Intelligent Interaction: A Case Study of Web Page Prediction

Andrea Bačić and Andrina Granič*

Siemens d.d., Siemens IT Solutions and Services, Put brodarice 6, 21000 Split, Croatia
E-mail: andrea.bacic@siemens.com

*Faculty of Science, University of Split, Nikole Tesle 12, 21000 Split, Croatia
E-mail: andrina.granic@pmfst.hr

Abstract. In this paper preliminary work in the area of web page prediction is presented. The designed and implemented prototype offers personalized interaction by predicting the user’s behaviour from previous web browsing history. Those predictions are afterwards used to simplify the user’s future interactions. Rather simple and feasible prototype enhancements are offered and discussed. Its simplicity and effectiveness makes it potentially useful for widespread application.

Keywords. web browsing, web page prediction, personalized interaction

1. Introduction

Human-Computer Interaction (HCI) research places an individual as the focus of all theoretical and practical advances, stressing the importance to design technologies for human needs. Intelligent user interfaces (IUIs) have been recommended as a means of making systems individualized or personalized, thus enhancing the systems flexibility and attractiveness [2; 11]. Specific applications of intelligence to areas as diverse as intelligent hypermedia, recommender systems, intelligent filtering, explanation systems, intelligent help and intelligent tutoring could be identified. Thus, the intelligence in the interface of such a system can make the system adapt to the needs of different users, can take initiative and make suggestions to the user, can learn new concepts and techniques, can provide explanation of its actions or can predict the next action of the user cf. [3; 14].

The ability to predict the user’s next action allows the system to envisage the user’s needs and to adapt to and improve upon the user’s work, aiding the human-computer interaction process. In this paper we present preliminary work in the prediction of the user’s next requested web page. The designed and implemented prototype offers personalized interaction by predicting the user’s behaviour from her/his previous web browsing history and then uses these predictions to simplify her/his future interactions with the browser. The paper is organized as follows. In Section 2 we present a short outline of research in the area of user actions prediction. Section 3 is dedicated to describing used technology, the architecture of the system and the developed prototype for web page prediction. In Section 4 we provide discussion, suggestions for prototype improvement along with directions for further work.

2. Related research and motivation

Various systems able to predict the user’s next action are briefly presented in the following. Early work in command line prediction includes Greenberg’s (1988) work on dataset’s, still usable even today [9], Darragh, Witten and James’s (1990) Reactive Keyboard [5] and Greenberg, Darragh, Maulsby and Witten’s (1991) Predictive Interfaces [10]. More recently, Davison and Hirsh (1997) have modified the UNIX shell in order to memorize a command history and subsequently (by means of different methods) calculated the probability of a certain command input [6; 7]. Based on their work is the research of Korvemaker and Greiner (2000) who have studied the prediction of the next UNIX command applying diverse statistical methods [13]. In addition to the command input, they have also attempted to predict the command parameters. Jacobs and Blockeel’s (2003) have tried to improve Davison and Hirsh's command prediction by employing a time component [12]. That is, they assumed that the time of the command input should be relevant for next UNIX command prediction.

Several authors have based their research on predicting the next requested web site. Zukerman, Albrecht and Nicholson (1999) have addressed the ways for minimizing waiting for a requested web page by predicting which web page the user will ask for next [17]. Based on their work is the research of Friaz-Martinez and Karamchethi (2002), who have enhanced their predicting model.
with an important parameter – the time between two requests [8]. Supplementary work in the prediction of a sequence of user actions encompass prediction of the input of the next word or entire sentence in text editing [4; 15], prediction of motor actions based on eye movement [16] or prediction of the complete movement of the human body during the performance of some simple actions [1].

From the above discussion it is evident that predicting the sequences of the user’s next actions is a topic that has been explored from many angles, from diverse sciences and utilizing a variety of methods in an attempt to find the best solution. We have tried to provide a contribution to this challenging topic by taking a slightly different approach. To be precise, whether the case of prediction of the next UNIX command and specially the next web page is considered, the user can for example be on one side of the globe and the server with the prediction model on the other side. Therefore for the prediction of the next user action, the specific data about the user (for example Internet Protocol (IP) address, location and other user specific information) should be transferred via Internet. Such transfer could be time consuming and unreliable. For that reason, a fairly simple application installed on a user’s computer is developed. It merges a user’s profiled behaviour with an application logic that is located on the users’ personal computer. The design, the implementation and the basic workflow of the application is introduced in the following section.

