h4", "h5", "h6", "a", "title", "cite", "strong", "big", "em", "i", "b", "u", "blink", and "s".

The good query word predictor could be used to implement an inverse search engine as follows: We would rank each word on the page using our likelihood of being a good query word function and select the top k. This process would turn a web page into a query.

We train this function as follows. We start with a large subset of web pages, indexed by the OpenDirectory chosen to represent a diverse set of possible domains and web page types. We then initialize the parameters of our linear query word predictor to random values. Next we use the function to assign a query to each page. We send each query to a search engine and see if it actually returns the page it was generated from. This gives us an accuracy score for the

function's current parameters over our entire sample of web pages. We then alter the parameters and try again. If the accuracy is improved, we keep the parameters. If not, we retain the previous best parameters. In this way, we perform a brute force search for parameters. The training process is slow and network intensive but results in a compact linear function that can be applied quickly in our main application described above: labeling words as good query words.

3.2 *W*ebIC's "MarkIC" Function

Most users will only use *W*ebIC's "Suggest" facility. However, whenever a user has found a page that satisfies his/her current information need, s/he also has the option of clicking the "MarkIC" button, to tell *W*ebIC that the current page is an IC-page. *W*ebIC can then use this information to produce a personalized recommendation system. *N.b.*, the "MarkIC" functionality is optional; in general, the user need not use this facility. In that case, *W*ebIC will simply use the generic model to recommend pages, based on our training data obtained from the LILAC study participants; see next section.

4 LILAC Study

The two goals of the LILAC (Learn from the Internet: Log, Annotation, Content) study were to gather data, from people working on their day-to-day tasks, to train our browsing behavior models, and to evaluate these models.

A total of 104 subjects participated in the five-week LILAC study, from both Canada (97) and USA (7); 47% of participants were female and 53% were male, over a range of ages (everyone was at least 18, and most were between 21-25).

LILAC considered four models: the three IC-models discussed above — IC-word, IC-Relevant, and IC-Query— and "Followed Hyperlink Word" (FHW), which is used as a baseline. Basically, FHW collects the words found in the anchor text of the followed hyperlinks in the page sequence. Note there is no training involved with this model. This is similar to the "Inferring User Need by Information Scent (IUNIS)" model [CPCP01].

We used the data collected during the study period to retrain each of our IC-models. That is, the users initially used the IC-word$_0$, IC-Relevant$_0$ and IC-Query$_0$ models, which were based on data obtained prior to the study. For the 2nd week, participants used the IC-word$_1$, IC-Relevant$_1$ and IC-Query$_1$ models, based on the training data obtained from week 1, as well as the prior model. And so forth.

4.1 Modifications to *W*ebIC for the User Study

We modified *W*ebIC for the LILAC study. Here, whenever the user requests a recommendation by clicking the "Suggest" button, *W*ebIC will select one of the 4 models randomly to generate a recommendation page. As one of the goals of the LILAC study is to evaluate our various models, this version of *W*ebIC therefore asked the user to evaluate this proposed page, as described below.

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, *W*ebIC will recommend an alternative web page as if the user had clicked the "Suggest" button just before reaching this IC-page. (Hence it will use the first $n-1$ pages of the session, excluding the current IC-page.) Once again, *W*ebIC will then ask the user to evaluate this recommended page.

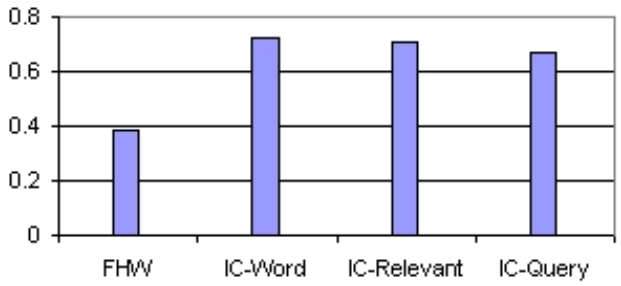

Figure 4: How often the User rated a Recommended Page as "Related", after clicking the "Suggest" button

As part of this evaluation, the user is instructed to "Tell us what you feel about the suggested page", to indicate whether the information provided on the page suggested by *W*ebIC was relevant to his/her search task. 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".

In addition, the user was asked to select informative "Descriptive Keywords" from a short list of words that *W*ebIC predicted as relevant. The information collected here will be used to train the IC-Relevant model.

4.2 Overall Results

The 104 LILAC subjects visited 93,443 Web pages, clicked "MarkIC" 2977 times and asked for recommendations by clicking the "Suggest" button 2531 times. As the rational for selecting these two conditions are significantly different, we analyzed the evaluation results for "Suggest" and "MarkIC" separately.

