Learning a Model of a Web User's Interests

Tingshao Zhu[1], Russ Greiner[1] and Gerald Häubl[2]

[1] Dept. of Computing Science, University of Alberta, Canada T6G 2E1
{tszhu, greiner}@cs.ualberta.ca
[2] School of Business, University of Alberta, Canada T6G 2R6
Gerald.Haeubl@ualberta.ca

Abstract. There are many recommender systems that are designed to help users find relevant information on the web. To produce recommendations that are relevant to an individual user, many of these systems first attempt to learn a model of the user's browsing behavior. This paper presents a novel method for learning such a model from a set of annotated web logs — *i.e.*, web logs that are augmented with the user's assessment of whether each webpage is an *information content (IC)* page (*i.e.*, contains the information required to complete her task). Our systems use this to learn what properties of a webpage, within a sequence, identify such IC-pages, and similarly what "browsing properties" characterize the words on such pages ("IC-words"). As these methods deal with *properties* of webpages (or of words), rather than specific URLs (words), they can be used anywhere throughout the web; *i.e.*, they are not specific to a particular website, or a particular task. This paper also describes the enhanced browser, AIE, that we designed and implemented for collecting these annotated web logs, and an empirical study we conducted to investigate the effectiveness of our approach. This empirical evidence shows that our approach, and our algorithms, work effectively.

1 Introduction

The World Wide Web has become the largest information source for most people in the world. Unfortunately, it can be difficult for a web user to locate the information she[3] finds relevant. As different users will want different data, this paper provides a way to learn this user-specific relevance information, based on the user's current click stream, and a general user model. A browser that incorporates this learning facilty could adapt to the user, and enable her to find information that is relevant to her, more efficiently.

Our goal is a recommender system that can suggest *Information-Content (IC)* pages to the user — *i.e.*, the webpages that the user must examine to complete her task. Our *IcFinder* system (including the *IcURLPredictor* and *IcWordFinder* subroutines) uses a sample of "annotated web logs" (which label each page as IC or not; see Section 3) to learn a general model that characterizes how users seek relevant information. Given a new sequence of webpages visited by a user, this model will help identify which further pages this particular user will find useful. We also explore ways to use the annotated web logs from one user, or from a small cluster of users, to learn a recommender model specific to an individual user.

[3] We will use the female pronoun ("she", "her") when referring to users of either gender.

IcURLPredictor takes as input an annotated sequence of pages $\boldsymbol{U}^{\pm} = \{\langle U_1, \pm_1 \rangle, \ldots, \langle U_t, \pm_t \rangle\}$ and a target URL V, as well as a learned page-recommender model M_P, and predicts whether V is an IC-page — *i.e.*, if V contains information relevant to the user who has just visited $\boldsymbol{U}$. This M_P model examines properties of the webpages, both in the target V and sequence $\boldsymbol{U}^{\pm}$, such as the domain type, whether it followed a search engine, etc., to learn rules of the form

> any URL whose domain has been accessed more than 10 times,
> and whose depth is more than 3 (1)
> is likely to be an IC-page

Notice this rule is not about any specific page — *e.g.*, it is not about `http://www.cs.ualberta.ca` — but instead identifies a page based on the user's browsing patterns. Although this system is fairly accurate, it does require the user to annotate the pages (in $\boldsymbol{U}^{\pm}$) while she is browsing.

IcWordFinder takes as input an *unannotated* sequence of visited webpages $\boldsymbol{U} = \langle U_1, \ldots, U_n \rangle$ as well as a learned word-recommender model M_W, and returns a list of IC-words — *i.e.*, words that are likely to appear within the IC-page associated with $\boldsymbol{U}$ (and that, presumably, reflect the user's information need). In particular, it considers every word w that appears in any U_i, then assigns "browsing properties" to this w based on how w appears within this session — *e.g.*, did w appear in the title of any webpage, did it ever appear as a query to a search engine, etc. *IcWordFinder* then uses M_W to classify each w, determining the chance that a word with these browsing properties will be an IC-word. Notice that this classifier bases its decisions on the browsing properties of a word, rather than on the word itself; *i.e.*, it might claim that

> any word that appears in at least two titles in the session,
> but was never part of a seach engine query, (2)
> is likely to be an IC-word

but it will not make claims about, say, the IC-word-ness of "moose".

