12 Web Usage Mining

By *Bamshad Mobasher*

With the continued growth and proliferation of e-commerce, Web services, and Web-based information systems, the volumes of **clickstream** and **user data** collected by Web-based organizations in their daily operations has reached astronomical proportions. Analyzing such data can help these organizations determine the life-time value of clients, design cross-marketing strategies across products and services, evaluate the effectiveness of promotional campaigns, optimize the functionality of Web-based applications, provide more personalized content to visitors, and find the most effective logical structure for their Web space. This type of analysis involves the automatic discovery of meaningful patterns and relationships from a large collection of primarily semi-structured data, often stored in Web and applications server access logs, as well as in related operational data sources.

Web usage mining refers to the automatic discovery and analysis of patterns in clickstream and associated data collected or generated as a result of user interactions with Web resources on one or more Web sites [114, 387, 505]. The goal is to capture, model, and analyze the behavioral patterns and profiles of users interacting with a Web site. The discovered patterns are usually represented as collections of pages, objects, or resources that are frequently accessed by groups of users with common needs or interests.

Following the standard data mining process [173], the overall Web usage mining process can be divided into three inter-dependent stages: data collection and pre-processing, pattern discovery, and pattern analysis. In the pre-processing stage, the clickstream data is cleaned and partitioned into a set of user transactions representing the activities of each user during different visits to the site. Other sources of knowledge such as the site content or structure, as well as semantic domain knowledge from site **ontologies** (such as **product catalogs** or **concept hierarchies**), may also be used in pre-processing or to enhance user transaction data. In the pattern discovery stage, statistical, database, and machine learning operations are performed to obtain hidden patterns reflecting the typical behavior of users, as well as summary statistics on Web resources, sessions, and users. In the final stage of the process, the discovered patterns and statistics are further processed, filtered, possibly resulting in aggregate user models that can be

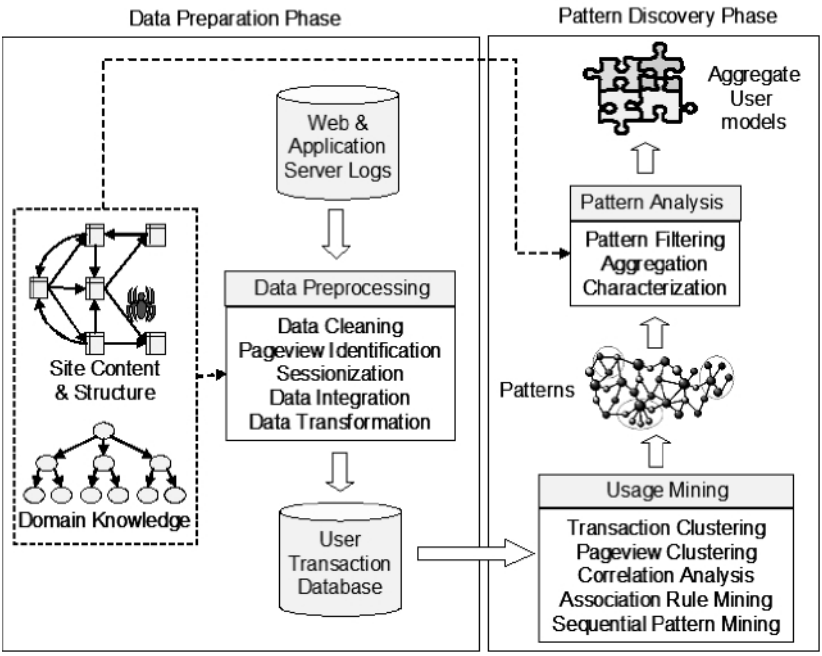


Fig. 12.1. The Web usage mining process

used as input to applications such as **recommendation engines**, visualization tools, and Web analytics and report generation tools. The overall process is depicted in Fig. 12.1.

In the remainder of this chapter, we provide a detailed examination of Web usage mining as a process, and discuss the relevant concepts and techniques commonly used in all the various stages mentioned above.

12.1 Data Collection and Pre-Processing

An important task in any data mining application is the creation of a suitable target data set to which data mining and statistical algorithms can be applied. This is particularly important in Web usage mining due to the characteristics of clickstream data and its relationship to other related data collected from multiple sources and across multiple channels. The data preparation process is often the most time consuming and computationally intensive step in the Web usage mining process, and often requires the use of special algorithms and heuristics not commonly employed in other domains. This process is critical to the successful extraction of useful patterns

