Measuring the Accuracy of Sessionizers for Web Usage Analysis

Bettina Berendt, Bamshad Mobasher, Myra Spiliopoulou, Jim Wiltshire

Humboldt-University Berlin, Inst. of Pedagogy and Informatics, berendt@educat.hu-berlin.de;
DePaul University, Dept. of Computer Science, mobasher@cs.depaul.edu;
Humboldt-University Berlin, Fac. of Economics, Inst. of Information Systems,
myra@wiwi.hu-berlin.de;
IBM eBusiness Solutions Center, jwiltshire@us.ibm.com

Abstract

Companies with web presence rely on web usage analysis to obtain insights on customer behavior, associations among products, impact of advertisement banners, web marketing campaigns and product promotions. The validity of these results depends heavily on the accurate reconstruction of the visitors' activities in the web site. To this end, many sites employ cookies that distinguish among different users coming from the same proxy server or anonymizer. However, the set of activities thus grouped together refer to the whole lifetime of a cookie at the user's host. The activities performed during each visit to the web site, the "sessions", are not grouped properly, thus prohibiting the monitoring of changes in the user's behaviour and in her interaction with the site during each session.

The reconstruction of user sessions, the so-called "sessionizing" is blurred by client caches and multiple instantiations of the user's browser. Sessionizing tools exploit information on the site's topology and statistics on its usage, in order to assess the correct contents of a user session. These tools are based on heuristic rules and on assumptions about the site's usage, and are therefore prone to error.

In this study, we provide a formal framework for the evaluation of the accuracy of sessionizing tools. We introduce a set of measures that compute the extent to which real sessions are successfully reconstructed by different sessionizers. The wide range of measures proposed reflects the fact that some web usage analysis applications require exact reconstruction of a session, while for others ordering and page revisits are not important.

On the basis of these measures, we compute and evaluate a number of sessionizing tools using the log data of a real web site.

1 Introduction

Web mining is rapidly becoming a cutting edge technology, in much the same way as "data mining" a few years ago. Web site owners analyze the usage of their sites to gain insights about the image of their company among their potential customers, about the contact and the conversion efficiency of their site, about the success of their offerings and the ROI of their electronically purchased products, about the profiles and the navigational behavior of the users. The validity of the results of this analysis depends on the quality of the original data. The quality of web usage data is affected by difficulties in (a) distinguishing among different visitors of the site and (b) identifying when a user has abandoned the site and (c) reconstructing all activities of the user in the site.

The effect of the first ambiguity can be best understood if we draw a parallel from market basket analysis. Imagine a supermarket, in which most customers shop for themselves *and* for their friends. Some put into their cart the purchases for themselves and for their friends indistinguishably, while others use one basket for themselves and one for each of their friends but occasionally mix things so that baskets attain approximately the same weight. Drawing conclusions about the buying behavior of *individuals* in this store would lead to grossly erroneous results.

The effect of the second and third ambiguities is more apparent if we look closer at the behaviour of people inside a conventional store, say a bookstore. When does a person enter the store, how long does she take before choosing something to buy, what books does she consult? The act of buying books and paying for them is recorded, but this act does not imply that the customers leave the store. Furthermore, there is no recording about the people that do not buy anything. Hence, even simple statistics like ratio of buyers to visitors or average visiting time in the store cannot be computed safely. Although a Web server records many of the activities performed by a site visitor, it neither records them all, nor can it name the last one.

Source of the above ambiguities is the HTTP standard specification about the information that can be recorded for each client request [FGM+99]. This information is not always sufficient to distinguish among users accessing a site from the same host or through the same proxy, nor to detect the end of a visit. Tools have soon been devised to overcome these shortcomings, including cookies, timeout-based sessionizing mechanisms and topology-aware heuristics [CMS99]. However, all these tools are of heuristic nature and their output is a *guess* on the identification of a user and/or the end of a visit. Before performing statistical analysis on data prepared by these heuristics, it is imperative to quantify the error they introduce.

In this study, we respond to this necessity, which is often overseen. We propose a formal framework, in which we measure the performance of heuristics that identify the end of a site visit and reconstruct its contents, a process often called "sessionizing". To this purpose, we have designed a comprehensive set of measures, intended for a variety of web mining applications. On the basis of these measures, we compare the output of the heuristics for a real web site, and we identify the characteristics of the site and of its usage that are expected to most affect the performance of each heuristic. Thus, our measurements can be generalized and conclusions on the applicability of each heuristic in a given type of site can be drawn.

This paper is organized as follows. In the next section we discuss the sources of error for heuristics applied to identify users (like cookies) and for heuristics deployed to reconstruct the activities of a user during one visit to a site. We concentrate on the second group of heuristics, assuming a reliable user identification, and we give an overview of such heuristics. We then present our framework for the comparison of heuristics on the basis of a set of measures. We propose more than one measure of comparison, because different web mining applications concentrate on different facets of user behaviour. In section 4, we compare the performance of the heuristics for a real web site. The last section concludes our study.