Figure 4 indicates how often the user considered the page recommended as a result of clicking the "Suggest" button to be *related* to the current search task (which is calculated as the sum of the four *related* categories described above). Clearly, each of our 3 IC-models generated a higher percentage of relevant pages for a user as compared to the baseline model — each was over 65%, versus the 38% for FHW.

Figure 5 shows the evaluation results for the recommended pages after the user clicked the "MarkIC" button.

Again, these results demonstrate that the three IC-models work better than FHW. The scores for IC-word and IC-Relevant remained roughly similar in value as compared to the "Suggest" case, while the IC-Query model increased from 66% to 74%, which is slightly better than the other two IC-models in the case of "MarkIC". We observed that IC-Query can achieve an overall performance comparable to

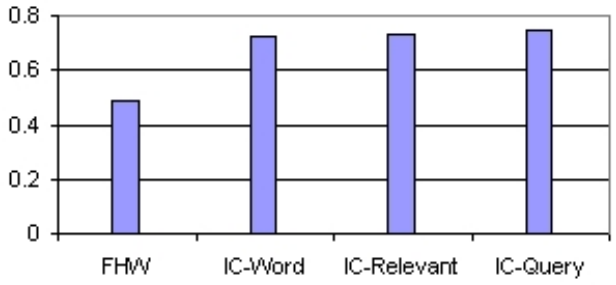

Figure 5: How often the User rated a Recommended Page as "Related", after clicking the "MarkIC" button

IC-word and IC-Relevant. Indeed, given the significant imbalance in the training data for IC-Query (e.g., few IC-Query words in a browsing session compared to hundreds of IC-words), and the approximate nature of the IC-Query identification function (i.e., $Google^{-1}$) that IC-Query is based on, IC-Query is found to work fairly well.

We also observed that FHW increased by almost 10%. To explain these data, 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.

4.3 Alternate Training for IC-Query Model

In order to train our IC-word and IC-Relevant models, the study participants must actively label IC-pages and relevant words while browsing the Web; this is both inconvenient for the user, and unrealistic in a production version of the *W*ebIC product. We can partially avoid this problem for the IC-Query model, by *passively* training this IC-Query 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, which produces a page that we show to the user, who then evaluates that page. If we assume that the search engine (here Google) remains relatively consistent over time, we can infer the evaluation of the search query from the actual evaluation of the recommended page. Thus we can label each query as one of the five evaluation outcomes (i.e., "Fully", "Somewhat", "Interesting", "Remotely", and "Irrelevant"). From these results, we can extract only the queries that are evaluated as "Fully" as belonging to the IC-Query model.

That is, imagine one of our models produced the words W, which leads Google to return $Google(W) = p_1$. *W*ebIC shows p_1 to the user. If the user states that this page "Fully answered my question", then this set W is labeled as a positive example; otherwise, W is labeled negatively.

The IC-Query models derived in the first four weeks of the LILAC study were trained based on IC-Query words. In the fifth week we changed the experimental protocol to train the IC-Query model based on all queries that resulted in a "Fully" evaluation in the previous weeks data. The evaluation results of the two training methods are presented in Figure 6.

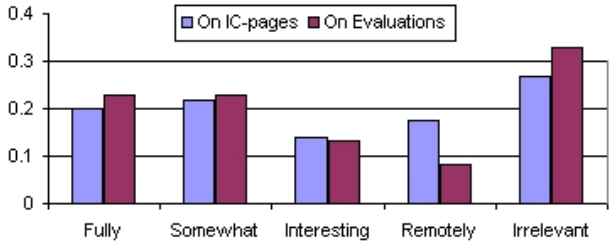

Figure 6: Training IC-Query Models

This figure suggests that the results of this approach are similar to the results of training the IC-Query model directly on the original IC-pages. This observation is significant as it will allow us to continuously refine the IC-Query model without requiring the user's annotation. This alternate training method will allow us to use *W*ebIC in more realistic real-world situations, as opposed to mainly research environments.

5 Related Work

Correlation-based recommenders [AS94; AS95] point users to pages that other users have visited, which are not necessarily the pages that these other users found useful nor will they necessarily useful to the current user. By contrast, note that *W*ebIC's models are explicitly trained to find useful IC-page. Content-based recommenders [BP99; JH93; AH02], which learn content models of a specific domain or set of sites, cannot reliably recommend pages for other sites. However, *W*ebIC's use of browsing patterns allow it to deal with any content and user, even from novel Web pages that involve novel words.