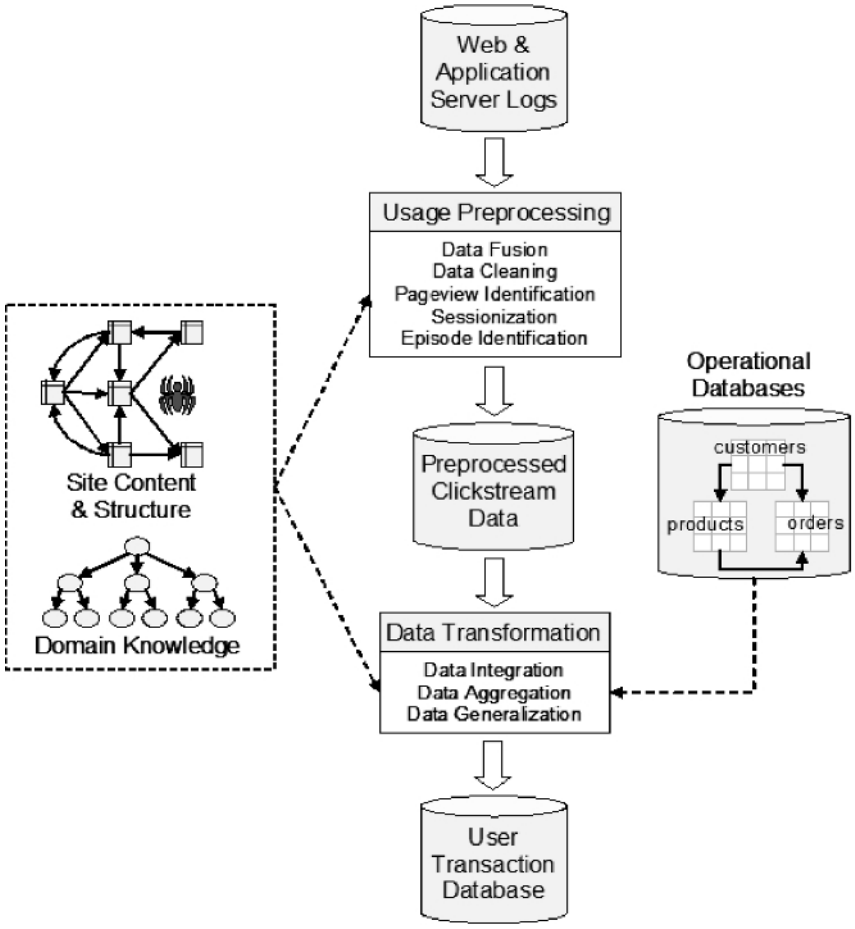


Fig. 12.2. Steps in data preparation for Web usage mining.

from the data. The process may involve pre-processing the original data, integrating data from multiple sources, and transforming the integrated data into a form suitable for input into specific data mining operations. Collectively, we refer to this process as *data preparation*.

Much of the research and practice in usage data preparation has been focused on pre-processing and integrating these data sources for different analysis. Usage data preparation presents a number of unique challenges which have led to a variety of algorithms and heuristic techniques for preprocessing tasks such as data fusion and cleaning, user and session identification, pageview identification [115]. The successful application of data mining techniques to Web usage data is highly dependent on the correct application of the pre-processing tasks. Furthermore, in the context of e-

commerce data analysis, these techniques have been extended to allow for the discovery of important and insightful user and site metrics [286].

Figure 12.2 provides a summary of the primary tasks and elements in usage data pre-processing. We begin by providing a summary of data types commonly used in Web usage mining and then provide a brief discussion of some of the primary data preparation tasks.

12.1.1 Sources and Types of Data

The primary data sources used in Web usage mining are the **server log** files, which include **Web server access logs** and **application server logs**. Additional data sources that are also essential for both data preparation and pattern discovery include the site files and meta-data, operational databases, application templates, and domain knowledge. In some cases and for some users, additional data may be available due to client-side or proxy-level (Internet Service Provider) data collection, as well as from external clickstream or **demographic data** sources such as those provided by data aggregation services from ComScore (www.comscore.com), NetRatings (www.nielsen-netratings.com), and Acxiom (www.acxiom.com).

The data obtained through various sources can be categorized into four primary groups [115, 505].

Usage Data: The log data collected automatically by the Web and application servers represents the fine-grained navigational behavior of visitors. It is the primary source of data in Web usage mining. Each hit against the server, corresponding to an HTTP request, generates a single entry in the server access logs. Each log entry (depending on the log format) may contain fields identifying the time and date of the request, the IP address of the client, the resource requested, possible parameters used in invoking a Web application, status of the request, HTTP method used, the user agent (browser and operating system type and version), the referring Web resource, and, if available, client-side **cookies** which uniquely identify a repeat visitor. A typical example of a server access log is depicted in Fig. 12.3, in which six partial log entries are shown. The user IP addresses in the log entries have been changed to protect privacy.

For example, log entry 1 shows a user with IP address "1.2.3.4" accessing a resource: "/classes/cs589/papers.html" on the server (maya.cs.depaul.edu). The browser type and version, as well as operating system information on the client machine are captured in the agent field of the entry. Finally, the referrer field indicates that the user came to this location from an outside source: "http://dataminingresources.blogspot.com/". The next log entry shows that this user has navigated from "papers.html" (as re-

1	2006-02-01 00:08:43 1.2.3.4 - GET /classes/cs589/papers.html - 200 9221 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+2.0.50727) http://dataminingresources.blogspot.com/
2	2006-02-01 00:08:46 1.2.3.4 - GET /classes/cs589/papers/cms-tai.pdf - 200 4096 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+2.0.50727) http://maya.cs.depaul.edu/~classes/cs589/papers.html
3	2006-02-01 08:01:28 2.3.4.5 - GET /classes/ds575/papers/hyperlink.pdf - 200 318814 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1) http://www.google.com/search?hl=en&lr=&q=hyperlink+analysis+for+the+web+survey
4	2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/announce.html - 200 3794 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1) http://maya.cs.depaul.edu/~classes/cs480/
5	2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/styles2.css - 200 1636 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1) http://maya.cs.depaul.edu/~classes/cs480/announce.html
6	2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/header.gif - 200 6027 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1) http://maya.cs.depaul.edu/~classes/cs480/announce.html

