Frequent Generalized Subsequences – A Problem From Web Mining

Wolfgang Gaul and Lars Schmidt-Thieme

Institut für Entscheidungstheorie und Unternehmensforschung,
Universität Karlsruhe, D-76128 Karlsruhe, Germany

Abstract. The everlasting growth of the web in terms of, e.g., amount of information, size of the net, and number of users is demanding for tools that help to tackle content (re)structuring, discover navigation patterns of users, support marketing activities of sellers (e.g., advertising and cross selling), and attract potential customers in this new environment. This paper describes how user navigation paths can be extracted from raw web logfiles and how frequent subsequences can be discovered in those paths. To better cope with navigational behaviour in the large, generalized sequences containing wildcards are used and a new algorithm for mining all frequent generalized subsequences from a given database is presented.

1 Introduction

In Bock (1974) the author writes in his preface that "automatic classification deals with the problem to structure a large set of objects into smaller homogeneous and more useful classes or groups [...] and [...] that the analysis of the ever increasing data sets is only possible with the help of computers" (translation from German)—sentences that could serve as well as introduction into this paper. Bock's early contributions have helped to establish *data analysis and classification* as a direction different from traditional statistics although he himself has incorporated statistical elements in his work wherever appropriate. In the last years *data mining* has been used as a new label for tackling large data sets (see, e.g., the critical review by Gaul and Schader (1999)), and, just recently, *web mining* is establishing itself as a promising area for the application of mining methods to data from the world wide web.

Web mining embraces two complementary aspects, *content mining* as an advancement of text mining methods to documents containing a hyperlink structure, and *usage mining* as research direction where methods for processing, detecting and analysing user navigational behaviour on web sites can be applied. Of course, buying behaviour can be analysed by traditional methods, a more recent data mining application is market basket analysis with the help of association rules. While those approaches are limited to a very narrow aspect of the buying process, the final transaction, modern e-commerce is demanding for web usage mining that aims at incorporating earlier steps of the buying process like browsing the internet, collecting information, getting aware of new products, filling the virtual market basket on a trial basis without actual buying, and thereby extending the focus from classical buying behaviour analysis to data mining concerning all kinds of contacts with potential customers before the buying decision has been made.

In this paper the starting point is the webserver's logfile. In section 2 we outline the problems involved in constructing user navigation paths from raw logfile entries. Section 3 abstracts this situation to finding frequent subsequences in a database (or set) of sequences, fixes some terminology and presents a known solution to this problem. In section 4 generalized sequences containing wildcards are used and a new approach (an appropriate adaptation of the algorithm of the previous section) is presented. Section 5 describes a reduced example to demonstrate the usefulness of the findings obtained in earlier sections. We conclude with an application to association rules and an outlook to further research.

2 From web logs to navigation paths

2.1 Problems with navigation path detection

The main source of information about anonymous visitors of a web site stems from the logfile of the web server, that lists all HTTP-requests of clients in the order they occur. Each HTTP-request is represented by a one-line-entry of the logfile within the format explained in table 1.

[ip] [name] [login] [date] [request] [status] [size] [referrer] [agent]

ip	numerical address (ip address) of the client host,
name	name of the user,
login	login of the basic HTTP-authentification given by the user,
date	date and time of the request,
request	HTTP-request line, containing request method, URL of the requested resource and desired protocol,
status	3-digit status code returned by the server,
size	size (as number of bytes) of the resource actually returned by the server,
referrer	URL of the resource containing the link to the requested resource,
agent	name of the client agent (browser).

Table 1. Logfile format: Each one-line-entry contains nine fields separated by a blank with the meaning described in this table. (More exactly this format is called *combined logfile format*, the most often used variant of an *extended logfile format*. The older *common logfile format* did not contain the referrer and user agent fields.)

We will refer to a logfile as a set (more correctly a list) L of logfile entries $l \in L$ and to the ip, referrer, and agent fields of each entry l by $l_{\text{ip}}, l_{\text{referrer}}$, and l_{agent} and to the subfield of the request field containing the URL of the requested resource by l_{res}. The other fields are of minor interest for path construction and left unlabeled.

From the information in the logfile simple usage statistics can be determined, e.g., about the home countries of visitors, the providers they use, the web sites they come from (including search keywords used to retrieve a link to the underlying site), the resources requested most often on the site, and the time (day of the week and/or time of the day) of the user activity. Accordingly, there is already a plethora of free and commercial software products to accomplish tasks of this kind (e.g., *webalizer* (http://www.mrunix.net/webalizer/) and *analog* (http://www.statslab.cam.ac.uk/~sret1/analog/) to name only two of the free and most common used tools). See Zaiane et al. (1998) for using an OLAP framework in this context.

Deeper knowledge about how visitors use a website can be gained by looking at the paths they take on the site. At first glance, it might look trivial to extract such navigation paths from the logfile: just sort all the requests by the requesting user. Unfortunately, there are several obstacles for doing so:

First: Almost no client agent reveals information about the user name, so that the name field has to be left blank; on the other hand, the login field requires the user to authenticate himself by typing a login: a questionable demand due to low acceptance by many users. Client identification can also be done by cookies (and logged in an additional field of the logfile), by a remote agent, for example an Java applet (see Shahabi et al. (1997)), or by coding session identifiers into URLs. Though easier to handle for users as there is no interaction involved, many users advise their agents not to accept cookies and not to start Java applets from unknown sites for privacy and security reasons. The acceptance of user identification can be raised by several kinds of incentives, e.g., many software producers make available technical information and early access releases for registered users only. Website personalization can be understood as another, rather general measure for convincing visitors to allow to identify them. Furthermore, the client ip seen by the server often is not the ip of the user's machine, but the ip of the proxy of a provider, i.e. all user's connecting to the internet via the same proxy of a provider are mapped to the same ip (masquerading).