2 Identifying users and reconstructing their sessions

The analysis of web usage does not require knowledge about a user's identity. However, it is necessary to distinguish among different users. The information available according to the HTTP standard is not adequate to distinguish among users from the same host, proxy or

anonymizer. The most widespread remedy amounts to the usage of cookies. A cookie is a small piece of code associated with a web site; it installs itself in the user's host and associates a cookie identifier with the user's browser. This identifier is sufficient to recognize the user that launches each URL request, as soon as the same browser is being employed.

Cookies have several drawbacks. First, they are not always welcome: Many users see a privacy threat in them, while others are concerned about compromising the security of their host by letting an alien piece of software install itself in it. Second, a cookie is associated with the user's browser: If the user decides to operate multiple browsers, she is perceived as multiple persons by the web server. Third, a cookie provides for user identification across multiple visits to the site, but has no means of recognizing the end of a visit and the beginning of the next, i.e. of the borders of user "sessions", nor of reconstructing all activities performed by the user.

In this study, we concentrate on the problem of reconstructing a user's session, i.e. the group of activities performed by the user from the moment she entered the site to the moment she left it. We assume that user identification is already performed in a reliable way. The error introduced by heuristics for user identification and its impact on session reconstruction errors will be investigated in a forthcoming paper.

2.1 The notion and content of a session

In accordance with W3C [W3C99], we term as "session" or "visit" the group of activities performed by a user from the moment she enters the site to the moment she leaves it. Since a user may visit a site more than once, the Web server log records multiple sessions for each user. We use the name "user activity log" for the sequence of logged activities belonging to the same user. Thus, "sessionizing" is the process of segmenting the user activity log of each user into sessions. A tool that implements this process is a "sessionizing *heuristic*": it reconstructs a session on the basis of assumptions about user behaviour.

The contents of a (re)constructed session depend on the requirements of the mining application. In many applications, including market basket analysis and establishment of usage profiles, the expert is interested in the pages being accessed during a session but not in the order of access, nor on revisitations. Hence, a session is a *set* of activities. If the navigational behaviour of users is studied, the order of access is of interest. Then, a session must be modelled as a *sequence* of activities. If the effect of revisitations is of interest, e.g. to investigate the causes of disorientation, then it is necessary to expand each session with the pages revisited but not recorded, due to caching.

2.2 Sessionizing heuristics

A sessionizing heuristic partitions the user activity log into a set of "constructed sessions", thereby deciding which activities of the same user belong together. Contrary to it, a "real session" contains the activities that the user performed together according to a reference model, which in our experiments is provided by the Web server of the test site. The quality of a heuristic is evaluated by juxtaposing its constructed sessions to the real ones, comparing them in terms of a measure and deriving a quality value.

We distinguish between *time-oriented heuristics* that rely on the temporal properties of the user activity log and *navigation-oriented heuristics* that derive their sessionizing rules from assumptions on the way users navigate.

2.2.1 Time-oriented heuristics

Time-oriented heuristics consider boundaries on the time spent on a page or in the entire site during a single visit.

Catledge and Pitkow [CP95] measured mean inactivity time within a site, and came to a value of 9.3 minutes; adding 1.5 standard deviations they got a 25.5 minute cut-off for the duration of a visit. This has been rounded up to 30 minutes in most applications, defining the maximal length of a session [CMS99, SF99].

During a visit, the time spent by a user to read and process the contents of any single page varies within certain limits. If a long time elapses between one request and the next, it is likely that the latter request is the first of a new visit. Obviously, the page stay time varies with the contents of the page and the nature of the application. Heuristics of this type are used in [CMS99, SF99].

2.2.2 Navigation-oriented heuristics

Navigation-oriented heuristics take the linkage between pages into account. The rationale behind their segmentation rules is that users usually follow hyperlinks to reach a page, rather than typing URLs.

A heuristic proposed by Cooley et al. in [CMS99, CTS99] says that if the requested web page is not reachable from previously visited pages, then it should be assigned to a different session. It must be noted that the hyperlink to the requested page P does not need to belong to the page visited immediately before P; the user may have moved backwards to another page accessed earlier and followed a hyperlink to P. Since backward moves are not always recorded in the Web server log due to caching, the original heuristic of [CMS99, CTS99] reconstructs the backward move of the user and adds the corresponding page accesses to the session.

The previous heuristic uses the topology of the web site graph. Consultation of the site topology presupposes the availability of the site's graph in an appropriate format. A less demanding heuristic proposed in [CMS99] relies on the exploitation of the referrer information recorded in the Web server log (optional entry, according to the HTTP specification): the "referrer" of a URL request is the page from which the request was issued. According to this heuristic, two consecutive URL requests `A` and `B` belong to the same session if and only if the referrer URL in the request for `B` is the previously invoked URL `A` [CMS99].