Fig. 12.3. Portion of a typical server log

flected in the referrer field of entry 2) to access another resource: "/classes/cs589/papers/cms-tai.pdf". Log entry 3 shows a user who has arrived at the resource "/classes/ds575/papers/hyperlink.pdf" by doing a search on Google using keyword query: "hyperlink analysis for the web survey". Finally, entries 4–6 all correspond to a single click-through by a user who has accessed the resource "/classes/cs480/announce.html". Entries 5 and 6 are images embedded in the file "announce.html" and thus two additional HTTP request are registered as hits in the server log corresponding to these images.

Depending on the goals of the analysis, this data needs to be transformed and aggregated at different levels of abstraction. In Web usage mining, the most basic level of data abstraction is that of a **pageview**. A pageview is an aggregate representation of a collection of Web objects contributing to the display on a user's browser resulting from a single user action (such as a click-through). Conceptually, each pageview can be viewed as a collection of Web objects or resources representing a specific "user event," e.g., reading an article, viewing a product page, or adding a product to the shopping cart. At the user level, the most basic level of behavioral abstraction is that of a **session**. A session is a sequence of pageviews by a single user during a single visit. The notion of a session can be

further abstracted by selecting a subset of pageviews in the session that are significant or relevant for the analysis tasks at hand.

Content Data: The content data in a site is the collection of objects and relationships that is conveyed to the user. For the most part, this data is comprised of combinations of textual materials and images. The data sources used to deliver or generate this data include static HTML/XML pages, multimedia files, dynamically generated page segments from scripts, and collections of records from the operational databases. The site content data also includes semantic or structural meta-data embedded within the site or individual pages, such as descriptive keywords, document attributes, semantic tags, or HTTP variables. The underlying domain ontology for the site is also considered part of the content data. Domain ontologies may include conceptual hierarchies over page contents, such as product categories, explicit representations of semantic content and relationships via an ontology language such as RDF, or a database schema over the data contained in the operational databases.

Structure Data: The structure data represents the designer's view of the content organization within the site. This organization is captured via the inter-page linkage structure among pages, as reflected through hyperlinks. The structure data also includes the intra-page structure of the content within a page. For example, both HTML and XML documents can be represented as tree structures over the space of tags in the page. The hyperlink structure for a site is normally captured by an automatically generated "site map." A site mapping tool must have the capability to capture and represent the inter- and intra-pageview relationships. For dynamically generated pages, the site mapping tools must either incorporate intrinsic knowledge of the underlying applications and scripts that generate HTML content, or must have the ability to generate content segments using a sampling of parameters passed to such applications or scripts.

User Data: The operational database(s) for the site may include additional user profile information. Such data may include demographic information about registered users, user ratings on various objects such as products or movies, past purchases or visit histories of users, as well as other explicit or implicit representations of users' interests. Some of this data can be captured anonymously as long as it is possible to distinguish among different users. For example, anonymous information contained in client-side cookies can be considered a part of the users' profile information, and used to identify repeat visitors to a site. Many personalization applications require the storage of prior user profile information.

12.1.2 Key Elements of Web Usage Data Pre-Processing

As noted in Fig. 12.2, the required high-level tasks in usage data pre-processing include the fusion and synchronization of data from multiple log files, data cleaning, pageview identification, user identification, session identification (or sessionization), episode identification, and the integration of clickstream data with other data sources such as content or semantic information, as well as user and product information from operational databases. We now examine some of the essential tasks in pre-processing.

Data Fusion and Cleaning

In large-scale Web sites, it is typical that the content served to users comes from multiple Web or application servers. In some cases, multiple servers with redundant content are used to reduce the load on any particular server. Data fusion refers to the merging of log files from several Web and application servers. This may require global synchronization across these servers. In the absence of shared embedded session ids, heuristic methods based on the "referrer" field in server logs along with various sessionization and user identification methods (see below) can be used to perform the merging. This step is essential in "inter-site" Web usage mining where the analysis of user behavior is performed over the log files of multiple related Web sites [513].

Data cleaning is usually site-specific, and involves tasks such as, removing extraneous references to embedded objects that may not be important for the purpose of analysis, including references to style files, graphics, or sound files. The cleaning process also may involve the removal of at least some of the data fields (e.g. number of bytes transferred or version of HTTP protocol used, etc.) that may not provide useful information in analysis or data mining tasks.