The models used by *IcWordFinder* (resp., *IcURLPredictor*) were learned from annotated webpages. Hence different users, who visit different webpages and give different annotations, can produce different models, and so obtain very different classifiers.

The general *IcFinder* system can use these subroutines to help find the IC-pages associated with a sequence of (un)annotated webpages: For example, it could obtain the likely IC-words from *IcWordFinder*, send these word to a search engine (such as Google), and then recommend the top-ranked returned pages. Alternatively, *IcFinder* could start from the user's current page and "scout" out ahead: following every link in that page, then every link on each of those "1-step away" pages, and so forth to some depth, stopping when *IcURLPredictor* thinks that the visited page qualifies as an IC-page, or when the words in that scouted page match the *IcWordFinder* results.

Section 2 surveys other approaches to building recommender systems, and notes how they differ from our method. In particular, it explains why many of the standard recommender systems, such as collaborative filtering and association rules, are not applicable to our situation. Section 3 describes the tool that we developed (AIE) and the study we conducted to collect the labeled data required in our research. Section 4 describes our actual *IcURLPredictor* and *IcWordFinder* techniques. This section also discusses some of the challenges we faced (such as dealing with an imbalanced dataset),

and presents the empirical results of our algorithms, emphasizing their ability to produce effective user models. (For more information about the study we performed, the data we collected, and our analyses, please see [12].)

2 Related Work

There is a great deal of research on generating recommendations for web users. This section will summarize several related approaches, and discuss how they differ from our approach.

Zukerman [15] distinguishes two main approaches to learning which pages will appeal to a user: A *content-based* system tries to learn a model based on the contents of the webpage, while a *collaborative* system bases its model on finding "similar" users, and assuming the current user will like the pages that those similar users have visited.

Recall that our goal is to find pages that contain information the user needs to complete her task. This differs from the implicit goal of standard collaborative systems, which is to identify pages that other similar users have visited, as those visited pages may correspond simply to irrelevant pages on the paths that others have taken towards their various goals, or worse, simply to standard dead-ends that everyone seems to hit. (This is why we need to use explicitly annotated web logs; see below.)

Our systems are basically content-based, as their decisions are based on the information within each webpage. However, our approach differs from the *standard* content-based systems: As many such systems are restricted to a single website, their classifiers can be based on a limited range of words or URLs; this means they can make predictions about the importance of specific URLs (see association rules [1]), or of specific hard-selected words [3, 7, 2]. We did not want to restrict ourselves to a single website, but wanted a system that could recommend pages anywhere on the web, which could therefore involve an unrestricted range of words. For this reason, we built our content-based classifiers based on characteristics ("browsing properties") of the words, or of the URLs. (E.g., *IcURLPredictor* may learn patterns like Equation 1 and *IcWordFinder* may find rules like Equation 2.) Notice this means our system is not restricted to predefined words, nor to webpages that already have been visited by this user, nor even to webpages that have been visited by similar users. As these browsing properties will appear across different websites, we expect them to be useful even in novel web environments.

3 AIE and Empirical Study

3.1 Specific Task

Recall that our overall goal is to determine which pages are IC — *i.e.*, which contain the information required to complete a task. To do this, we collected a set of *annotated web-logs*; each being a sequence of webpages that a user has visited, where each page is labeled (by the user) with a bit that indicates whether she views this page as an IC-page. We collected data from 129 participants (undergraduate business students), asking each participant to perform the following task:

1. Identify 3 novel vacation destinations — *i.e.*, places not visited in the past
2. Plan a *detailed* vacation to each destination, choosing specific travel dates, flight numbers, accomodations (hotels, campsites, ...), activities, etc.