3. Prototype for web page prediction

In order to provide intelligent services and personalized interaction, analysis of web server history has been used to model the behaviour of web users. Throughout browsing, users visit numerous web pages located on different servers and the prediction of their next requested web page has been pointed out as an important issue. Some studies have already addressed the prediction of a users sequence of action based on their behaviour on one server, see for example [8]. Nevertheless a user’s behaviour, when analysed and grouped, can be used for speeding up web browsing, for simplifying the usage of a web browser itself and for enabling personalized, intelligent interaction as well. This is the main reason why we have focused our work on the design of a fairly simple and small application that would be installed on a user’s computer. In that way the user data could be easily reached and her/his browsing history could be gathered from the browser integrated within the application.

Grouping of the users’ behaviour was accomplished according to a set of time periods. Five time intervals were defined and the history of the user’s browsing was saved according to the time frame in which it occurred. An initial percentage of the borderline frequency of a requested URL of a web page was set to 55%. Therefore, if the user had visited a particular web page in a certain time frame more than 55% of the time, instead of the user having to manually write the desired page address, the application automatically during start-up opens a new window with the predicted URL. Also the user is able to browse without any disturbance while the application is updating the browsing history.

Accordingly, the most important functionality of the designed and developed prototype is the automatic opening of a window with URLs of the web pages that satisfy the criterion of a “super request” (frequency of request is greater than 55% of total request frequency). Other than that, the application has a simple standardized graphical user interface, with login screen and a window with a list of the most frequent URLs in total and in the current time interval.

3.1. Used technologies

For the development of the application the following technologies were used:

- Java programming language – an object-oriented programming language with its biggest advantage being the fact that it is platform-independent (code is not dependent on the operating system on which the application will run);
- Eclipse software development platform – a multi-language software development platform comprising an IDE (Integrated Development Environment) and a plug-in system to extend it;
- MySQL Server for communication between database and application – a relational database management system which runs as a server providing multi-user access to a number of databases;
- MySQL Front – an application with simple user interface for easier handling of database tables and easier input of SQL (Structured Query Language) queries.

Main reason for choosing these technologies is their availability; they are free of charge for usage.
in non commercial software, like the one addressed in this paper.

3.2. Architecture of the application

Application architecture is presented in the following. First, Unified Modelling Language (UML) diagrams and a database schema are briefly introduced. Subsequently the methods that are “deciding on” which URL did cross the borderline frequency are explained. Finally workflow of the application is described.

3.2.1. UML diagrams

The application is composed of three interrelated parts: the presentation layer i.e. graphical user interface (GUI), the persistence layer i.e. database connection and the business logic layer. The main reason why we have chosen such architecture is its expendability. To be precise, if at some stage in the future there is a need for a change in the graphical part of the application, the main database connectivity along with business logic layer remain unaffected. On the other hand, if database connection has to be changed, graphical part of the application can remain identical. Such autonomy has been provided by dividing database classes and GUI classes in separate packages. Figure 1 shows an UML diagram of the database classes where the functionality and the dependencies between classes are emphasized. The DataCollector class is the “main” class in the db package; it communicates with other classes which retrieve data through SQL queries and generate objects that the application can use.

An UML diagram of the classes responsible for the graphical part of the application along with the main class Pska are shown in Figure 2. The class is located in a separate package; it communicates with the database classes and with the GUI classes. The GUI classes are organized in a way that each class creates its own window with its own functionality. According to the event that occurs, Pska opens a new window (instantiates next class) that has to become visible.

3.2.2. Database

For the needs of the application, a simple database was developed; the database consists of three tables. The first table named url holds information of the URLs that have been visited at any time by any user identified with a unique ID. The second table called users maintains information about the users. The table stores data concerning the identification of users (user ID, username and password) along with five counters that indicate how many times in a specific time frame a particular user has started the application. The third table named url_user is the table for joining data – one user (userID) is linked with one URL (urlID) and the counters of the time frame memorize how many times the user has requested an URL. In addition, the last date of a request for that particular URL is remembered in order to check if that web page has been called in the previous three months (otherwise it should be deleted).

Such structure of the application database enables calculation of the frequency of user’s requests for a particular web page URL in a specific time frame. The data in the tables are updated through SQL queries that originate from
the application itself, thus enabling real data collection along with updates of the database.

3.2.3. Key parts of the code

When a user logs in to the application, the method `getUltraPopular()` is called. The method returns an array of URLs that full-fill the requirement of crossing the borderline frequency and enables acquisition of the following four variables:

1. ID of the logged user,
2. list of all URLs that the logged in user has ever requested,
3. number of logins in the current time frame and
4. code of the current time frame

With this data the `isUltraPopular()` method is called for every URL (see Figure 3 to get insight of the related pseudo code). By calculating the frequency of requests, the method checks if the given URL does qualify for the automatic request of the URL. Namely, the number of user’s requests for the given URL in the current time frame is divided with the number of user’s logins in the same time frame. The result is then multiplied with one hundred, thus providing the percentage of the user’s request for the particular web page URL. If the percentage is greater than 55% (borderline frequency) the method returns the Boolean value `true`; otherwise the returned value is `false`.