Second: Almost all agents request auxiliary resources (as stylesheets and images) automatically without user interaction; the webserver cannot distinguish between explicit requests from the user and follow-up requests from his agent and logs them all, thereby polluting the logfile with a lot of (almost) irrelevant information.

Third: Normally, the client agent locally caches the requested resources and presents the user the cached copies if he requests the same resource again. In this case, even excplicit requests of resources visited shortly before may not go through to the server and so the server cannot log them.

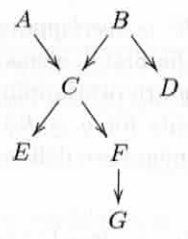
The variant with path completion by shortest path looks as follows:

```
{path completion (here: by shortest path)}
Compute a shortest path $t$ from $\tilde{s}_{\text{last}}$ to $l_{\text{referrer}}$ in the subgraph
of the site graph consisting of the resources $\tilde{s}_1, \dots, \tilde{s}_{\text{last}}$
(allowing backtracking along $\tilde{s}$ as first steps).
$S := S \setminus \{\tilde{s}\} \cup \{(\tilde{s}_{\text{ip}}, \tilde{s}_{\text{agent}}, (\tilde{s}_1, \dots \tilde{s}_{\text{last}}, t_2, \dots, t_{\text{last}}, l_{\text{res}}))\}$
```

As soon as one has more than one path started by the same (ip, agent)-pair, requests cannot be grouped to paths uniquely, if two paths intersect and a referrer page is contained in more than one path (i.e. $|S_{\text{referrer in path}}| > 1$). Here, the grouping can only be guessed by heuristics. The number of resources which have to be inserted can be used to decide to which partial path the next resource should be attached. To find plausible groupings, it is not enough just to look at the number of resources that have to be inserted to attach the next resource, but the consequences of such an attachment for the further path building process also have to be taken into consideration. Figure 2 gives an example. In the sixth step, the attachment of resource F to the second partial path BC looks more promising, as here no resource has to be inserted. As can be seen in the following two steps, this attachment probably could be wrong; the second path oszillates between resource D and G, many intermediate resources have to be inserted. If, initially, resource F would have been attached to the first partial path ACE (by inserting C), one gets a much more plausible result: the second path goes back to D (via B), the first path proceeds forward from F to G, only 2 instead of 5 resources have to be inserted. Thus, to take into account consequences with respect to the further path building process, one has to look ahead a given number of steps (by a branch and bound algorithm) and select a possibility with the lowest number of necessary insertions.—Temporal neighbourship is another criterion for attaching resources to one of several possible partial paths. If the last request of one partial path is say 15 minutes ago, but the last request of another partial path only 2 minutes, attaching the next requested resource to the latter path may be the better alternative. We summarize this step as choosing a suitable partial path $\tilde{s}$ in algorithm 1 cases dramatically by using a cut-off value for the time between two requests: if there are no requests for a fixed amount of time for a partial path, that path is closed, i.e. regarded as finished; in the literature cut-off values of 25 and 30 minutes are used, see, e.g., Cooley et al. (1999b) which perform this step separately as *session identification*; Yan et al. (1996) uses a cut-off value of 24 hours.

3 Mining frequent subsequences

Having built the set of user paths one looks for common structure in them. Generally, there are two approaches here: the first looks for subsequences



(a) site graph

step	logfile information		analyzed paths	
	requested resource	referrer	first alternative	second alternative
1	A	-	A	
2	C	A	AC	
3	E	C	ACE	
4	B	-	ACE, B	
5	C	B	ACE, BC	
6	F	C	ACE(C)F, BC	ACE, BCF
7	D	B	ACE(C)F, BC(B)D,	ACE, BCF(CB)D
8	G	F	ACE(C)FG, BC(B)D,	ACE, BCF(CB)D(BCF)G

(b) requests and paths

Fig. 2. Path detection: grouping logfile entries to user paths. Entries inserted by path completion are enclosed in brackets.

occuring in many of the user paths, the second tries to compare paths by means of a similarity or distance measure. In the following we outline algorithms used for mining sequences and generalized sequences and close with an own adaptation of a standard sequence mining algorithm to generalized sequences.

As we are interested in approaches that can handle path information coming from different groups of users we skip an approach that operates on aggregated data useful for a homogenous group of users (e.g., Borges and Levene (1998, 1999a,b)).

Let R be an arbitrary set (a set of resources) and $R^\star := \bigcup_{i\in\mathrm{N}} R^i \cup \{\emptyset\}$ the set of finite sequences of elements of R (with $\emptyset$ as the empty sequence), here used to model user paths. For a sequence $x \in R^\star$ the *length* $|x|$ is the number of symbols in the sequence ($|x| := n$ for $x \in R^n$, $|\emptyset| := 0$). Let $x, y \in R^\star$ be two such sequences: $x = (x_1, \dots, x_n), y = (y_1, \dots, y_m)$. We say that *$x$ is a subsequence of y* ($x \leq y$), if there is an index $i \in \{1, \dots, m-n+1\}$ with $(x_1, \dots, x_n) = (y_i, \dots, y_{i+n-1})$. x is a *strict subsequence of* y ($x < y$), if it is a subsequence of y but not equal to y ($x \leq y \wedge x \neq y$).

