2008

Web User Session Reconstruction Using Integer Programming

Dell, Robert F.

Web User Session Reconstruction Using Integer Programming

Robert F. Dell
Operations Research Department
Naval Postgraduate School
Monterey, California, USA
dell@nps.edu

Pablo E. Román*, Juan D. Velásquez
Department of Industrial Engineering
University of Chile
República 701 Santiago, Chile.
proman@ing.uchile.cl, jvelasqu@dii.uchile.cl

Abstract

An important input for web usage mining is web user sessions that must be reconstructed from web logs (sessionization) when such sessions are not otherwise identified. We present a novel approach for sessionization based on an integer program. We compare results of our approach with the timeout heuristic on web logs from an academic web site. We find our integer program provides sessions that better match an expected empirical distribution with about half of the standard error of the heuristic.

1. Introduction

A log file from a web server (*web log*) contains records of users' browsing activities and are a potentially large source of data on customer preferences [4]. A web log is a large text file with each line (*register*) containing the following: the time of a document (web page) access, the IP address of the user, the *agent* field that identifies the user's browser, and the document retrieved. A log file contains evidence of each web user's activities and serves as a huge electronic survey of a web site. This has motivated considerable research on how to mine this information, a field coined Web Usage Mining [4].

A web log by itself does not necessarily reflect a sequence of an individual user's document access, it registers every retrieval action but without a unique identification for each user. This originates the need to reconstruct a user's session from the information available (*sessionization*). Prior work on sessionization has relied on heuristics most commonly based on limited session duration [6]. In this paper, we propose an integer program to construct sessions and demonstrate its advantages using web logs from an academic web site.

*Corresponding Author

2. Our Approach

Our integer program for sessionization, like the heuristic approaches, groups log registers from the same IP address and agent as well as ensuring the link structure of the site is followed in any constructed session. Unlike the heuristics, it does not construct the sessions one at a time but instead simultaneously constructs all sessions. The ability of the integer program does come at the cost of increased solution time. The integer program we present here has been reformulated several times to improve performance as we experimented with data.

Each constructed session from a web log is an ordered list of log registers where each register can only be used once in only one session. Our integer program uses a binary variable X_{ros} that has value one if log register r is assigned to the oth position during session s and zero otherwise. Each index r identifies a unique register, each index s identifies a unique user session, and the index o is the ordered position of a register during a session. In the same session, it is possible for register $r2$ to occur immediately after register $r1$ if the two registers share the same IP address and agent, a link exists from the page requested by $r1$ to the page requested by $r2$, and the request time for register $r2$ is within an allowable time window since the request time for register $r1$.

We present the integer programming formulation below in NPS standard format [1].

2.1. Indices

o	Order of a log register during a session ($e.g.$ $o = 1, 2, \cdots, 20$). The cardinality defines the maximum length of a session.

p, p'	Web page.
r, r'	Web log register.
s	Web user session.

2.2. Index Sets

$r' \in bpage_r$	The set of registers that can be the register immediately before register r in the same session.
$r \in first$	set of registers that must be first in a session.

2.3. Data [units]

Used to produce the index sets above:

$time_r$	the time of register r [seconds].
ip_r	the IP address for register r.
$agent_r$	the agent for register r.
$page_r$	the page for register r.
$\underline{mtp}, \overline{mtp}$	the minimum, maximum time between pages in a session [seconds].
$adjacent_{p,p'}$	one if a page p' can be reached in one click from page p.

Used in formulation:

C_o	the objective function coefficient of having a register assigned to the oth position in a session.

2.4. Binary Variables

X_{ros}	1 if log register r is assigned to the oth position during session s and zero otherwise.

2.5. Formulation

Maximize $\sum_{ros} C_o X_{ros}$

Subject to:

$$\sum_{os} X_{ros} \leq 1 \qquad \forall r \qquad (1)$$

$$\sum_{r} X_{ros} \leq 1 \qquad \forall o, s \qquad (2)$$

$$X_{r,o+1,s} \leq \sum_{r' \in bpage_r} X_{r',o,s} \qquad \forall r, o, s \qquad (3)$$

$$X_{ros} \in \{0,1\}, \quad \forall r, o, s,$$
$$X_{ros} = 0, \quad \forall r \in first, o > 1, s$$

The objective function expresses a total reward for sessions where a reward of $\sum_{o' \leq o} C_{o'}$ is obtained for any session of size o. As an example, setting $C_3 = 1$ and $C_o = 0 \quad \forall o \neq 3$ provides an objective function for maximizing the number of sessions of size three. Section 3 reports on how we varied the values of C_o and the results obtained.

Constraint set (1) ensures each register is used at most once. Constraint set (2) restricts each session to have at most one register assigned for each ordered position. Constraint set (3) ensures the proper ordering of registers in the same session. $X_{ros} \in \{0,1\} \forall r, o, s$ defines variables as binary. To improve solution time, we can fix (or eliminate) a subset of these binary variables to zero ($X_{ros} = 0, \forall r \in first, o > 1, s$). After forming the set $bpage_r$, the set $first$ is easily found ($r \in first$ if $bpage_r = \emptyset$).