Data cleaning also entails the removal of references due to crawler navigations. It is not uncommon for a typical log file to contain a significant (sometimes as high as 50%) percentage of references resulting from search engine or other crawlers (or spiders). Well-known search engine crawlers can usually be identified and removed by maintaining a list of known crawlers. Other "well-behaved" crawlers which abide by standard robot exclusion protocols, begin their site crawl by first attempting to access to exclusion file "robots.txt" in the server root directory. Such crawlers, can therefore, be identified by locating all sessions that begin with an (attempted) access to this file. However, a significant portion of crawlers references are from those that either do not identify themselves explicitly (e.g., in the "agent" field) or implicitly; or from those crawlers that delib-

erately masquerade as legitimate users. In this case, identification and removal of crawler references may require the use of heuristic methods that distinguish typical behavior of Web crawlers from those of actual users. Some work has been done on using classification algorithms to build models of crawlers and Web robot navigations [510], but such approaches have so far been met with only limited success and more work in this area is required.

Pageview Identification

Identification of pageviews is heavily dependent on the intra-page structure of the site, as well as on the page contents and the underlying site domain knowledge. Recall that, conceptually, each pageview can be viewed as a collection of Web objects or resources representing a specific "user event," e.g., clicking on a link, viewing a product page, adding a product to the shopping cart. For a static single frame site, each HTML file may have a one-to-one correspondence with a pageview. However, for multi-framed sites, several files make up a given pageview. For dynamic sites, a pageview may represent a combination of static templates and content generated by application servers based on a set of parameters.

In addition, it may be desirable to consider pageviews at a higher level of aggregation, where each pageview represents a collection of pages or objects, for examples, pages related to the same concept category. In e-commerce Web sites, pageviews may correspond to various product-oriented events, such as product views, registration, shopping cart changes, purchases, etc. In this case, identification of pageviews may require *a priori* specification of an "event model" based on which various user actions can be categorized.

In order to provide a flexible framework for a variety of data mining activities a number of attributes must be recorded with each pageview. These attributes include the pageview id (normally a URL uniquely representing the pageview), static pageview type (e.g., information page, product view, category view, or index page), and other metadata, such as content attributes (e.g., keywords or product attributes).

User Identification

The analysis of Web usage does not require knowledge about a user's identity. However, it is necessary to distinguish among different users. Since a user may visit a site more than once, the server logs record multiple sessions for each user. We use the phrase **user activity record** to refer to the sequence of logged activities belonging to the same user.

In the absence of authentication mechanisms, the most widespread approach to distinguishing among unique visitors is the use of client-side cookies. Not all sites, however, employ cookies, and due to privacy concerns, client-side cookies are sometimes disabled by users. IP addresses, alone, are not generally sufficient for mapping log entries onto the set of unique visitors. This is mainly due to the proliferation of ISP proxy servers which assign rotating IP addresses to clients as they browse the Web. It is not uncommon to find many log entries corresponding to a limited number of proxy server IP addresses from large Internet Service Providers such as America Online. Therefore, two occurrences of the same IP address (separated by a sufficient amount of time), in fact, might correspond to two different users. Without user authentication or client-side cookies, it is still possible to accurately identify unique users through a combination of IP addresses and other information such as user agents and referrers [115].

Consider, for instance, the example of Fig. 12.4. On the left, the figure depicts a portion of a partly preprocessed log file (the time stamps are given as hours and minutes only). Using a combination of IP and Agent fields in the log file, we are able to partition the log into activity records for three separate users (depicted on the right).

Time	IP	URL	Ref	Agent
0:01	1.2.3.4	A	-	IE5;Win2k
0:09	1.2.3.4	B	A	IE5;Win2k
0:10	2.3.4.5	C	-	IE6;WinXP;SP1
0:12	2.3.4.5	B	C	IE6;WinXP;SP1
0:15	2.3.4.5	E	C	IE6;WinXP;SP1
0:19	1.2.3.4	C	A	IE5;Win2k
0:22	2.3.4.5	D	B	IE6;WinXP;SP1
0:22	1.2.3.4	A	-	IE6;WinXP;SP2
0:25	1.2.3.4	E	C	IE5;Win2k
0:25	1.2.3.4	C	A	IE6;WinXP;SP2
0:33	1.2.3.4	B	C	IE6;WinXP;SP2
0:58	1.2.3.4	D	B	IE6;WinXP;SP2
1:10	1.2.3.4	E	D	IE6;WinXP;SP2
1:15	1.2.3.4	A	-	IE5;Win2k
1:16	1.2.3.4	C	A	IE5;Win2k
1:17	1.2.3.4	F	C	IE6;WinXP;SP2
1:26	1.2.3.4	F	C	IE5;Win2k
1:30	1.2.3.4	B	A	IE5;Win2k
1:36	1.2.3.4	D	B	IE5;Win2k

User 1

0:01	1.2.3.4	A	-
0:09	1.2.3.4	B	A
0:19	1.2.3.4	C	A
0:25	1.2.3.4	E	C
1:15	1.2.3.4	A	-
1:26	1.2.3.4	F	C
1:30	1.2.3.4	B	A
1:36	1.2.3.4	D	B

User 2

0:10	2.3.4.5	C	-
0:12	2.3.4.5	B	C
0:15	2.3.4.5	E	C
0:22	2.3.4.5	D	B

User 3

0:22	1.2.3.4	A	-
0:25	1.2.3.4	C	A
0:33	1.2.3.4	B	C
0:58	1.2.3.4	D	B
1:10	1.2.3.4	E	D
1:17	1.2.3.4	F	C

Fig. 12.4. Example of user identification using IP + Agent

Sessionization

Sessionization is the process of segmenting the user activity record of each user into sessions, each representing a single visit to the site. Web sites without the benefit of additional authentication information from users and without mechanisms such as embedded session ids must rely on heuristics methods for sessionization. The goal of a sessionization heuristic is to reconstruct, from the clickstream data, the actual sequence of actions performed by one user during one visit to the site.