A pair of sequences $x, y \in R^\star$ *is overlapping on* $k \in \mathbb{N}_0$ *elements*, if the last k elements of x are equal to the first k elements of y ($x_{last-k+i} = y_i \quad \forall i = 1, \ldots k$). (Note that if x and y are overlapping on k elements, they are also overlapping on $q \in \mathbb{N}_0$ elements for $q \leq k$.) For such a pair of sequences $x, y \in R^\star$ overlapping on k elements we define the *k-telescoped concatenation of x and y* to be

$$x +_k y := (x_1, \ldots, x_{\text{last}-k}, y_1, \ldots, y_{\text{last}}) = (x_1, \ldots, x_{\text{last}}, y_{k+1}, \ldots, y_{\text{last}}).$$

Note that any two sequences are 0-overlapping and the 0-telescoped concatenation of two sequences is just their arrangement one behind the other. For a pair of sets of sequences $X, Y \subseteq R^\star$ we denominate the set of k-overlapping pairs $x \in X, y \in Y$ by $X \oplus_k Y$ and the set of k-telescoped sequences of all k-overlapping pairs shortly as the *set of k-telescoped sequences of X and Y*:

$$\begin{aligned} X +_k Y &:= +_k(X \oplus_k Y) \\ &= \{x +_k y \mid x \in X, y \in Y \text{ are overlapping on } k \text{ elements}\}. \end{aligned}$$

Now let $S \subset R^\star$ be a finite set of such sequences (allowing multiplicities, modeling users taking the same paths). For an arbitrary sequence $x \in R^\star$ we denominate the relative frequency of sequences of S containing x as subsequence as *support of x with respect to S*:

$$\operatorname{sup}_S(x) := \frac{|\{s \in S \mid x \leq s\}|}{|S|}$$

The task of *searching all frequent subsequences* in the given set of sequences S means to find all sequences $x \in R^\star$ with at least a given minimal support, i.e. with $\operatorname{sup}_S(x) \geq \text{minsup}$ and $\text{minsup} \in \mathbb{R}^+$ a given constant. As the support of subsequences of a sequence is greater than or equal to the support of the sequence itself, one can build frequent subsequences recursively starting from the sequences of length $n = 1$. With all sequences of length 1 as initial *set of candidates* the algorithm performs two steps: first, it computes the support values of all candidates and selects those candidates as frequent subsequences that satisfy the minimal support constraint; second, it builds a new set of candidates of length $n + 1$ for the next step by trying to join frequent subsequences of length n in the following manner: two sequences c and d of length n are joined to a sequence of length $n+1$ if they overlap on $n-1$ elements, i.e. $(c_2, \ldots, c_n) = (d_1, \ldots, d_{n-1})$; the joined sequence is $c +_{n-1} d$. Algorithm 2 gives the formal description of this procedure. Please note, that for the special case of sequences describing paths on a graph, in the first join step only 0-overlapping pairs of sequences of length 1, i.e., pairs of nodes of the graph, have to be considered that are linked by an edge.—This adaption of the classical apriori algorithm for sets (see Agrawal and Srikant (1994)) to sequences has first been published by Agrawal and Srikant (1995) (with modifications by Srikant and Agrawal (1996)). It has been used for finding subsequences in web mining paths by Chen et al. (1996) and other authors afterwards (Viveros et al. (1997), Chen et al. (1998), Cooley et al. (1999a)).

Algorithm 2 Apriori algorithm adapted for sequences

```
Require: set of items $R$ (resources), list of (finite) sequences $S \subset R^\star$ (user paths),
  minimal support value minsup $\in \mathbb{R}^+$.
Ensure: set of frequent subsequences $F := \bigcup_{n \in \mathbb{N}} F_n$ of the sequences of $S$ with
  support of at least minsup.
  $C := \{\{r\} \mid r \in R\}$ set of initial candidates, $n := 1$.
  while $C \neq \emptyset$ do
    compute $\text{sup}_S(c)$  $\forall c \in C$ by counting the number of occurence of each $c$ in $S$
    (one loop through $S$).
    $F_n := \{c \in C \mid \text{sup}_S(c) \geq \text{minsup}\}$
    $C := F_n +_{n-1} F_n$ {compute new candidate sequences with length n+1}
    $n := n + 1$
  end while
```

4 Mining frequent generalized subsequences

By a *generalized sequence in R* we mean a (finite ordinary) sequence in the symbols $R \cup \{\star\}$ with an additional symbol $\star \notin R$ called *wildcard*, such that no two wildcards are adjacent:

$$R^{\text{gen}} := \{x \in (R \cup \{\star\})^\star \mid \nexists i \in \mathbb{N} : x_i = x_{i+1} = \star\}$$

The wildcard symbol $\star$ will be used to model partially indeterminate sequences, matching arbitrary subsequences. For a generalized sequence $x \in R^{\text{gen}}$ we define its *length* $|x|$ as the length of the sequence in the symbols $R \cup \{\star\}$, i.e. $|x| := n$, if $x \in (R \cup \{\star\})^n$. Now let $x, y \in R^{\text{gen}}$ be two generalized sequences. We say that x *matches* y or y *generalizes* x ($y \vdash x$), if there exists a mapping

$$m : \{1, \ldots, |x|\} \to \{1, \ldots, |y|\}$$

(called *matching*) with the following properties:

1. m maps indices of elements of x to indices of elements of y that coincide or to a wildcard ($y_{m(i)} = x_i$ or $y_{m(i)} = \star$).
2. m covers all indices of y of non-wildcard elements ($y_i \in R \Rightarrow m^{-1}(i) \neq \emptyset$).
3. m is weakly monotonic increasing.
4. m is even strictly monotonic at places where its image does not belong to a wildcard ($m(i) = m(i+1) \Rightarrow y_{m(i)} = \star$).