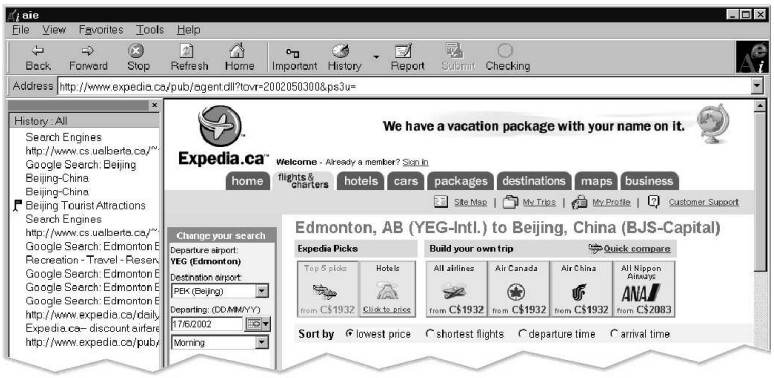

Fig. 1. AIE Browser

Subjects were given access to our augmented browsing tool (AIE; see Section 3.2), which recorded their specific web-logs, and required them to provide the "IC-page" annotation. The participants also had to produce a short report summarizing the vacation plans, and citing the specific important webpages that were involved in these decisions — these, of course, are IC-pages. To help motivate the participants to take this exercise seriously, they knew that two of them, selected at random, would win up to $500 towards one of the three specific vacations they were about to plan.

3.2 AIE: Annotation Internet Explorer

To help us collect the relevant information, we built an enhanced version of INTERNET EXPLORER, called AIE (shown in Figure 1) which we installed on all of the computers in the lab used in our study. As with all browsers, the user can see the current webpage. In addition, this tool incorporates several relevant extensions — see the toolbar across the top of Figure 1. First, the user can declare the current page to be an "IC-page", by clicking the *Important* button on the top bar. Second, the *History* button on the toolbar brings up the left side-panel (as shown in Figure 1), which shows the user the set of all pages seen so far, with a flag indicating which pages the user tagged as IC. The *Report* button will switch the browse view to the report editor, which subjects used to enter their reports.

After completing their reports, users submitted entire session using the *Submit* button. This recorded the entire sequence of web-sites visited, together with the user's IC-page annotations, as well as other information, such as time-stamps for all pages.

3.3 Some Aspects of the Web Log Data

The 129 study participants collectively requested 15,105 pages, involving 5,995 distinct URLs; hence each URL was requested 2.52 times on average. (If we disregard the 3,039 pages corresponding to 11 search engine pages, each of the remaining URLs was requested only 2.02 times on average.) We found that 82.39% of the URLs were visited only once or twice — hence very few URLs had strong support in this dataset. The subjects labeled 1,887 pages as IC, which corresponds to 14.63 IC-pages per participant.

4 Learning Task

As noted above, we use these annotated web logs to train our classifiers: *IcURLPredictor* for predicting which URLs will be IC-pages, and *IcWordFinder* to identify which words will be IC-words.[4] We first extracted certain "browsing features" of the URLs and the words from the raw data, and labeled each page (resp., word) to indicate whether it is an IC-page (resp., IC-word). We summarize these features in Sections 4.2 and 4.3.

To better describe the performance of our predictors, we computed precision and recall values for both prediction tasks, where "IC-Precision" is TrueIC/PredictedAsIC and "Recall" is TrueIC/AllRealIC; see [10]. We similarly define nonIC-Precision and nonIC-Recall for non-IC-pages.

4.1 Data Cleaning

Our system parses the log files to produce the sequence of pages that have been downloaded. Unfortunately some of these pages 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.), and had our algorithms ignore a URL if its domain name is in this list.

While we did record time information, we were unable to use it in the learning process. This is because many subjects switched modes (to "Report mode") on finding each IC-page, which means that much of the time between requesting an IC-page and the next page was not purely viewing time, but also includes the time spent writing this part of the report. Unfortunately, as we did not anticipate this behavior, we did not record the time spent in Report mode.

4.2 IC-page Prediction

An IC-page classifier tries to determine whether the current URL is an IC-page, based on the available information. That is, assume the user has visited the "annotated page sequence" $\langle\, (U_1, \pm_1),\ (U_2, \pm_2),\ (U_3, \pm_3),\ \ldots,\ (U_{t-1}, \pm_{t-1}),\ U_t \,\rangle$, where each $\pm_i$ is "$+$" if this page was deemed an IC-page and "$-$" otherwise. The challenge is to use this information (augmented with other data, see below) to determine whether U_t is an IC-page — *i.e.*, the value of $\pm_t$.

Our *IcURLPredictor* tries to learn this IC-page classifier: Given a number of such annotated web logs, learn a classifier M_P that can take an annotated page sequence as input, and determine whether the final page is IC or not.

Attributes Used: There are many attributes we could extract from the pages in the web sequence. We considered many of them, but after some preliminary analyses, we selected the 14 attributes shown below, which we compute during a preprocessing step during both training and testing.