We denote the "conceptual" set of real sessions by R, representing the real activity of the user on the Web site. A sessionization heuristic h attempts to map R into a set of constructed sessions, denoted by C_h. For the ideal heuristic, h^*, we have $C_{h^*} = R$. In other words, the ideal heuristic can re-construct the exact sequence of user navigation during a session. Generally, sessionization heuristics fall into two basic categories: time-oriented or structure-oriented. Time-oriented heuristics apply either global or local time-out estimates to distinguish between consecutive sessions, while structure-oriented heuristics use either the static site structure or the implicit linkage structure captured in the referrer fields of the server logs. Various heuristics for sessionization have been identified and studied [115]. More recently, a formal framework for measuring the effectiveness of such heuristics has been proposed [498], and the impact of different heuristics on various Web usage mining tasks has been analyzed [46].

As an example, two variations of time-oriented heuristics and a basic navigation-oriented heuristic are given below. Each heuristic h scans the user activity logs to which the Web server log is partitioned after user identification, and outputs a set of constructed sessions:

- ***h*1:** Total session duration may not exceed a threshold θ. Given t_0, the timestamp for the first request in a constructed session S, the request with a timestamp t is assigned to S, iff $t - t_0 \leq \theta$.
- ***h*2:** Total time spent on a page may not exceed a threshold δ. Given t_1, the timestamp for request assigned to constructed session S, the next request with timestamp t_2 is assigned to S, iff $t_2 - t_1 \leq \delta$.
- ***h*-ref:** A request q is added to constructed session S if the referrer for q was previously invoked in S. Otherwise, q is used as the start of a new constructed session. Note that with this heuristic it is possible that a request q may potentially belong to more than one "open" constructed session, since q may have been accessed previously in multiple sessions. In this case, additional information can be used for disambiguation. For example, q could be added to the most recently opened session satisfying the above condition.

User 1

Time	IP	URL	Ref
0:01	1.2.3.4	A	-
0:09	1.2.3.4	B	A
0:19	1.2.3.4	C	A
0:25	1.2.3.4	E	C
1:15	1.2.3.4	A	-
1:26	1.2.3.4	F	C
1:30	1.2.3.4	B	A
1:36	1.2.3.4	D	B

Session 1

0:01	1.2.3.4	A	-
0:09	1.2.3.4	B	A
0:19	1.2.3.4	C	A
0:25	1.2.3.4	E	C

Session 2

1:15	1.2.3.4	A	-
1:26	1.2.3.4	F	C
1:30	1.2.3.4	B	A
1:36	1.2.3.4	D	B

Fig. 12.5. Example of sessionization with a time-oriented heuristic

User 1

Time	IP	URL	Ref
0:01	1.2.3.4	A	-
0:09	1.2.3.4	B	A
0:19	1.2.3.4	C	A
0:25	1.2.3.4	E	C
1:15	1.2.3.4	A	-
1:26	1.2.3.4	F	C
1:30	1.2.3.4	B	A
1:36	1.2.3.4	D	B

Session 1

0:01	1.2.3.4	A	-
0:09	1.2.3.4	B	A
0:19	1.2.3.4	C	A
0:25	1.2.3.4	E	C
1:26	1.2.3.4	F	C

Session 2

1:15	1.2.3.4	A	-
1:30	1.2.3.4	B	A
1:36	1.2.3.4	D	B

Fig. 12.6. Example of sessionization with the h-ref heuristic

An example of the application of sessionization heuristics is given in Fig. 12.5 and Fig. 12.6. In Fig. 12.5, the heuristic h_1, described above, with $\theta = 30$ minutes has been used to partition a user activity record (from the example of Fig. 12.4) into two separate sessions.

If we were to apply h_2 with a threshold of 10 minutes, the user record would be seen as three sessions, namely, A→B→C→E, A, and F→B→D. On the other hand, Fig. 12.6 depicts an example of using *h*-ref heuristic on the same user activity record. In this case, once the request for F (with time stamp 1:26) is reached, there are two open sessions, namely, A→B→C→E and A. But F is added to the first because its referrer, C, was invoked in session 1. The request for B (with time stamp 1:30) may potentially belong to both open sessions, since its referrer, A, is invoked both in session 1 and in session 2. In this case, it is added to the second session, since it is the most recently opened session.