Please note that as the set of ordinary sequences $R^\star$ is a subset of the set of generalized sequences R^{gen}, this also defines the notion of an ordinary sequence matching a generalized sequence. Obviously matchings are not uniquely determined by two generalized sequences x and y. A trivial example is $\star A \star \vdash AA$ with the two matchings $m_1 = \{(1,1),(2,2)\}$ and $m_2 = \{(1,2),(2,3)\}$. Finally we carry over the notions of subsequence and of

k-telescoped concatenation from ordinary sequences to generalized sequences without any change: for two generalized sequences $x, y \in R^{\text{gen}}$ x is called *subsequence of y* ($x \leq y$), if there is an index $i \in \{0, \ldots, |y| - |x|\}$ with $x_j = y_{i+j} \quad \forall j = 1, \ldots, |x|$. The condition for subsequences is literal containment without any matching taking place. Note the difference between $A \star C$ not being a subsequence of $ABCD$ but matching a subsequence of it (i.e. $A \star C \vdash ABC$ and $ABC \leq ABCD$).

Again, let $S \subset R^*$ be a finite set of ordinary sequences (user paths). For an arbitrary generalized sequence $x \in R^{\text{gen}}$ we denominate the relative frequency of sequences containing a subsequence which matches x as *support of x with respect to S*:

$$\text{sup}_S(x) := |\{s \in S \mid \exists y \leq s : x \vdash y\}| / |S|$$

Now, *mining frequent generalized subsequences* is the label for the task to find all generalized sequences with at least a given minimal support. As we are looking for subsequences anyway, we can narrow our view to *closed generalized subsequences*, i.e. generalized subsequences without leading or trailing wildcard ($x \in R^{\text{gen}}$ with $x_1, x_{\text{last}} \in R$).

Up to now no general algorithm for finding all frequent generalized subsequences in a set of sequences is known. Spiliopoulou (1999) has invented an algorithm for finding frequent generalized subsequences in a limited subspace of the search space: her generalized sequence miner (GSM) looks only for generalized sequences of a given length and wildcards at given positions (such subspaces are described by so called *templates*; see Buechner et al. (1999) for another approach using templates to limit the search space; templates are useful in the framework of interactive tools like WUM, see Spiliopoulou and Faulstich (1998) and Spiliopoulou et al. (1999)).

We present a modification of the apriori algorithm for sequences to generalized sequences, resulting in a general algorithm for finding frequent generalized subsequences. The idea is pretty simple. As we are looking only at closed generalized sequences, the support of any subsequence of such a closed generalized sequence again is greater than or equal to the support of the sequence itself. Now, as adjacent wildcards are not allowed in generalized sequences, we can get every generalized sequence of length $n+1$ (for $n \geq 3$) as junction of two overlapping closed generalized sequences of the kind described in table 2. Thus, we only have to modify the join step of the apriori algorithm for building new candidates of length $n+1$ in such a way that we not only use the frequent (closed generalized) subsequences of length n but also those of length $n-1$ from the step before, and try all possible combinations. Closed generalized subsequences of length 3 containing a wildcard have the form $(x, \star, y)$ with $x, y \in R$, shorter closed generalized subsequences cannot contain wildcards.

Algorithm 3 gives the exact formulation of the necessary comparisions. Of course, the computation of the support values of the candidate generalized sequences also has to be modified.

	sequence	length		sequence	length
	ab ... cd	n+1		a⋆b ... cd	n+1
=	ab ... c	n	=	a⋆b ... c	n
$+_{n-1}$	b ... cd	n	$+_{n-2}$	b ... cd	n-1
	ab ... c⋆d	n+1		a⋆b ... c⋆d	n+1
=	ab ... c	n-1	=	a⋆b ... c	n-1
$+_{n-2}$	b ... c⋆d	n	$+_{n-3}$	b ... c⋆d	n-1

Table 2. Construction of closed generalized subsequences of length ≥ 4.

Algorithm 3 Apriori algorithm adapted for generalized sequences

```
Require: set of items $R$ (resources), list of (finite) sequences $S \subset R^*$ (user paths),
    minimal support value minsup $\in \mathbb{R}^+$.
Ensure: set of frequent (closed) generalized subsequences $F := \bigcup_{n \in \mathbb{N}} F_n$ of the
    sequences of $S$ with support of at least minsup.
  $C := \{\{r\} \mid r \in R\}$ set of initial candidates, $n := 1$.
  while $C \neq \emptyset$ do
    compute $\sup_S(c)$ $\forall c \in C$ by counting the number of occurence of each $c$ in $S$
    (one loop through $S$).
    $F_n := \{c \in C \mid \sup_S(c) \geq \text{minsup}\}$
    $C := F_n +_{n-1} F_n$ {compute new candidate sequences with length n+1}
    if $n = 2$ then {introduce wildcards}
      $C := C \cup \{(x, \star, y) \mid x, y \in F_{n-1}\}$
    else if $n > 2$ then {additional joins considering wildcards}
      $C := C \cup \{x +_{n-2} y \mid (x, y) \in F_n \oplus_{n-2} F_{n-1}, x_2 = \star\}$
               $\cup \{x +_{n-2} y \mid (x, y) \in F_{n-1} \oplus_{n-2} F_n, y_{\text{last}-1} = \star\}$
               $\cup \{x +_{n-3} y \mid (x, y) \in F_{n-1} \oplus_{n-3} F_{n-1}, x_2 = y_{\text{last}-1} = \star\}$
    end if
    $n := n + 1$
  end while
```