Referrer-based heuristics are more restrictive than topology-based ones: If a pair of consecutive URLs `A, B` is classified as belonging to one session by a referrer heuristic, it will also be classified as belonging to one session by a topology heuristic. This is because if `B` *was* reached from `A`, there must have been a hyperlink from `A` to `B`. There are also cases where there is such a hyperlink, but `A` is not `B`'s referrer. Here, referrer-based heuristics would assign the URLs to two sessions, topology-based heuristics would assign them to one session. This makes referrer-based heuristics more suitable than topology-based heuristics for highly connected sites.

The exploitation of referrer data for sessionizing becomes more complicated for sites using frames. Then, a page consists of a set of frames, its "frameset", but the referrer-based heuristic cannot know that because it only consults the Web server log for sessionizing. Upon a request for this page, all frames of the frameset are loaded in sequel, all of them having the same referrer. Since the referrer-based heuristic does not consider the site topology, it is not aware

of the fact that the frames belong to the same page. In the next section, we will describe how this problem can be addressed.

2.3 A selection of sessionizing heuristics for evaluation

In the present study, we evaluate the performance of the following variations of the time-oriented and navigation-oriented heuristics described thus far. Each heuristic h scans the user activity logs to which the Web server log is partitioned after user identification. For each user activity log u with n_u users, it outputs the constructed sessions $C_h(u,1),\ldots,C_h(u,n_u)$ comprising it.

h1: Time-oriented heuristic: The duration of a session may not exceed a threshold θ. Let t_0 be the timestamp of the first URL request in a constructed session. A URL request with timestamp t is assigned to this session iff $t - t_0 \leq \theta$. The first URL request with timestamp larger than $t_0 + \theta$ becomes the first of the next constructed session.

h2: Time-oriented heuristic: The time spent on a page may not exceed a threshold δ. Let t' be the timestamp of the URL most recently assigned to a constructed session. The next URL request belongs to the same session iff for its timestamp t'' it holds that $t'' - t' \leq \delta$. Otherwise, this URL becomes the first of the next constructed session.

h-ref: Referrer-based heuristic: Let p and q be two consecutive page requests with p belonging to a session S. Let t_p and t_q denote the timestamps for p and q, respectively. Then, q will be added to S, if the referrer for q was previously invoked within S, or if the referrer is undefined and $(t_q - t_p) \leq \Delta$, for a specified time delay Δ. Otherwise, q is added to a new constructed session.

To understand the details of the referrer-based heuristic, we need to take into account the special role of the "undefined" referrer (here denoted by "-"). In many logs, this may be recorded in various situations: (1) As the referrer of the start page, or of a page that was entered after a brief excursion to a subsite or a foreign server. This may happen, for example, because a site does not record external referrers. (2) As the referrer of a typed-in or bookmarked URL. (3) When a frameset page is reloaded in mid session. (4) For all these pages, when they are reached via the back button during the real session. (5) In a frame-based site: as the referrer of the first frames that are loaded when the start page containing the top frameset is requested.

The time delay Δ in the above definition is necessary to allow for proper loading of frameset pages whose referrer is undefined, and to account for other situations resulting in mid-session requests to have undefined referrers. In our current study we have used a time delay of 10 seconds.

These heuristics are applicable across a wide range of site types. In particular, sites with dynamically generated pages lend themselves to sessionizing. All the heuristics mentioned above can be applied to sequences of such pages, because at least the URL stem is recorded in the logs like any other URL. (In fact, sessionizing will often be easier for sites with dynamic pages, because these sites are less affected by caching.) In addition, application-specific versions of referrer heuristics may become possible if the site's log file can record the parameters

that control the generated pages in the URL query strings. For example, two consecutive URLs can be assigned to the same session if the second query string contains all parameters of the first plus some more, thereby refining the first (cf. [BS00]). (A site not recording the parameters controlling dynamic pages (e.g. because POST is used as the request method), may however encounter problems in mining and analysing the patterns within and across the sessions constructed. This problem can require the integration of data from application servers and databases, and potentially, some content generation.)

3 Measuring the Success of a Heuristic in Reconstructing a Session

3.1 Basic entities

We denote by U the set of URLs in the web-site, including all static web pages and all URLs that can be dynamically generated by different parameter settings of scripts.

Then, the server log is a file L of invocations of URLs in U. L can be regarded as a set. However, its records are ordered by timestamp of the invocation, so that it is meaningful to speak of the i^{th}-entry in the log, referred to as $L[i]$. According to the HTTP specifications, an entry $l = L[i] \in L$ contains information such as the requested URL, the timestamp of the request, the host identifier, the status of the request etc. Optionally, the referrer URL and the agent used by the user are also recorded. We adhere to the notation $l.url$ for referring to the URL requested in the entry $l \in L$, and use self-explaining names for all data fields recorded in the log.