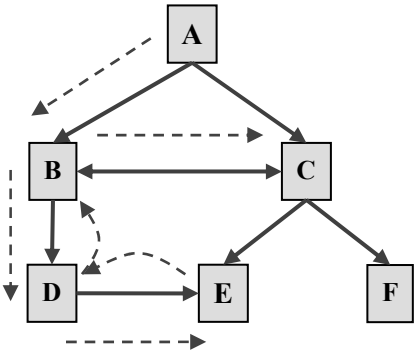
Episode identification can be performed as a final step in pre-processing of the clickstream data in order to focus on the relevant subsets of pageviews in each user session. An **episode** is a subset or subsequence of a session comprised of semantically or functionally related pageviews. This task may require the automatic or semi-automatic classification of page-

views into different functional types or into concept classes according to a domain ontology or concept hierarchy. In highly dynamic sites, it may also be necessary to map pageviews within each session into "service-based" classes according to a concept hierarchy over the space of possible parameters passed to script or database queries [47]. For example, the analysis may ignore the quantity and attributes of an item added to the shopping cart, and focus only on the action of adding the item to the cart.

Path Completion

Another potentially important pre-processing task which is usually performed after sessionization is **path completion**. Client- or proxy-side caching can often result in missing access references to those pages or objects that have been cached. For instance, if a user returns to a page A during the same session, the second access to A will likely result in viewing the previously downloaded version of A that was cached on the client-side, and therefore, no request is made to the server. This results in the second reference to A not being recorded on the server logs. **Missing references** due to caching can be heuristically inferred through path completion which relies on the knowledge of site structure and referrer information from server logs [115]. In the case of dynamically generated pages, form-based applications using the HTTP POST method result in all or part of the user input parameter not being appended to the URL accessed by the user (though, in the latter case, it is possible to recapture the user input through **packet sniffers** which listen to all incoming and outgoing TCP/IP network traffic on the server side).

A simple example of missing references is given in Fig. 12.7. On the left, a graph representing the linkage structure of the site is given. The dotted arrows represent the navigational path followed by a hypothetical user. After reaching page E, the user has backtracked (e.g., using the browser's "back" button) to page D and then B from which she has navigated to page C. The back references to D and B do not appear in the log file because these pages where cached on the client-side (thus no explicit server request was made for these pages). The log file shows that after a request for E, the next request by the user is for page C with a referrer B. In other words, there is a gap in the activity record corresponding to user's navigation from page E to page B. Given the site graph, it is possible to infer the two missing references (i.e., E → D and D → B) from the site structure and the referrer information given above. It should be noted that there are, in general, many (possibly infinite), candidate completions (for example, consider the sequence E → D, D → B, B → A, A → B). A simple heuristic



User's actual navigation path:

A → B → D → E → D → B → C

What the server log shows:

URL	Referrer
A	--
B	A
D	B
E	D
C	B

Fig. 12.7. Missing references due to caching.

that can be used for disambiguating among candidate paths is to select the one requiring the fewest number of "back" references.

Data Integration

The above pre-processing tasks ultimately result in a set of user sessions (or episodes), each corresponding to a delimited sequence of pageviews. However, in order to provide the most effective framework for pattern discovery, data from a variety of other sources must be integrated with the preprocessed clickstream data. This is particularly the case in e-commerce applications where the integration of both user data (e.g., demographics, ratings, and purchase histories) and product attributes and categories from operational databases is critical. Such data, used in conjunction with usage data, in the mining process can allow for the discovery of important business intelligence metrics such as **customer conversion ratios** and **lifetime values** [286].

In addition to user and product data, e-commerce data includes various product-oriented events such as shopping cart changes, order and shipping information, **impressions** (when the user visits a page containing an item of interest), **click-throughs** (when the user actually clicks on an item of interest in the current page), and other basic metrics primarily used for data analysis. The successful integration of these types of data requires the creation of a site-specific "event model" based on which subsets of a user's clickstream are aggregated and mapped to specific events such as the addition of a product to the shopping cart. Generally, the integrated e-commerce data is stored in the final transaction database. To enable full-featured Web analytics applications, this data is usually stored in a data warehouse called an **e-commerce data mart**. The e-commerce data mart

is a multi-dimensional database integrating data from various sources, and at different levels of aggregation. It can provide pre-computed e-metrics along multiple dimensions, and is used as the primary data source for OLAP (Online Analytical Processing), for data visualization, and in data selection for a variety of data mining tasks [71, 279]. Some examples of such metrics include frequency or monetary value of purchases, average size of market baskets, the number of different items purchased, the number of different item categories purchased, the amount of time spent on pages or sections of the site, day of week and time of day when a certain activity occurred, response to recommendations and online specials, etc.

12.2 Data Modeling for Web Usage Mining

Usage data pre-processing results in a set of n pageviews, $P = \{p_1, p_2, \cdots, p_n\}$, and a set of m **user transactions**, $T = \{t_1, t_2, \cdots, t_m\}$, where each t_i in T is a subset of P. **Pageviews** are semantically meaningful entities to which mining tasks are applied (such as pages or products). Conceptually, we view each transaction t as an l-length sequence of ordered pairs:

$$t = \left\langle (p_1^t, w(p_1^t)), (p_2^t, w(p_2^t)), \cdots, (p_l^t, w(p_l^t)) \right\rangle,$$

where each $p_i^t = p_j$ for some j in $\{1, 2, \cdots, n\}$, and $w(p_i^t)$ is the weight associated with pageview p_i^t in transaction t, representing its significance. The weights can be determined in a number of ways, in part based on the type of analysis or the intended personalization framework. For example, in **collaborative filtering** applications which rely on the profiles of similar users to make recommendations to the current user, weights may be based on user ratings of items. In most Web usage mining tasks the weights are either binary, representing the existence or non-existence of a pageview in the transaction; or they can be a function of the duration of the pageview in the user's session. In the case of time durations, it should be noted that usually the time spent by a user on the last pageview in the session is not available. One commonly used option is to set the weight for the last pageview to be the mean time duration for the page taken across all sessions in which the pageview does not occur as the last one. In practice, it is common to use a normalized value of page duration instead of raw time duration in order to account for user variances. In some applications, the log of pageview duration is used as the weight to reduce the noise in the data.