As algorithms of the apriori type return all subsequences of the frequent sequences found, one often prunes the result set by removing all subsequences of a frequent sequence contained in the result set, thus retaining only the "maximal" subsequences:

$$F := \{c \in F \mid \nexists d \in F : c < d\}$$

For generalized subsequences the algorithm also returns all generalizations of all subsequences found. Reasonably one prunes the result set further, by removing all generalizations of a sequence contained in the result set, thus retaining only the "most concrete" subsequences:

$$F := \{c \in F \mid \nexists d \in F : c \vdash d\}$$

We call these two pruning steps *subsequence pruning* and *generalization pruning*, respectively.

5 Example

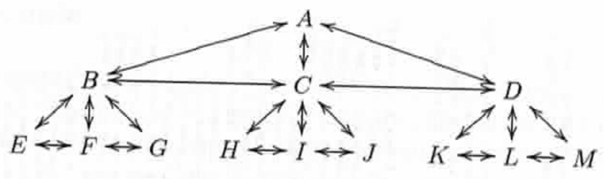
Figure 3 shows an example logfile with only 46 entries (because of space restrictions), figure 4 the site graph of the underlying website and an analysis of the paths in the logfile (only the first four path are from the logfile fragment shown). As one can see, the first path is started by a client with ip-address 129.13.122.23 and an agent called Konqueror. At 16:53:45 a second path is started by a client with ip 193.197.80.33 and agent Mozilla coming from a search engine (Altavista) which looks for the keyword "webmining". At 17:02:21 a third path is started by a client with the same ip-address and agent name as the client of the second path, so it could be very well the same user; the path is marked by a missing referrer information. Then, at 17:03:32 a fourth path is started, again by a client with the same ip-address; but this time the agent name is different, so probably the fourth path is taken by another user. Retrieving the DNS-name of ip-address 193.197.80.33 in fact reveals, that it is a proxy. Most auxiliary resources already have been removed beforehand; e.g., one can see a stylesheet page.css showing up each time page A.xml is requested.

Figure 4 shows the result of the path construction of the four paths from the logfile fragment shown in figure 3 together with eight further paths constructed via path completion by backtracking as well as via path completion by shortest path. Looking for ordinary frequent subsequences by applying the apriori algorithm for sequences (algorithm 2) does not give very useful results here: using the paths reconstructed by backtracking, one finds the sequences CHI with a support of 8/12 and BCH with a support of 7/12. The first sequences containing more than three resources appear at support 4/12: BCHI and EBCH. Using the paths reconstructed by shortest path completion one also finds the two sequences CHI and BCH of length 3 (BCH only with support 6/12), but now longer sequences appear at support 3/12, e.g., the sequence BCHI, this time accompanied by a bunch of other sequences (ABEF, BCHC, CHIJ, BEFGB). This demonstrates that the results of sequence mining depends on the navigation path completion scheme applied beforehand.