As discussed in section 2.2, we distinguish between a "real session", namely a sequence of URL requests performed by a user, and a "constructed session", namely a sequence of URL requests reconstructed by a heuristic from the log data. The log L is partitioned into an ordered collection of real sessions $\mathcal{R}$. Each heuristic partitions L into an ordered collection of constructed sessions $\mathcal{C}$. The ideal heuristic would partition each log in such a way that $\mathcal{C} = \mathcal{R}$.

Similarly to the notation for the log L, $\mathcal{R}[i]$ denotes the i^{th} real session, while $\mathcal{C}[i]$ denotes the i^{th} constructed session.

3.2 Relationships between the logs and the sessions

A heuristic h is modeled as a partition scheme over L, mapping the entries of L into elements of constructed sessions, such that:

1. Each entry in L is mapped to exactly one element of a constructed session.
2. The mapping is order-preserving.

Thus, each heuristic h produces a set of constructed sessions $\mathcal{C}_h$. The ideal heuristic ih would produce the set of real sessions, i.e. $\mathcal{C}_{ih} = \mathcal{R}$.

Reconstructing a real session. A heuristic h builds constructed sessions. The measures we propose in the following quantify the successful mappings of real sessions to constructed sessions, i.e. the "reconstructions of real sessions".

Definition 1 A real session is "completely reconstructed" *if all its elements are contained in the same constructed session.*

More formally, the real session r having n elements $r[1], \ldots, r[n]$ is completely reconstructed if there exists a constructed session $c \in \mathcal{C}$ with m elements $c[1], \ldots, c[m]$ such that:

$$\forall i = 1 \ldots n \exists j \in \{1, \ldots, m\} : r[i] = c[j] \tag{1}$$

In this study, we operate on data already segmented by users, where parallel or overlapping activities of one user are always part of one real session. And the sessionizing heuristics we employ treat two subsequent requests A,B by either assigning B to the same constructed session as A, or they start a new constructed session starting with B. Therefore, constructed sessions cannot contain gaps. I.e., it is not possible for a sequence of requests A,B,C from one real session to be split up such that A,C are part of one constructed session and B part of another. Therefore, we can deduce that a session r that is completely reconstructed by a session c is also *continuously* reconstructed, i.e. that r is a subsequence of c.

3.3 Measures of goodness-of-fit

A measure M evaluates a heuristic h based on the differences between $\mathcal{C}_h$ and $\mathcal{R}$. It assigns to h a value $M(h) \in [0, 1]$ so that $M(ih) = 1$.

In our analysis, the set $\mathcal{R}$ is known. We consider different heuristics and a variety of measures assessing the quality of each heuristic. We distinguish between "categorical" and "gradual" measures. Categorical measures count only the number of sessions constructed in a certain way, and are normalized by the total number of sessions built. Gradual measures also take into account how close a constructed session is to a real session.

3.3.1 Categorical Measures

According to Def. 1 on completely reconstructed real sessions, we define a *base categorical measure*:

- The measure M_{cr} grades a heuristic h by the number of completely reconstructed real sessions contained in $\mathcal{C}_h$ divided by the total number of real sessions $|\mathcal{R}|$.

From this, we establish several *derivative categorical measures* by examining the containment relationship between a real and a constructed session in more detail

1. The measure $M_{cr,start}$ considers only completely reconstructed real sessions whose first element is also the first element of the constructed session, i.e. each real session r for which there is a constructed session c such that:

$$(\forall i = 1 \ldots n \exists j \in \{1, \ldots, m\} : r[i] = c[j]) \wedge (r[1] = c[1])$$

2. The measure $M_{cr,end}$ considers only completely reconstructed real sessions whose last element is also the last element of a constructed session, i.e. each real session r for which there is a constructed session c such that:

$$(\forall i = 1 \ldots n \exists j \in \{1, \ldots, m\} : r[i] = c[j]) \wedge (r[n] = c[m])$$

3. The measure $M_{cr,start-end}$ considers sessions that are reconstructed with correct starts *and* ends, analogously to the previous measures. $M_{cr,start-end}$ gives the number of constructed sessions that are identical to the corresponding real session.

In Fig 1, we show the different relationships between a real and a constructed session that give rise to these measures.