In the case of a positive returned value of the `isUltraPopular()` method, the particular URL is added to the list of those which have to be automatically requested. Once all the web page URLs have been checked and the list filled with the ones that are frequent enough, application goes through the updated list and opens browsers with the web page URLs that have been filtered.

3.3. Workflow of the application

At the start-up of the application the login screen is shown. It enables the user to write his or her credentials or to select a button to open a window for creating a new user. Both windows have an “OK” button for confirming the input data; if the data is not correct, a popup window is shown to inform the user of her/his mistake.

After the successful login, the application checks if the logged user has the URLs that are requested with a frequency higher than 55%. If the result is positive, a browser is opened for each URL that satisfies that criteria (see Figure 4 on the next page). Additionally, a window with a list containing a maximum of ten most often requested URLs in general and ten most frequent URLs in a current time frame is provided (illustrated on Figure 3). Hence the user can chose to open an additional web page or just an empty browser. While collecting and storing data of a user’s behaviour, the application at the same time enables the user efficient and uninterrupted work.

![Figure 4. Window with most frequent URLs (in total on the left and in during the afternoon time frame on the right)](image)

4. Discussion and future work

This paper presents an interesting and potentially useful Internet browsing aid. The developed prototype discussed here is on its own distinctive. While the authors in the similar research field have used various statistical models like the Sequential behavioural model [8], the Markov predicting model [12; 16; 17] or the Prefix method [6; 7; 12; 13], for the prediction achievement we have used a rather simple statistical computation.

```plaintext
isUltraPopular()
x = number of times user requested url in current time
y = number of times user did login in current time frame
border = (x / y) * 100
if (border > 55) return true else
```

Figure 3. Pseudo code of `isUltraPopular()` method
In addition, according to the available related literature and our knowledge there is no application that groups URLs of requested web pages according to the time of the user’s request. Such grouping enables better prediction of a user’s behaviour during her/his web browsing (surfing).

At first glance a number of prototype improvements could be identified. First, some aspects of the designed GUI should be redesigned; for example multiple URLs should be opened in tabs instead of windows, the user name should not be offered as a choice from a combo box, an address bar should occupy the whole length of a window. Second, database installation and configuration is still fairly complicated; namely it is unrealistic to expect from the end user to configure a MySQL Server and to write commands in the console.

On the other hand, the developed prototype as a working version of the software that will be upgraded before its release and employment in everyday web browsing, as such those current shortcomings would be easily overcome. When considering possible options we have concluded that the development of a new browser is not realistic and feasible because of the high competition in the marketplace (see e.g. http://en.wikipedia.org/wiki/Browser_statistics). Furthermore the used browser is a free plug-in for Java and is characterized with a high response time. Consequently, both problems could be resolved by modifying our application in order to make it a new add-on for the Mozilla web browser.

Mozilla Firefox is an open source application which allows modifications of its code and extra functionalities developed by third parties. Since Firefox is the second most popular web browser in the world market (see e.g. http://en.wikipedia.org/wiki/Usage_share_of_web_browsers in order to check recent usage share of web browsers), the modified and upgraded application, would be available to a larger user population.

Concerning the identified weaknesses, the problem of the current rather complicated configuration of the database would be automatically solved. Explicitly, Firefox is using the MySQL Lite database and, from the prototype side, the only required modification should be a change of the database connection. SQL queries would remain the same. The problem of the web browser speed would be resolved as well, because the speed of the browser would be actually the speed of Firefox. The GUI of the application should be fundamentally redesigned and adjusted to the Mozilla Firefox standards.

Furthermore, regardless of the planned modifications related to the Firefox web browser, there are a number of prototype characteristics which should be improved:
• the prototype customization is the most important issue; the user should be able to adjust the application according to her/his preferences, for example to individually choose number of time frames during the day along with the duration of each one;
• the borderline frequency of 55% may not be suitable for all users and should be subject to adjustments according to their preferences as well;
• the possibility for the user to block the automatic opening of some URLs (unblocking should also be possible) or to disregard a few URLs from the calculation related to their request.

All these modifications do require some changes in the application database and in the classes that are responsible for database communication, and could be achieved with minor effort. Additionally, in order to apply a common strategy – design, evaluate and then modify the design based on the conducted analysis – a usability assessment, as an integration of empirical methods into a laboratory-based usability test will be conducted. We conclude that with rather simple and feasible changes this preliminary work in the area of web page prediction may be turned into effective and useful browsing aid for widespread application.

5. Acknowledgements

This paper describes the results of research being carried out within the project 177-0361994-1998 Usability and Adaptivity of Interfaces for Intelligent Authoring Shells funded by MZOS of the Republic of Croatia. The authors would like to thank to Marija Đapić and Henning Schaefer for their useful help during development of the application.

6. References