Searching for frequent generalized sequences with algorithm 3 results in the same set of three sequences with high support regardless of the path completion heuristics used: B⋆C⋆H⋆I with support 12/12 and two lightly more specialized sequences B⋆CH⋆I and BC⋆H⋆I with support 11/12 and 10/12 respectively (the latter is 9/12 for the paths reconstructed using shortest path completion). Of course, the algorithm finds all literal subsequences of these sequences as well as all more general sequences (like B⋆H⋆I etc.), but these less useful subsequences are pruned by the two pruning steps (subsequence pruning and generalization pruning) presented at the end of section 4.—Already this simple example gives some insights into the good properties of generalized sequences: first, they are more robust than ordinary sequences against artefacts coming from navigation path construction steps; second,

```
129.13.122.23 - - [18/Apr/2000:16:48:03 +0200] "GET /A.xml HTTP/1.0" 200 1258 "-" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:16:48:04 +0200] "GET /page.css HTTP/1.0" 200 323 "http://etupc23.wiwi.uni-karlsruhe.de/A.xml" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:16:50:32 +0200] "GET /B.xml HTTP/1.0" 200 2044 "http://etupc23.wiwi.uni-karlsruhe.de/A.xml" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:16:51:06 +0200] "GET /E.xml HTTP/1.0" 200 1332 "http://etupc23.wiwi.uni-karlsruhe.de/B.xml" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:16:51:07 +0200] "GET /z.png HTTP/1.0" 200 1637 "http://etupc23.wiwi.uni-karlsruhe.de/E.xml" "Konqueror/1.1.2"
193.197.80.33 - - [18/Apr/2000:16:53:45 +0200] "GET /A.xml HTTP/1.0" 200 1258 "http://www.altavista.de/cgi-bin/query?q=webmining" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14
i686)"
193.197.80.33 - - [18/Apr/2000:16:53:46 +0200] "GET /page.css HTTP/1.0" 200 323 "http://etupc23.wiwi.uni-karlsruhe.de/A.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
129.13.122.23 - - [18/Apr/2000:16:55:24 +0200] "GET /F.xml HTTP/1.0" 200 1878 "http://etupc23.wiwi.uni-karlsruhe.de/E.xml" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:16:56:34 +0200] "GET /C.xml HTTP/1.0" 200 906 "http://etupc23.wiwi.uni-karlsruhe.de/F.xml" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:16:56:35 +0200] "GET /x.png HTTP/1.0" 200 1332 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Konqueror/1.1.2"
193.197.80.33 - - [18/Apr/2000:16:58:02 +0200] "GET /C.xml HTTP/1.0" 200 906 "http://etupc23.wiwi.uni-karlsruhe.de/A.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:16:58:03 +0200] "GET /x.png HTTP/1.0" 200 1332 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
129.13.122.23 - - [18/Apr/2000:17:00:21 +0200] "GET /H.xml HTTP/1.0" 200 2439 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:17:00:21 +0200] "GET /page.css HTTP/1.0" 200 323 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Konqueror/1.1.2"
193.197.80.33 - - [18/Apr/2000:17:02:21 +0200] "GET /B.xml HTTP/1.0" 200 2044 "-" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:03:32 +0200] "GET /A.xml HTTP/1.1" 200 1258 "http://marketing.wiwi.uni-karlsruhe.de/research.html" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:03:33 +0200] "GET /page.css HTTP/1.1" 200 323 "http://etupc23.wiwi.uni-karlsruhe.de/A.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:10:43 +0200] "GET /C.xml HTTP/1.0" 200 906 "http://etupc23.wiwi.uni-karlsruhe.de/B.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:10:44 +0200] "GET /x.png HTTP/1.0" 200 1332 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
129.13.122.23 - - [18/Apr/2000:17:12:17 +0200] "GET /I.xml HTTP/1.0" 200 1240 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Konqueror/1.1.2"
129.13.122.23 - - [18/Apr/2000:17:12:26 +0200] "GET /J.xml HTTP/1.0" 200 1646 "http://etupc23.wiwi.uni-karlsruhe.de/I.xml" "Konqueror/1.1.2"
193.197.80.33 - - [18/Apr/2000:17:12:58 +0200] "GET /B.xml HTTP/1.0" 2044 1258 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:14:03 +0200] "GET /E.xml HTTP/1.0" 200 1332 "http://etupc23.wiwi.uni-karlsruhe.de/B.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:14:04 +0200] "GET /z.png HTTP/1.0" 200 1637 "http://etupc23.wiwi.uni-karlsruhe.de/E.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:16:22 +0200] "GET /J.xml HTTP/1.0" 200 1646 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:17:12 +0200] "GET /H.xml HTTP/1.0" 200 2439 "http://etupc23.wiwi.uni-karlsruhe.de/J.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:17:13 +0200] "GET /page.css HTTP/1.0" 200 323 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:19:55 +0200] "GET /B.xml HTTP/1.1" 200 2044 "http://etupc23.wiwi.uni-karlsruhe.de/A.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:20:34 +0200] "GET /G.xml HTTP/1.1" 200 1473 "http://etupc23.wiwi.uni-karlsruhe.de/B.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:21:13 +0200] "GET /H.xml HTTP/1.0" 200 2439 "http://etupc23.wiwi.uni-karlsruhe.de/E.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:21:14 +0200] "GET /page.css HTTP/1.0" 200 323 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:23:52 +0200] "GET /I.xml HTTP/1.0" 200 1240 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:25:35 +0200] "GET /E.xml HTTP/1.1" 200 1332 "http://etupc23.wiwi.uni-karlsruhe.de/G.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:25:36 +0200] "GET /z.png HTTP/1.1" 200 1637 "http://etupc23.wiwi.uni-karlsruhe.de/E.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:27:28 +0200] "GET /C.xml HTTP/1.1" 200 906 "http://etupc23.wiwi.uni-karlsruhe.de/E.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:27:29 +0200] "GET /x.png HTTP/1.1" 200 1332 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:29:32 +0200] "GET /D.xml HTTP/1.0" 200 2760 "http://etupc23.wiwi.uni-karlsruhe.de/I.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:29:32 +0200] "GET /y.png HTTP/1.0" 200 1070 "http://etupc23.wiwi.uni-karlsruhe.de/D.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:29:33 +0200] "GET /z.png HTTP/1.0" 200 1637 "http://etupc23.wiwi.uni-karlsruhe.de/D.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:31:02 +0200] "GET /I.xml HTTP/1.0" 200 1240 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Mozilla/4.72 [en] (X11; I; Linux 2.2.14 i686)"
193.197.80.33 - - [18/Apr/2000:17:33:21 +0200] "GET /H.xml HTTP/1.1" 200 2439 "http://etupc23.wiwi.uni-karlsruhe.de/C.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:33:22 +0200] "GET /page.css HTTP/1.1" 200 323 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:34:22 +0200] "GET /I.xml HTTP/1.1" 200 1240 "http://etupc23.wiwi.uni-karlsruhe.de/H.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:35:47 +0200] "GET /D.xml HTTP/1.1" 200 2760 "http://etupc23.wiwi.uni-karlsruhe.de/I.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:35:47 +0200] "GET /y.png HTTP/1.1" 200 1070 "http://etupc23.wiwi.uni-karlsruhe.de/D.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
193.197.80.33 - - [18/Apr/2000:17:35:47 +0200] "GET /z.png HTTP/1.1" 200 1637 "http://etupc23.wiwi.uni-karlsruhe.de/D.xml" "Mozilla/4.0 (MSIE 5.01; Windows NT)"
```

Fig. 3. Example logfile.