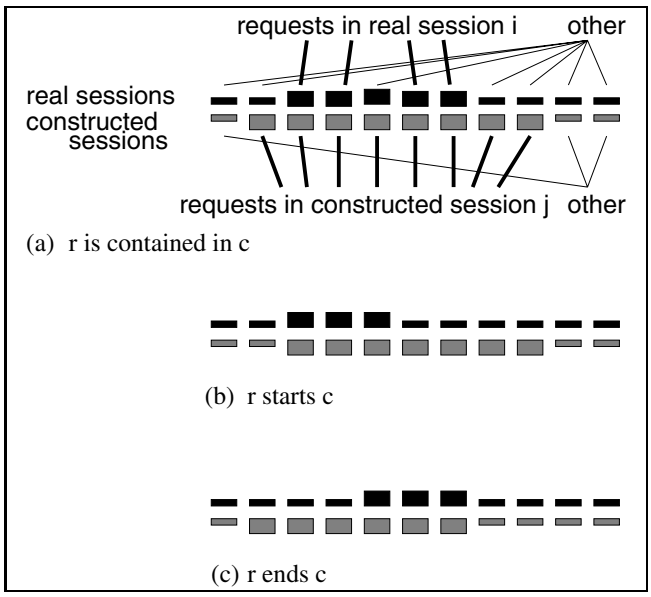

Figure 1: Possible relations between a real session r and a constructed session c that completely reconstructs r (examples).

The above categorical measures grade a heuristic by the number of real sessions contained in the constructed sessions it has built. In an analogous way, we can define measure that grade a heuristic by the number of constructed sessions contained in the real ones. We do not consider such measures any further.

3.3.2 Gradual measures

The categorical measures introduced in the previous paragraphs grade a heuristic by the number of real sessions it has reconstructed in their entirety. In most cases, the sessions constructed by a heuristic will overlap partially with the real sessions. Then, we are interested in measuring the degree of overlap between real and constructed sessions.

Definition 2 *The "degree of overlap between a real and a constructed session" is the number of elements they have in common divided by the total number of elements of the real session.*

Formally, for a real session r with n elements and a constructed session c with m elements, the degree of overlap $deg_o(r, c)$ is the ratio:

$$deg_o(r, c) = \frac{|\{i \in \{1, \ldots, n\} | \exists j \in \{1, \ldots, m\} : r[i] = c[j]\}|}{n} \quad (2)$$

The degree of overlap in Def. 2 refers to a particular constructed session. To compute the degree of overlap for a real session, we need a function f that selects among the degrees of overlap of the individual constructed sessions. For example, f can be defined as the average or the maximum degree over all constructed sessions. In this study, we define f as the maximum, so that:

$$f(r, h, deg_o) = \max_{c \in \mathcal{C}_h} \{deg_o(r, c)\} \quad (3)$$

Finally, to grade a heuristic on the degree of overlap, we need a function g that returns an aggregate of the degrees of overlap computed for the real sessions. We can define g as the average degree of overlap achieved for a real session, or again as the maximum. We opt for the former, so that for a heuristic h and its degree of overlap function deg_o:

$$g(h, deg_o) = avg_{r \in \mathcal{R}} \{f(r, h, deg_o)\} \quad (4)$$

On the basis of these definitions, we introduce the gradual measure: $M_o(g)$ that grades a heuristic h by the value of function g on the degree of overlap among the sessions in $\mathcal{C}_h$:

$$M_o(g)(h) = g(h, deg_o)$$

and, substituting from Eq. 4:

$$M_o(g)(h) = avg_{r \in \mathcal{R}} \{f(r, h, deg_o)\}$$

As an illustration, consider the example session pairs of Fig. 1. Their overlap values are all 1.

However, these examples also illustrate that the length of the constructed sessions is not taken into account, i.e. the parts of other sessions erroneously assigned to the current session does not affect the value. We therefore also include the following measure:

Definition 3 *The "degree of similarity between a real and a constructed session" is the number of elements they have in common divided by the total number of elements of the union of the real and the reconstructed sessions.*

Formally, for a real session r with n elements and a constructed session c with m elements, let $w_a = |\{i \in \{1, \ldots, n\} | \exists j \in \{1, \ldots, m\} : r[i] = c[j]\}|$. The degree of similarity $deg_s(r, c)$ is the ratio:

$$deg_s(r, c) = \frac{w_a}{n + m - w_a} \quad (5)$$

The measure $M_s(g)(h)$ is defined analogously to $M_o(g)(h)$.

Fig. 2 shows a real session that is not completely contained in a constructed session, but overlaps it to a significant degree.

When sessionizing a log that has already been partitioned into different (real) users activities, a complete reconstruction is desirable for two reasons. First, it correctly shows the set of pages accessed together. This is appropriate for applications that group pages together and associate objects [PE98, CTS99, ZXH98], because the exact succession, without any intermediate requests, is not of primary importance here. Second, a complete reconstruction also maintains the order of page requests. This is appropriate for applications analyzing user behavior, because here the exact sequence matters [Spi99, BBA+99, BL99, CY96].