3. Web Log Processing

We consider a university web site that hosts the main page of the Industrial Engineering Department of the University of Chile (http://www.dii.uchile.cl), sub-sites of research groups, personal homepages, a web mail site, academic programs and related project sub-sites. As a general purpose site, it has a lot of diversity and reasonably high traffic, although much of this traffic comes from web mail which we do not consider.

We collected $3,756,006$ raw registers over a time window of one month, April 2008. We want to find sessions consisting of just web pages so we filtered out any other multimedia objects or errors. The final total for our study is $102,303$ clean registers of static html pages and some dynamic pages (php and jsp), for a total of 172 different pages. These 172 pages have $1,228$ links between them. We find that only a few IP addresses account for the vast majority of all clean registers. Over 98 percent ($16,785$ out of a total of 16,985 unique IP addresses) have less than 50 register for the entire month. Figure 1 displays the number of registers for the 100 IP addresses that account for the most registers.

We also found how many unique web pages are visited by each IP address; IP addresses that visit many unique web pages have more diverse sessions. Almost 84 percent of the IP addresses ($14,265$ out of $16,985$) visit three or less different pages for the entire month.

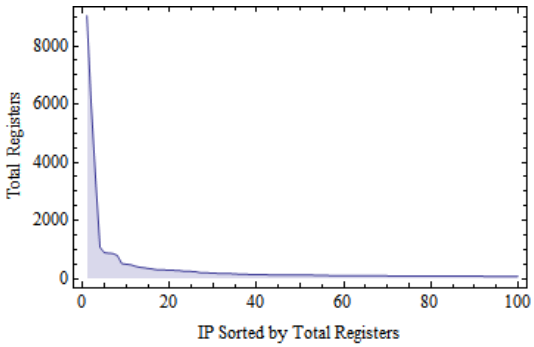

Figure 1. The number of registers for the 100 IP addresses that account for the most registers.

3.1. The Sessionization Experiments

We partition a web log into chunks where each one is formed such that no register in one chunk could ever be part of a session in another. This is easily accomplished by partitioning chunks so that each one corresponds to a unique IP and agent combination. A chunk may be further divided whenever the time difference between two consecutive registers (where the registers are sorted by time) exceeds $\overline{mtp}$.

We select the most relevant chunks taking in account the diversity of pages that it contains. The measure of diversity we use is entropy, $S = \sum_p f_p Log_N(1/f_p)$, where f_p is the frequency of page p occurrence over all register entries for the same IP address and N is the number of unique pages visited by the same IP address. S takes values from zero to one. When the value of S is near zero, most register entries are for the same page, if the value is near one all the pages are visited with similar frequency. There are many IP addresses with diversity near zero (visiting one page most of the time) and many IP addresses with high diversity but a low number of registers for the entire month. We concentrated on the IP addresses with high diversity and a high number of registers reckoning that these are the most interesting (and most difficult to solve) for sessionization.

Selecting the IP addresses with more than 50 registers and S greater than 0.5, results in 130 IP addresses with 17,709 registers (17.3% of the total number of registers). We obtain 403 chunks when we partition these registers such that each chunk has at least 50 registers and $\overline{mtp} = 300$.

All computation is done using 1.6Ghz PC with two Gbs of RAM. We generate the integer program using GAMS and solve it using CPLEX version 10.1.0 [2], controlled by a php script and MySQL 5.0.27 as a data storage engine.

The 403 different integer programs (with $\underline{mtp} = 0$, $\overline{mtp} = 300$, and a maximum session size of 20) range in size from about $4,000$ to $292,000$ binary variables and $8,000$ to $281,000$ constraints. For each integer program, we set the maximum time limit to 300 seconds and the $relative\ gap$ to one percent. With such limits, the CPLEX solver terminates when it has a solution guaranteed to be with one percent of optimal or it reaches a 300 second (five minute) limit. If it reaches a 300 second limit, it provides the best solution it has found by that time.

Without knowledge of the real sessions, it is not straightforward to quantify the quality of sessions produced by a heuristic or our integer program. We propose to measure the quality of the distribution of session sizes matching the empirically observed power law distribution [3, 7]. We use linear regression on the logarithm of the size and the logarithm of the number of sessions. We report the regression correlation coefficient and standard error as our measures of sessionization quality. The closer the correlation coefficient is to one and the standard error near to zero, the better the sessionization result.

We performed many experiments with the objective function coefficients C_o and found most sets of values that reward sessions of longer size produced good quality sessions. Table 1 presents five sets of different objective function coefficient values used along with the resulting correlation coefficient and standard error. Figure 2 shows how many of each session size were found by our integer program for sizes two and higher, the power law distribution fit, and the resulting correlation coefficient for the third set of objective function coefficients.