(a) site graph

nr	reconstructed paths (by backtracking)	(by shortest path)
1	ABEF(EB)CHIJ	ABEF(B)CHIJ
2	ACBE(BC)HI(HC)D	ACBE(C)HI(C)D
3	BCJ(C)HI	BCJ(C)HI
4	ABG(B)E(B)CH(C)I(C)D	ABG(B)E(B)CH(C)I(C)D
5	ABEFG(FEB)CH(C)JI	ABEFG(B)CH(C)JI
6	ACJ(C)D(C)B(C)HI	ACJ(C)D(C)B(C)HI
7	BEFG(FEB)CHIJ(IHC)DKLM	BEFG(B)CHIJ(C)DKLM
8	ABF(B)CIH(I)J	ABF(B)CIH(I)J
9	ADK(D)L(D)AB(A)CHI	ADK(D)L(D)AB(A)CHI
10	ABEFG(FEBA)CJ(C)HI(HC)D	ABEFG(BA)CJ(C)HI(C)D
11	ABCD(C)HIJ(IHCD)M	ABCD(C)HIJ(CD)M
12	CBF(BC)H(C)DK(DC)I(CKDCHCB)E	CBF(BC)H(C)DK(DC)I(CB)E

(b) anaylzed paths

Fig. 4. Example web site and example set of paths analyzed with help of the two different path completion heuristics from section 2.

they can cope with local deviations of the navigation paths, thus resulting in longer paths with higher support values, i.e. they better sketch user navigational behaviour in the large, contrary to local descriptions by ordinary subsequences.

6 Applications to association rules and outlook

In web mining contexts frequent generalized subsequences may be of interest on their own, as they give an idea about the navigation paths users take on a website.

Additionally, the retrieval of frequent (generalized) subsequences is the hard part of the generation of association rules. An *association rule* is (described by) a pair of (generalized) sequences $x, y \in R^{\text{gen}}$ with the meaning that if x (the body of the rule) has occured then—under conditions to be explained in the following—y (the head of the rule) will occur, too, where occurance is related to the underlying set S of sequences. We suggest different interpretations of the rule notation that all have their origin in the web site traversal behaviour of users as reconstructed via path completion and depicted in S. First, $x \to y \cong x +_1 y = (x_1, \ldots, x_{\text{last}} = y_1, \ldots, y_{\text{last}})$ which best corresponds to the usage of ordinary navigation paths. Second, $x \rightsquigarrow y \cong x \star y = (x_1, \ldots, x_{\text{last}}, \star, y_1, \ldots, y_{\text{last}})$, i.e., a wildcard is used to combine x and y. Both cases can be handled with the tools described so far. In addition to

$$\text{sup}_S(x \to y) := \text{sup}_S(x +_1 y) \quad \text{or} \quad \text{sup}_S(x \rightsquigarrow y) := \text{sup}_S(x \star y)$$

we need the confidence

$$\text{conf}_S(x \to y) := \frac{\text{sup}_S(x +_1 y)}{\text{sup}_S(x)} \quad \text{or} \quad \text{conf}_S(x \rightsquigarrow y) := \frac{\text{sup}_S(x \star y)}{\text{sup}_S(x)}$$

a number that counts the occurence of $x +_1 y$ or $x \star y$ given x.

From early papers on web usage mining, the idea of feeding back the usage information extracted from the logfiles to the hyperlink structure of the underlying website has been suggested as an application of the results found by various data analysis tasks (see Yan et al. (1996)). Recently this idea has been revived by the name of *recommender system* making use of frequent item sets and association rules (see Mobasher (2000)). The paths of active users are compared to the left sides (the bodies) of a rule set previously extracted from the logfiles and (parts of) the right sides (the heads) of the matching rules with highest confidence are recommended via dynamically included direct hyperlinks.

Using only sets of resources (and not sequences) as the base for recommendations has the drawback of neglecting the order of the navigation patterns, and, thus, may result in directing users back to resources, they might no longer have an interest in. On the other hand, using ordinary subsequences as base for recommendations retains the order information, but only catches local navigational behaviour. Generalized subsequences combine the strengths of the two methods, retaining order and not being bound to local behaviour (by allowing deviations).

Let us go back to the example from the previous section and look at user 9. Imagine he has already done ADK(D)L(D)AB(A)C. Using frequent ordinary subsequences we cannot recommend a next resource, because no subsequence of the frequent literal subsequences (CHI, BCH, BCHI and EBCH) can be found in his partial path. But the subsequence B$\star$C of the frequent generalized subsequence B$\star$C$\star$H$\star$I matches a subsequence of the tail B(A)C of his partial path. Thus, using the assocation rules B$\star$C$\rightsquigarrow$H and B$\star$C$\rightsquigarrow$I (both with support and confidence 1) we can recommend H and I for subsequent browsing, exactly the resources, he visits afterwards.

At the end, different outlooks are possible. One can focus on recommender systems and discuss the degree to which web mining results of the kind presented in this paper should be incorporated in such tools. A more theoretically oriented outlook can stress further algorithmic aspects, e.g., of how to cluster subsets of sequences that show different kinds of navigational behaviour.