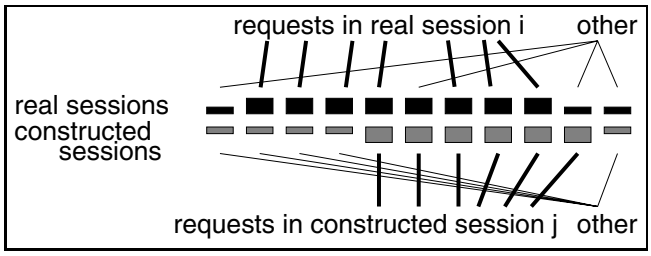

Figure 2: Overlapping real and constructed sessions (example).

However, the differentiation in the proposed measures may be helpful to distinguish between heuristics depending on the goal of the data mining analysis. For example, an analysis that investigates the main entry points to the site requires a sessionizing heuristic that performs well on $M_{cr,start}$. A different analysis may search for those pages where many users leave the site, in order to improve these pages design. Such analysis requires a sessionizing heuristic that performs well on $M_{cr,end}$. Even if two heuristics fare poorly on complete reconstruction, the one that produces larger overlaps is likely to generate more constructed sessions containing important information on associations between pages and objects.

4 Evaluation of the Heuristics

4.1 Description of the Test Site

The test data were from a university site and contained 174663 requests issued between the 18^{th} and 29^{th} of November, 2000. During that time, caching was disabled by the site to reflect the navigation behavior of users as accurately as possible.

The site is cookie-based, and the server generates session ids internally. This mechanism starts a new real session each time a browser instance is activated (even if from the same user). The session mechanism occurs whether or not there is a user cookie. We obtained a set of 4400 users and 14279 real sessions. Removing the boundaries between real sessions, we generated constructed sessions according to the different heuristics.

The site is frame-based. It records referrers in the way described in section 2.3.

4.2 Experiments

In four experimental suites, both categorical and gradual measures were tested.

h1-30 was the "duration: total" heuristic with 30 minutes as cutoff, and h2-10 was the "duration: time spent on a page" heuristic with a 10-minute cutoff. The 30-minute cutoff was used based on the results mentioned in section 2. Heuristic h-ref was the referrer heuristic described in the previous section. Also, experiments conducted with various thresholds (using 10 minute intervals from 10 to 60 minutes) show that these are appropriate choices for the two temporal heuristics. (These threshold experiments show that the measures h1 and h2 are not only useful for sessionization, but also to perform cross-validation and benchmarking, for example to determine the optimum value for session timeouts.)

Figures 3 and 4 show the results for our test site, the heuristics defined in section 2, and the measures defined in section 3.

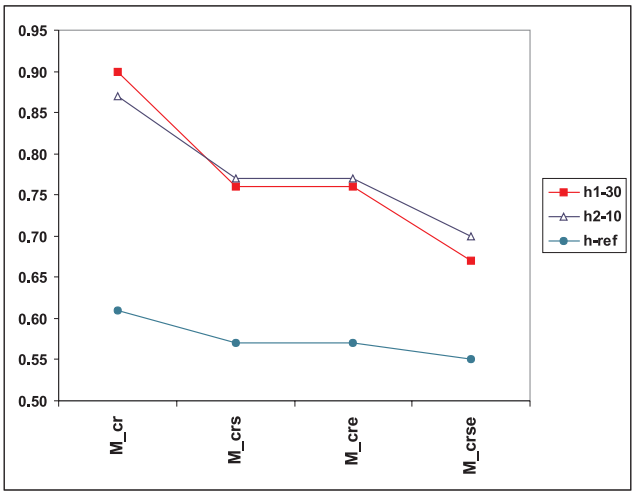

Figure 3: Categorical measures for all the heuristics. ("s" = "start", "e" = "end").

4.3 Discussion

Our results need to be analyzed in two ways: First, there are a number of logical dependencies. The measures M_{crs} and M_{cre} add constraints to M_{cr} and can therefore not be higher. The same holds for M_{crse} compared to M_{crs} or M_{cre}. Within the gradual measures, the overlap measures must be higher than the similarity measures because they do not take into account additional requests that have been erroneously assigned (cf. the discussion in section 3.3.2).

The results show that the simple rule of thumb, a 30 minute maximal duration for the whole session, performs very well (h1-30). h2-10 produces results of similar quality. However, h2-10 is slightly better on measures taking start- and endpoints into account. A likely reason for this is that the data contain a number of subsequent very short sessions, e.g. with intervals between a users requests like (10 sec – 20 sec – 10 sec – 11 min – 10 sec – 25 sec – 13 min – ...). While these requests all fit into one 30 minute total duration constructed session, h2-10 detects the comparatively long intervals between one real session's endpoint and the next real session's startpoint. Conversely, all (or most of) these sessions correctly reconstructed by h2-10 are also correctly reconstructed by h1-30, plus some more that were erroneously segmented by h2-10. This explains the better performance of h1-30 on M_{cr}. (Note that M_{cr} and also M_o favor "conservative" heuristics that insert few segmentations. The boundary case of only one constructed session would yield the value 1 for both these measures.)