For many data mining tasks, such as clustering and association rule mining, where the ordering of pageviews in a transaction is not relevant, we can represent each user transaction as a vector over the n-dimensional space

Sessions / users	Pageviews A	B	C	D	E	F
user0	15	5	0	0	0	185
user1	0	0	32	4	0	0
user2	12	0	0	56	236	0
user3	9	47	0	0	0	134
user4	0	0	23	15	0	0
user5	17	0	0	157	69	0
user6	24	89	0	0	0	354
user7	0	0	78	27	0	0
user8	7	0	45	20	127	0
user9	0	38	57	0	0	15

Fig. 12.8. An example of a user-pageview matrix (or transaction matrix)

of pageviews. Given the transaction t above, the transaction vector **t** (we use a bold face lower case letter to represent a vector) is given by:

$$\mathbf{t} = \left(w^t_{p_1}, w^t_{p_2}, \cdots, w^t_{p_n}\right),$$

where each $w^t_{p_i} = w(p^t_j)$, for some j in $\{1, 2, \cdots, n\}$, if p_j appears in the transaction t, and $w^t_{p_i} = 0$ otherwise. Thus, conceptually, the set of all user transactions can be viewed as an $m \times n$ **user-pageview matrix** (also called the **transaction matrix**), denoted by *UPM*.

An example of a hypothetical user-pageview matrix is depicted in Fig. 12.8. In this example, the weights for each pageview is the amount of time (e.g., in seconds) that a particular user spent on the pageview. In practice, these weights must be normalized to account for variances in viewing times by different users. It should also be noted that the weights may be composite or aggregate values in cases where the pageview represents a collection or sequence of pages and not a single page.

Given a set of transactions in the user-pageview matrix as described above, a variety of unsupervised learning techniques can be applied to obtain patterns. These techniques such as clustering of transactions (or sessions) can lead to the discovery of important user or visitor segments. Other techniques such as item (e.g., pageview) clustering and association or sequential pattern mining can find important relationships among items based on the navigational patterns of users in the site.

As noted earlier, it is also possible to integrate other sources of knowledge, such as semantic information from the content of Web pages with the Web usage mining process. Generally, the textual features from the content of Web pages represent the underlying semantics of the site. Each

pageview p can be represented as a r-dimensional feature vector, where r is the total number of extracted features (words or concepts) from the site in a global dictionary. This vector, denoted by $\mathbf{p}$, can be given by:

$$\mathbf{p} = \left(fw^p(f_1), fw^p(f_2), ..., fw^p(f_r)\right)$$

where $fw^p(f_j)$ is the weight of the jth feature (i.e., f_j) in pageview p, for $1 \leq j \leq r$. For the whole collection of pageviews in the site, we then have an $n \times r$ **pageview-feature matrix** $PFM = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$. The integration process may, for example, involve the transformation of user transactions (in user-pageview matrix) into "content-enhanced" transactions containing the semantic features of the pageviews. The goal of such a transformation is to represent each user session (or more generally, each user profile) as a vector of semantic features (i.e., textual features or concept labels) rather than as a vector over pageviews. In this way, a user's session reflects not only the pages visited, but also the significance of various concepts or context features that are relevant to the user's interaction.

While, in practice, there are several ways to accomplish this transformation, the most direct approach involves mapping each pageview in a transaction to one or more content features. The range of this mapping can be the full feature space, or feature sets (composite features) which in turn may represent concepts and concept categories. Conceptually, the transformation can be viewed as the multiplication of the user-pageview matrix *UPM*, defined earlier, with the pageview-feature matrix *PFM*. The result is a new matrix, $TFM = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_m\}$, where each $\mathbf{t}_i$ is a r-dimensional vector over the feature space. Thus, a user transaction can be represented as a content feature vector, reflecting that user's interests in particular concepts or topics.

As an example of content-enhanced transactions, consider Fig. 12.9 which shows a hypothetical matrix of user sessions (user-pageview matrix) as well as a document index for the corresponding Web site conceptually represented as a **term-pageview matrix**. Note that the transpose of this term-pageview matrix is the pageview-feature matrix. The user-pageview matrix simply reflects the pages visited by users in various sessions. On the other hand, the term-pageview matrix represents the concepts that appear in each page. For simplicity we have assumed that all the weights are binary (however, note that in practice weights in the user transaction data are usually not binary and represent some measure of significance of the page in that transaction; and the weights in the term-pageview matrix are usually a function of term frequencies).