Note that a "site-session" is the click stream within a single web domain — *i.e.*, a new site-session begins whenever the user enters a new website. The site-session is only a click stream segmentation method; we do not assume that each site-session concentrates on exactly one task.

[4] This is just for the training phase; *IcWordFinder* will later use *unannotated* web logs to make its predictions.

Table 1. Empirical Results for Predicting IC (nonIC) Pages

	IC-page		non IC-page	
	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
BNB	0.669 ± 0.048	0.70 ± 0.04	0.6861 ± 0.04	0.65 ± 0.07

1. *URL Properties of target webpage, U_t*
   This set consists of some general features of the current URL, including: **URL-type** (search engine, dynamic, static, etc.), **DomainType** (edu, com, net, org, gov, etc.), and **URL-depth** (Number of "/"s in the URL).

2. *User Click Stream $\langle U_1, U_2, \ldots, U_{t-1} \rangle$ vs U_t*
   FollowSearchEngine: Does U_t immediately follow some search engine?
   isLastEntry: Is U_t the last one in the site-session.
   inTotalNumberofPage: the number of pages that have been visited within this site-session.
   inTotalNumberofICPage: the number of pages, within this site-session, that have been labeled as IC
   inLastIC: the number of pages that have been visited since the last IC-page within this site-session.
   TotalNumberOfPages: the number of pages that have been visited.
   TotalNumberOfICPages: the number of IC-pages that have been visited.
   LastIC: the number of pages that have been visited since the last IC-page.
   PercentageDomain: the percentage of the pages that have the same domain as U_t
   PercentageIC: percentage of IC-pages
   PercentageSameDomainIC: percentage of IC-pages that have the same domain as U_t

Empirical Results for *IcURLPredictor*: As only 10% of the pages are IC, there is a trivial way to obtain 90% accuracy: simply returning "Not IC" on each instance. Of course, this will not serve our needs. To address this problem of "imbalanced data" [5], we used oversampling to generate testing and training data [6]: Form a 200-element testing set by randomly selecting (with replacement) 100 IC-pages and 100 non-IC-pages. (Notice some IC-pages may appear several times in a single training sample.)

After data preparation, we ran several classification algorithms on the data set, producing decision tree (C4.5) [9], NaïveBayes (NB) [4], and Boosted NaïveBayes (BNB) [11]. In all cases, we performed 10-fold cross validation. The results, averaged over all 10 CV folds, appear in Table 1 in the form of "mean±standard-deviation". Notice that BNB has the best "worst-case" over these 4 values, averaging around 65%.

4.3 IC-word Prediction: *IcWordFinder*

This section describes the *IcWordFinder* subroutine, which learns a classifier (M_W) that can predict whether a word will be an IC-word. Here too we first pre-processed the annotated web logs: We first segmented the user's entire click stream into *IC-sessions* — sequences of webpages that end with an IC-page; see below. For each IC-session, we then extract all the words in the pages except the last one (which is an IC-page),

compute various browsing features of each word, and label each word as an IC-word or not. *IcWordFinder* uses this information to train a classifier to predict when a word will be an IC-word.

IC-session Identification: An "IC-session" is a consecutive sequence of pages that ends with an IC-page, or the end of the user's entire session. While we expect that each IC-session pertains to a single information seeking task or one aspect of such a task, we never explicitly use this assumption.

To identify meaningful IC-sessions, we used the heuristic that if the page after an IC-page is a new search query, this indicates the start of a new session, since it is very common that when one task has been done, users will go to a search engine to begin the next task.

We defined each IC-session as composed of *pageviews*, where a *pageview* is what the user actually sees. In the case of frames, a pageview can be composed of a number of individual URLs. 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 were recorded also.

Attribute Extraction: We consider all words that appear in all pages, removing stop words and stemming [8]. We next compute the 25 attributes, for each remaining word, from each IC-session. (This section only sketches the attributes; see [12, 14].)

Search Query Category: As our data set includes many requests to search engines, we include several attributes that relate to the words in search-result pages. We consider only information produced by every search engine: the title (*i.e.*, the first line of the result) and the snippet (*i.e.*, the text below the title). We can tag each entry in each search-result page as one of the following: *Chosen*, *Skipped*, and *Untouched*. If the user follows a link, the words in its title and snippet will be considered "*Chosen*". The words that appear in the links that the user did not follow, before the last chosen one, will be deemed "*Skipped*", and all entries after the last chosen link will be considered "*Untouched*".