The quality of h-ref turned out to be comparatively low. This indicates that the referrer heuristic, in spite of the improvements to a naive version described above, still segments too many real sessions. There are several conceivable reasons for this: (1) Real sessions composed of temporally overlapping activities in different browser windows, are segmented into separate constructed sessions. However, this occurred only rarely (about 3% of the real sessions in our sample contained such parallel activities). (2) Pages with undefined referrers (see section 2.3) are regarded for too long. This causes the heuristic to construct a new session. Such errors propagate: Once a frame is misclassified, the subsequent pages will also be misclassified. In further work, the referrer heuristic should therefore be combined with temporal heuristics in more ways than done here.

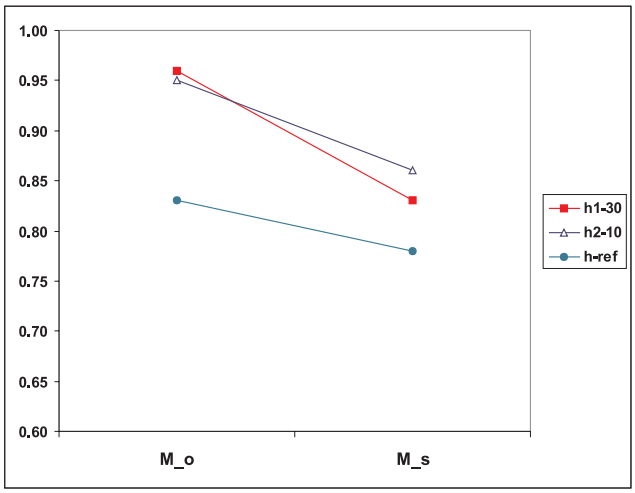

Figure 4: Gradual measures for all the heuristics.

The fact that the tested heuristics scored differently on the measures shows why it is important to perform careful sessionizing for applications where it is important to avoid (a) erroneous conclusions regarding sets of URLs accessed together based on constructed sessions comprising several real sessions (produced if M_o is high, but M_s is not), and (b) erroneous conclusions regarding navigation behavior, e.g. common entry or exit points for the site (produced if M_{cre}, M_{crs}, M_{crse} are low).

Overall, we see that the 30-minute whole-session duration heuristic performs very well, is robust, and is fully adequate for the type of site we have investigated.

4.4 Impact of caching on heuristic performance

In the current study, we have examined how the heuristics segment the sequence of requests from the user activity log that is recorded by the web server. Due to caching, the information recorded here may not contain all those pages that the user actually requested and saw.

Two questions arise. The first is in how far knowledge about these requests is essential for the analysis of web usage. The answer to this depends on the type of analysis: On the one hand, local caching does not affect the set of registered requests, as long as the usual browser setting causes each page to be retrieved when it is first requested during a browser session. On the other hand, repeated requests may be of interest during the analysis, because they indicate backward moves performed by the users.

The second question, which is more central for the present study, concerns the possible effects of caching on the performance of sessionizing heuristics, independently of whether the subsequent analysis concentrates on sets or on sequences.

Local caching could affect some of the heuristics by artificially inflating the temporal interval between two subsequent requests `A,B` reaching the server: A user may re-examine previously cached pages $\mathtt{X}_1, \ldots, \mathtt{X}_n$, and during this time, no request reaches the server. This re-examination process may take a while, in particular, when numerous pages are requested by repeated use of the back button. The browser's back button is one of the most popular navigation tools [TG97]. This may affect the referrer heuristic: If page B was indeed

requested from a previous page in the same real session (e.g., A), then B's referrer must be in the set of pages of the current constructed session, or be "-". In other words, the referrer heuristic we have proposed does take caching into account. However, the time spent re-examining $X_1, \ldots, X_n$ may be so long that a new constructed session is started with request B. In principle, the temporal heuristics based on the maximal time spent on a single page can be affected in the same way. However, this would require the re-examination of $X_1, \ldots, X_n$ for at least 10 minutes (as opposed to times of around 10 seconds in the referrer heuristic). This seems comparatively unlikely. Apart from this, local caching would affect real sessions as defined above in the same way as constructed sessions.

Proxy server caching introduces a further problem for referrer heuristics: a page seen by a user may not be found in the set of pages in the session constructed so far, because it was delivered to the user from the proxy server. Therefore, the referrer page of request B may be defined, but not appear in the set of pages of the session constructed so far. Then, a new constructed session is begun at B.

Bearing this in mind, the current solution appeared to be the best practically possible way of dealing with the problem of caching. Studies of the *quantitative* impact of caching on sessionizing heuristics will be the subject of further work. The only way to truly study the impact of caching, is to deploy (at a very large scale) client-side agents that keep track of users' actions.