Pirolli and Fu [PF03] try to identify information need based on the *SNIF-ACT* model: production rules and a spreading activation network. These SNIF-ACT production rules resemble the patterns that we are attempting to learn. However, our IC-models differ by *learning* these rules by observing the user's actions together with the page's content; hence our systems do not rely on any prior knowledge of the words appearing on the pages. Our system also addresses the challenges of finding the relevant pages, from that set of words.

The *Cognitive Walkthrough for the Web* [BPKL02] requires the user to provide an explicit goal statement; it then uses this to predict which of the links the user will follow. Our models differ by (1) not requiring the user to provide an explicit goal, and (2) attempting to find pages anywhere on the web, as opposed to just the pages linked to the current page.

Choo et al. [CDT98] uses feedback from web users, as they performed ordinary tasks, to build a model of how web users seek information. They then use this model to categorize what the new user is doing (undirected viewing, conditioned viewing, informal search, or formal search). By contrast, our models use user information to learn a model for predicting a user's current information need based on

the current session; note our models are basically user-independent, if session-specific.

Spink et al. [SWJS01; JSS00] analyzed Web queries passed to the Excite search engine, and found several interesting characteristics of Web search behaviors — e.g., most searches involve very few search terms, few modified queries, and rarely use advanced search features, etc. This differs from our research as our focus is on finding useful information based on information gathered innocuously, rather than characterizing how users interact with search engines.

The Letizia [Lie95] agent helps a user browse the Web by using a best-first search augmented by heuristics to infer a user interest from browsing behavior. Watson [BH99] observed users interacting with everyday applications and then anticipated their information needs using heuristics to automatically form queries, which were sent to information retrieval systems (e.g., search engines) to get the related information for a user. The heuristics 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.

The earlier *W*ebIC publications [ZGH03b; ZGH03a] focused on ways to learn the browsing patterns, corresponding to the IC-word model. This paper significantly extends those earlier results by describing the challenges in finding the *useful pages* themselves. To do this, we introduce IC-Query for identifying query keywords, describe our current implementation in *W*ebIC, and present our findings of a recent user study, which evaluates the effectiveness of such models in a real-world environment.

6 Conclusion and Future Work

We are currently investigating other related techniques for generating useful recommendations. We plan to incorporate other browsing features, such as additional page content information. Each of our current models is basically user-independent; we are considering ways to train a personalized model, and then use it in combination with the generic ones. We plan to explore Natural Language processing systems to extend the range of our predicted relevant words, other Machine Learning algorithms (e.g., supervised sequence learning) to make better predictions, and the emerging Web technology such as semantic Web to get a better understanding of the context of arbitrary pages.

This paper explores the challenge of automatically finding pages, from anywhere on the Web, that address the user's current information need, without requiring the user to provide any explicit input. In particular, we investigated a general way to extract relevant information based only on the user's current web session: by finding browsing properties of the words that appear, then using a classifier to determine which of these words should be sent as keywords to a search engine. We propose two new models, IC-Query and IC-Relevant, to augment the prior IC-word model. We also conducted a large empirical study (LILAC), using our implementation of these ideas (*W*ebIC). Our results show that all three models are superior to a plausible alternative approach (FHW). Moreover, we provided a way for training IC-Query to obtain comparable performance without requiring as much annotation from the users, which will help us in producing an even more practical Web recommendation system in the future. Please visit `http://www.web-ic.com` for more information about the *W*ebIC system in general.

Acknowledgements

The authors gratefully acknowledges the generous support from Canada's Natural Science and Engineering Research Council, the Alberta Ingenuity Centre for Machine Learning (http://www.aicml.ca), and the Social Sciences and Humanities Research Council of Canada Initiative on the New Economy Research Alliances Program (SSHRC 538-2002-1013). We also thanks for the help from David Woloschuk, Tong Zheng and Xiaoyuan Su.

References

[AH02] 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.

[AS94] 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.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the Int'l Conference on Data Engineering (ICDE)*, Taipei, Taiwan, Mar 1995.

[BH99] 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.

[BP99] 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.

[BPKL02] 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.

[CDT98] 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.

[CPCP01] 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.

[JH93] 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.

[JSS00] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227, 2000.

[Lie95] H. Lieberman. Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, Montreal, Canada, Aug 1995.

[LK97] David Lewis and Kimberly Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

[PF03] 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.

[Por80] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, Jul 1980.

[SWJS01] A. Spink, D. Wolfram, B. Jansen, and T. Saracevic. Searching the web: The public and their queries. *Journal of the American Society of Information Science and Technology*, 52(3):226–234, 2001.

[W3C] W3C. Html 4.01 specification.

[ZGH03a] 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.

[ZGH03b] 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.