Most of these search query attributes are the number of times the word appears in each state — *e.g.*, how many times the word is in {*Skipped*, *Chosen*, *Untouched*} title/snippet, the number of times that it appears within the query's keyword list, etc.

Sequential Attributes: Each of the sequential features is extracted from each page in an IC-session, except search-result pages and the last destination page. If the URL refers to a frame page, we determine the features based on that pageview.

Many features are based on the "weight" of a word, which is basically the word's frequency in the page, but with addition weight based on the word's location and layout in the webpage — *e.g.*, if the word is in the title, is in bold, etc.

We say a hyperlink (in page U) is "*backed*" if the user followed that link to a page, but went back to page U later. A page is "*backward*" if that page has been visited before; otherwise we say a page is "*forward*"; and one URL may be a forward page at first, but later a backward page elsewhere in the sequence. (Note we record every URL requested, even if it has been accessed before.) Sequential attributes are mainly the features for the weights of the words in the IC-session, such as the average/variance of a word's weight in all the pages, backward pages, and forward pages, how many times the word is in the

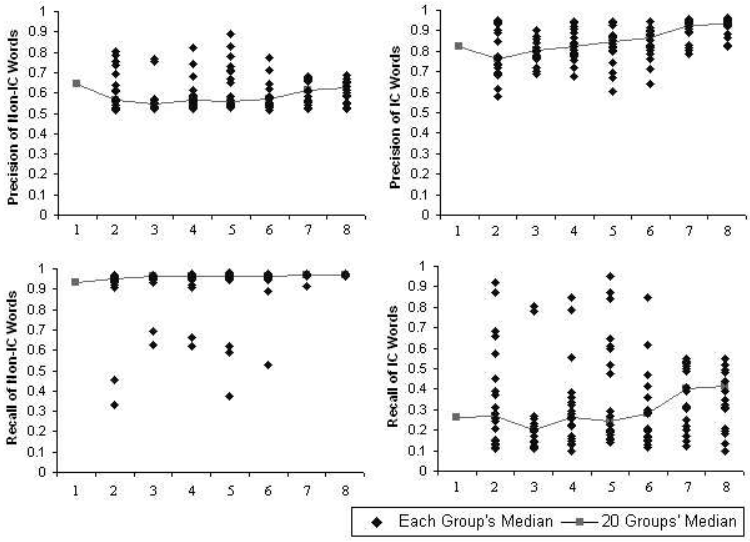

Fig. 2. IC-word Prediction: Testing Result

anchor text of the followed hyperlink, how many times the word is in the page's title, etc.

We compute these attributes for each word in an IC-session, and we also know whether it appears in the IC-page or not. When we train the classifier, we do not need the words themselves, just these attributes and whether the word appears in the IC-page.

Empirical Results for *IcWordFinder*: After preparing the data, our *IcWordFinder* used Weka [13] to produce a NaïveBayes (NB) classifier [4]. 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 $N \in \{1, 2, \ldots, 8\}$ subjects. For $N = 1$, we took each individual as a 1-user group; and for the other $N \in \{2, 3, \ldots, 8\}$, 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.

Here, we found an average accuracy of around $65-70\%$; the precision and recall results appear in Figure 2. Even though the average recall of IC-words is only about 45%, we anticipate this will be sufficient to to find IC-pages, which of course is our ultimate goal. Recall the two methods (from Section 1) for locating IC-pages given IC-words. *IcFinder* can scout ahead to find the IC-pages that match the predicted IC-words; knowing 45% of the words on that page makes it easy for the scout to correctly identify the page based on its content, or at least some pages very similar to it. Alternatively, *IcFinder* might try to build search queries using the predicted IC-words. Given the high