5 Conclusions and outlook

In this study, we have proposed a formal framework for the comparison of session reconstruction heuristics according to a set of measures. In the design of these measures, we took the variety of web mining applications into account: Some measures are appropriate for applications that discover correlations among pages or products offered on them; for these applications, sessions can be regarded as simple sets of URL requests. Other measures are intended for applications analyzing navigational behavior; they require specific information on typical entry and exit points for the site. We have used the web server log of a real frame-based site to compare the performance of several heuristics for each measure.

Our contribution is twofold. First, our results show the impact of the parameter settings on each heuristic and depict the performance of all heuristics for measures designed for different mining applications. On the basis of these factors, the web mining analyst can select the most appropriate heuristics to prepare the data for the mining application she is working on. Second, our framework is the basis of a toolkit for the application *and evaluation* of data preparation heuristics.

In this study, we have concentrated on measuring the performance of alternative data preparation heuristics in data already segmented into different users' requests. We are currently working on evaluating the performance of the heuristics in settings where the reconstruction of sessions also includes the assignment of requests to different users. This is the typical case for sites that employ neither cookie nor session id technology. One of the central difficulties here is that a sequence of requests from the same host and employing the same agent must be split into different users' activities. We expect that referrer heuristics will perform better relative to temporal threshold heuristics, because only referrer heuristics can sort a stream of activities that are parallel or overlapping in time into different constructed sessions.

Our future work includes the modeling and quantification of the error introduced by each heuristic and the measurement of its impact on the mining results, i.e. on the confidence, support and accuracy of the discovered patterns. Moreover, we intend to establish a wider palette of data preparation heuristics and to further extend the set of performance evaluation measures into a full-fledged toolkit, in which the performance of heuristics for web data preparation can be compared on test data before they are used in the process of the analysis.

References

[BS00] Berendt, B. & Spiliopoulou, M. (2000). Analysis of navigation behaviour in web sites integrating multiple information systems. *The VLDB Journal, 9,* 56–75.

[BL99] Jose Borges and Mark Levene. Data mining of user navigation patterns. In *[MS99]*, 1999.

[BBA+99] A. G. Büchner, M. Baumgarten, S. S. Anand, M. D. Mulvenna, and J. G. Hughes. Navigation pattern discovery from internet data. In *WEBKDD'99*, San Diego, CA, Aug. 1999.

[CP95] Catledge, L. & Pitkow, J. (1995). Characterizing browsing behaviors on the world wide web. *Computer Networks and ISDN Systems, 27.*

[CY96] J. Chen, M.-S. Han and P.S. Yu. Data mining: An overview from a database perspective. *IEEE Trans. on Knowledge and Data Engineering*, 8(6):866–883, Dec. 1996.

[CMS99] Robert Cooley, Bamshad Mobasher, and Jaidep Srivastava. Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems*, 1(1), 1999.

[CTS99] Robert Cooley, Pang-Ning Tan, and Jaideep Srivastava. WebSIFT: The web site information filter system. In *[MS99]*, 1999.

[FGM+99] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical report, The Internet Society, June 1999.

[MS99] Brij Masand and Myra Spiliopoulou, editors. *KDD'99 Workshop on Web Usage Analysis and User Profiling WEBKDD'99*, San Diego, CA, Aug. 1999. ACM. Springer, LNCS series. Online archive of the extended abstracts at http://www.acm.org/sigkdd/proceedings/webkdd99/.

[PE98] Mike Perkowitz and Oren Etzioni. Adaptive web pages: Automatically synthesizing web pages. In *Proc. of AAAI/IAAI'98*, pages 727–732, 1998.

[PPR96] Pirolli, P., Pitkow, J. & Ro, R. (1996). Silk from a sow's ear: extracting usable structures from the web. *CHI-96*, Vancouver.

[Spi99] Myra Spiliopoulou. The laborious way from data mining to web mining. *Int. Journal of Comp. Sys., Sci. & Eng., Special Issue on "Semantics of the Web"*, 14:113–126, Mar. 1999.

[SF99] Myra Spiliopoulou and Lukas C. Faulstich. WUM: A Tool for Web Utilization Analysis. In *extended version of Proc. EDBT Workshop WebDB'98*, LNCS 1590, pages 184–203. Springer Verlag, 1999.

[TG97] Linda Tauscher and Saul Greenberg. Revisitation patterns in world wide web navigation. In *Proc. of Int. Conf. CHI'97*, Atlanta, Georgia, Mar. 1997.

[W3C99] World Wide Web Committee Web Usage Characterization Activity. *W3C Working Draft: Web Characterization Terminology & Definitions Sheet.* www.w3.org/1999/05/WCA-terms/

[WYB98] Wu, K., Yu, P.S. & Ballman, A. (1998). Speedtracer: a web usage mining and analysis tool. *IBM Systems Journal, 37*.

[ZXH98] Osmar Zaïane, Man Xin, and Jiawei Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Advances in Digital Libraries*, pages 19–29, Santa Barbara, CA, Apr. 1998.