	$C_o =$	R^2	StdError
1	$1/\sqrt{o}$	0.9222	1.0548
2	$Log(o)$	0.9752	0.4126
3	$3/2Log(o) + (o-3)^2/12o$	0.9784	0.3827
4	o	0.9742	0.4242
5	o^2	0.9607	0.5145

Table 1. Five sets of different objective function coefficient values and the resulting correlation coefficient and standard error.

The computation time was similar for all sets of coefficients. We report details for the third set. Over 85% (344 out of 403) of the chunks obtained a solution within one percent of optimal. The average generation and solution

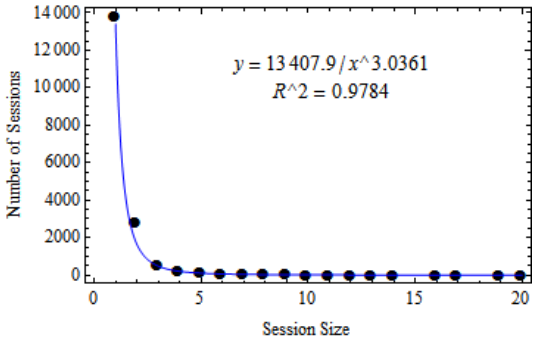

Figure 2. Session size found and the power law distribution fit.

time for these chunks was only 11 seconds. The 300 second limit was reached in only 59 out of the 403 chunks. Above 230,000 discrete variables, most of the instances reach the 300 second time limit.

For the 59 chunks reaching the limit, the average relative gap was 70%. The *relative gap* is the percentage difference between the best solution found and a theoretically best solution. We increased the solution time for several of these 59 chunks and found in most cases the additional time did not produce better solutions but did improve the bound on the theoretically best solution and therefore improved the relative gap. A further division of these 59 chunks into smaller chunks is another possible approach we tried on a few of these but this too didn't produce better sessions.

We also considered a change to the objective function so as to find the maximum number of sessions of a given size for use in other research [5]. Specifically, for a specific $size$ session, $C_o = 0\ \forall o \neq size$, and $C_o = 1$ for $o = size$. We see that relatively few (257) sessions of size six (or higher) are possible in comparison to sessions of size 2 (3000). These results simplify the task of finding combinations of parts of a session of fixed size.

In addition to changing objective function coefficients, we also changed several other parameters of the integer program such as doubling the maximum session size from 20 to 40 and $\overline{mtp}$ from 300 to 600 seconds. In all cases, we observed very little change to the resulting sessions.

3.2. Comparing With a Time-Oriented Heuristic

We compare our results with a traditional sessionization timeout heuristic. The timeout heuristic is substantially faster (only 13 seconds) but results in a distribution of sessions with only a $R^2 = 0.9181$ correlation coefficient (not as good as the $R^2 = 0.9784$ found by the integer program) and a standard error of 0.6401 (nearly twice the standard error of 0.3817 found by the integer program).

4. Conclusions and Future Research

We present a novel approach for sessionization using integer programming. When compared to a commonly used heuristic, the sessions produced by the integer program better match an expected empirical distribution. The integer program also allows us to determine the maximum number of sessions of a certain size. Future research will focus on improving solution time. We will also change our integer program to seek the maximum number of a specific session(s) (pattern matching) that could be possible for a given log file. We will also enhance our integer program to model the possibility of a user's activation of the back and forward browser button.

Acknowledgement

This work has been partially supported by the National Doctoral Grant from Conicyt Chile and by the Chilean Millennium Scientific Institute of Complex Engineering Systems.

References

[1] G. G. Brown and R. F. Dell. Formulating integer linear programs: A rogues' gallery. *INFORMS TRANSACTIONS ON EDUCATION*, 7(2):1–13, 2007.

[2] GAMS Development Corporation. Gams/cplex, 2008. Solver CPLEX for GAMS (http://www.gams.com/dd/docs/solvers/cplex.pdf).

[3] B. Huberman, P. Pirolli, J. Pitkow, and R. M. Lukose. Strong regularities in world wide web surfing. *Science*, 280(5360):95–97, 1998.

[4] J.D.Velásquez and V. Palade. *Adaptive Web Sites: A Knowledge Extraction from Web Data Approach*. IOS Press, Amsterdam, NL, 2008.

[5] P. Román and J. D. Velásquez. Markov chain for modeling web user browsing behavior: Statistical inference. In *XIV Latin Ibero-American Congress on Operations Research (CLAIO)*, 2008.

[6] M. Spiliopoulou, B. Mobasher, B. Berendt, and M. Nakagawa. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. *INFORMS Journal on Computing*, 15(2):171–190, 2003.

[7] A. Vazquez, J. G. Oliveira, Z. Dezso, K.-I. Goh, I. Kondor, and A.-L. Barabási. Modeling bursts and heavy tails in human dynamics. *PHYSICAL REVIEW E*, 73(3):036127, 2006.