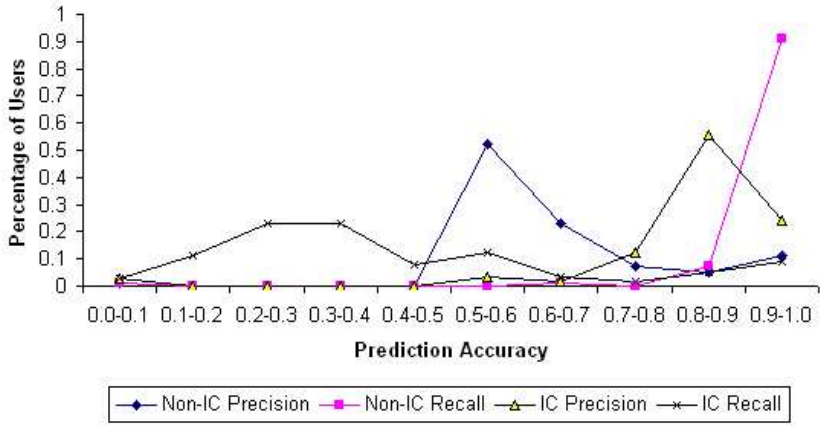

Fig. 3. Prediction result for individual users

precision of our IC-word prediction, even with recall around 45%, we can anticipate finding tens of words that will surely be in the IC-page. Since the predicted IC-words are exclusive of stop words, 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.

The single-user case ($N = 1$) 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 compute 10-fold training/testing; a histogram showing the median values for all 4 cases (Precision vs Recall; and IC vs non-IC) appears in Figure 3. The results indicate that some individual users are very predictable; note the very high precision and recall in many cases. This suggests that there is 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 individuals find useful information.

5 Conclusion and Future Work

Future Work: We collected annotated web data only for a single task — travel planning. As we suspect that other tasks may have different characteristics, we plan to conduct further empirical studies based on other tasks; *e.g.*, producing a research report by probing academic websites. We will also investigate ways to learn models for individual users, or small clusters of users, extending Figure 3. Here, we will explore ways to combine a general model, over all users, with the details relevant to an individual user. We will also experiment with yet other learning algorithms, as well as other techniques for dealing with imbalanced datasets.

Contributions: This paper provides two tools that can help web users find the information they need to complete their task: *IcURLPredictor*, which learns a classifier for identifying IC-pages, and *IcWordFinder*, which learns a classifier for identifying IC-words. Like many other webpage recommendation systems, our tools first learn a general user model from a large quantity of user data. As many of these alternative recommendation systems use specific pre-defined URLs or words, they are most effective when applied

in the same environment as the training scenario, but much less helpful in a new environment. By contrast, our tools first extract "browsing features" of the words or pages, then learn user-specific *characteristics* of words (or pages) that make them IC for the current user, based on these features. While the models we learn are general to a class of users, the performance systems then uses the actual webpages in the user's current session to produce recommendation specific to that person, in the current context, with the goal of retrieving the information that is most relevant to her. As our systems are also independent of any specific URL or word, these classifiers will be able to help users even in arbitrary novel websites.

To learn such browsing behavior, we first collected annotated web log data by conducting an empirical study, using a browsing tool, AIE, developed for this task. We then built our models M_P and M_W (used by *IcURLPredictor* and *IcWordFinder*) based on the observed browsing features: attributes of the visited URLs and of the words that appear in the annotated web log. The testing results show that our systems produce reasonably accurate predictions, in situations for which no other technique even applies.

6 Acknowledgments

The authors gratefully acknowledge 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).

References

1. 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.
2. Corin Anderson and Eric Horvitz. Web montage: A dynamic personalized start page. In *Proceedings of the 11th World Wide Web Conference (WWW 2002)*, 2002.
3. 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.
4. R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
5. Rob Holte, Nathalie Japkowicz, Charles Ling, and Stan Matwin. *AAAI'2000 Workshop on Learning from Imbalanced Data Sets*. AAAI Press, 2000.
6. N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI '2000)*, 2000.
7. 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.
8. M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, Jul 1980.
9. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, 1992.
10. C. Rijsbergen. *Information Retrieval*. 2nd edition, London, Butterworths, 1979.
11. Robert E. Schapire. Theoretical views of boosting and applications. In *Tenth International Conference on Algorithmic Learning Theory*, 1999.
12. http://www.cs.ualberta.ca/∼greiner/web-user-model.html.
13. Ian Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999.
14. T. Zhu, R. Greiner, and G. Häubl. An effective complete-web recommender system. In *Wide Web Conference(WWW2003)*, Budapest, 2003.
15. I. Zukerman and D. Albrecht. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, 11(1-2):5–18, 2001.