References

AGRAWAL, R. and SRIKANT, R. (1994): Fast Algorithms for Mining Association Rules. In: Bocca, J.B., Jarke, M., and Zaniolo, C. (eds.): *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, September 12-15, 1994, Santiago de Chile, Morgan Kaufmann, Chile, 487–499.

AGRAWAL, R. and SRIKANT, R. (1995): Mining Sequential Patterns. In: Yu, P.S., and Chen, A.L.P. (eds.): *Proceedings of the Eleventh International Conference on Data Engineering*, March 6-10, 1995, Taipei, Taiwan, IEEE Computer Society, 3–14.

BOCK, H.H. (1974): Automatische Klassifikation, Theoretische und praktische Methoden zur Gruppierung und Strukturierung von Daten (Clusteranalyse), Vandenhoeck & Ruprecht, Göttingen.

BORGES, J. and LEVENE, M. (1998): Mining Association Rules in Hypertext Databases. In: Agrawal, R. (ed.): *Proceedings / The Fourth International Conference on Knowledge Discovery and Data Mining*, August 27 - 31, 1998, New York, Menlo Park, Calif., 149–153.

BORGES, J. and LEVENE, M. (1999a): Mining Navigation Patterns with Hypertext Probabilistic Grammars. Research Note RN/99/08, Department of Computer Science, University College London, February 1999.

BORGES, J. and LEVENE, M. (1999b): Data Mining of User Navigation Patterns. In: *Proceedings of the Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, August 15, 1999, San Diego, CA, Springer, 31–36.

BUECHNER, A.G., BAUMGARTEN, M., ANAND, S.S., MULVENNA, M.D., and HUGHES, J.G. (1999): Navigation Pattern Discovery from Internet Data. In: *Proceedings of the Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, August 15, 1999, San Diego, CA, Springer, 25-30.

CHEN, M.-S., PARK, J.S., and YU, P.S. (1996): Data Mining for Path Traversal Patterns in a Web Environment. In: *Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS)*, May 27-30, 1996, Hong Kong, IEEE Computer Society, 385–392.

CHEN, M.-S., PARK, J.S., and YU, P.S. (1998): Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge & Data Engineering 10/2 (1998)*, 209–221.

COOLEY, R., MOBASHER, B., and SRIVASTAVA, J. (1999a): Web Mining: Information and Pattern Discovery on the World Wide Web. In: *9th International Conference on Tools with Artificial Intelligence (ICTAI '97)*, November 3-8, 1997, Newport Beach, CA.

COOLEY, R., MOBASHER, B., and SRIVASTAVA, J. (1999b): Data Preparation for Mining World Wide Web Browsing Patterns. *Journal of Knowledge and Information Systems 1/1 (1999)*.

GAUL, W. and SCHADER, M. (1999): Data Mining: A New Label for an Old Problem? in: Gaul, W. and Schader, M. (Hrsg.): *Mathematische Methoden der Wirtschaftswissenschaft*, Festschrift für Otto Opitz, Physica-Verlag, Heidelberg, 3–14.

MOBASHER, B. (2000): Mining Web Usage Data for Automatic Site Personalization. To appear in *Studies in Classification, Data Analysis, and Knowledge Organization, 2000.*

SHAHABI, C., ZARKESH, A.M., ADIBI, J., and SHAH, V. (1997): Knowledge Discovery from Users Web-Page Navigation. In: *7th International Workshop on Research Issues in Data Engineering (RIDE '97), High Performance Database Management for Large-Scale Applications,* April 7-8, 1997, Birmingham, UK.

SPILIOPOULOU, M. and FAULSTICH, L.C. (1998): WUM: A Tool for Web Utiliziation Analysis. In: Atzeni, P., Mendelzon, A., and Mecca, G. (eds.): *The World Wide Web and Databases, International Workshop WebDB'98,* Valencia, Spain, March 27-28, 1998, LNCS 1590, Springer, 184–203.

SPILIOPOULOU, M., FAULSTICH, L.C., and WINKLER, K. (1999): A Data Miner Analyzing the Navigational Behaviour of Web Users. In: *Proc. of the Workshop on Machine Learning in User Modelling of the ACAI'99 Int. Conf.,* Creta, Greece, July 1999.

SPILIOPOULOU, M. (1999): The Laborious Way from Data Mining to Web Mining. *Int. Journal of Comp. Sys., Sci. & Eng. 14 (1999),* Special Issue on "Semantics of the Web", 113-126.

SRIKANT, R. and and AGRAWAL, R. (1996): Mining Sequential Patterns: Generalizations and Performance Improvements. In: Apers, P.M.G., Bouzeghoub, M., and Gardarin, G. (eds.): *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology,* Avignon, France, March 25-29, 1996, Proceedings. LNCS 1057, Springer.

VIVEROS, M.S., ELO-DEAN, S., WRIGHT, M.A., and DURI, S.S. (1997): Visitor's Behaviour: Mining Web Servers. In: *Proceedings of the 1st International Conference on the Practical Application of Knowledge Discovery and Data Mining,* Blackpool 1997, 257–269.

YAN, T.W., JACOBSEN, M., GARCIA-MOLINA, H., and DAYAL, U. (1996): From User Access Patterns to Dynamic Hypertext Linking. In: *Fifth International World Wide Web Conference* May 6-10, 1996, Paris, France.

ZAIANE, O.R., XIN, M., and HAN, J. (1998): Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In: *Advances in Digital Libraries,* Santa Barbara 1998, 19–29.