In this case, the corresponding **content-enhanced transaction matrix** (derived by multiplying the user-pageview matrix and the transpose of the term-pageview matrix) is depicted in Fig. 12.10. The resulting matrix

	A.html	B.html	C.html	D.html	E.html
user1	1	0	1	0	1
user2	1	1	0	0	1
user3	0	1	1	1	0
user4	1	0	1	1	1
user5	1	1	0	0	1
user6	1	0	1	1	1

	A.html	B.html	C.html	D.html	E.html
web	0	0	1	1	1
data	0	1	1	1	0
mining	0	1	1	1	0
business	1	1	0	0	0
intelligence	1	1	0	0	1
marketing	1	1	0	0	1
ecommerce	0	1	1	0	0
search	1	0	1	0	0
information	1	0	1	1	1
retrieval	1	0	1	1	1

Fig. 12.9. Examples of a user-pageview matrix (top) and a term-pageview matrix (bottom)

	web	data	mining	business	intelligence	marketing	ecommerce	search	information	retrieval
user1	2	1	1	1	2	2	1	2	3	3
user2	1	1	1	2	3	3	1	1	2	2
user3	2	3	3	1	1	1	2	1	2	2
user4	3	2	2	1	2	2	1	2	4	4
user5	1	1	1	2	3	3	1	1	2	2
user6	3	2	2	1	2	2	1	2	4	4

Fig. 12.10. The content-enhanced transaction matrix from matrices of Fig. 12.9

shows, for example, that users 4 and 6 are more interested in Web information retrieval, while user 3 is more interested in data mining.

Various data mining tasks can now be performed on the content-enhanced transaction data. For example, clustering the enhanced transaction matrix of Fig. 12.10 may reveal segments of users that have common interests in different concepts as indicated from their navigational behaviors.

If the content features include relational attributes associated with items on the Web site, then the discovered patterns may reveal user interests at the deeper semantic level reflected in the underlying properties of the items that are accessed by the users on the Web site. As an example, consider a site containing information about movies. The site may contain pages related to the movies themselves, as well as attributes describing the properties of each movie, such as actors, directors, and genres. The mining

process may, for instance, generate an association rule such as: {"British", "Romance", "Comedy" ⇒ "Hugh Grant"}, suggesting that users who are interested in British romantic comedies may also like the actor Hugh Grant (with a certain degree of confidence). Therefore, the integration of semantic content with Web usage mining can potentially provide a better understanding of the underlying relationships among objects.

12.3 Discovery and Analysis of Web Usage Patterns

The types and levels of analysis, performed on the integrated usage data, depend on the ultimate goals of the analyst and the desired outcomes. In this section we describe some of the most common types of pattern discovery and analysis techniques employed in the Web usage mining domain and discuss some of their applications.

12.3.1 Session and Visitor Analysis

The statistical analysis of pre-processed session data constitutes the most common form of analysis. In this case, data is aggregated by predetermined units such as days, sessions, visitors, or domains. Standard statistical techniques can be used on this data to gain knowledge about visitor behavior. This is the approach taken by most commercial tools available for Web log analysis. Reports based on this type of analysis may include information about most frequently accessed pages, average view time of a page, average length of a path through a site, common entry and exit points, and other aggregate measures. Despite a lack of depth in this type of analysis, the resulting knowledge can be potentially useful for improving the system performance, and providing support for marketing decisions. Furthermore, commercial Web analytics tools are increasingly incorporating a variety of data mining algorithms resulting in more sophisticated site and customer metrics.

Another form of analysis on integrated usage data is Online Analytical Processing (OLAP). OLAP provides a more integrated framework for analysis with a higher degree of flexibility. The data source for OLAP analysis is usually a multidimensional data warehouse which integrates usage, content, and e-commerce data at different levels of aggregation for each dimension. OLAP tools allow changes in aggregation levels along each dimension during the analysis. Analysis dimensions in such a structure can be based on various fields available in the log files, and may include time duration, domain, requested resource, user agent, and referrers.

This allows the analysis to be performed on portions of the log related to a specific time interval, or at a higher level of abstraction with respect to the URL path structure. The integration of e-commerce data in the data warehouse can further enhance the ability of OLAP tools to derive important business intelligence metrics [71]. The output from OLAP queries can also be used as the input for a variety of data mining or data visualization tools.

12.3.2 Cluster Analysis and Visitor Segmentation

Clustering is a data mining technique that groups together a set of items having similar characteristics. In the usage domain, there are two kinds of interesting clusters that can be discovered: user clusters and page clusters.

Clustering of user records (sessions or transactions) is one of the most commonly used analysis tasks in Web usage mining and Web analytics. Clustering of users tends to establish groups of users exhibiting similar browsing patterns. Such knowledge is especially useful for inferring user demographics in order to perform market segmentation in e-commerce applications or provide **personalized Web content** to the users with similar interests. Further analysis of user groups based on their demographic attributes (e.g., age, gender, income level, etc.) may lead to the discovery of valuable business intelligence. **Usage-based clustering** has also been used to create Web-based "user communities" reflecting similar interests of groups of users [423], and to learn user models that can be used to provide dynamic recommendations in Web personalization applications [390].

Given the mapping of user transactions into a multi-dimensional space as vectors of pageviews (see Fig. 12.8), standard clustering algorithms, such as *k*-means, can partition this space into groups of transactions that are close to each other based on a measure of distance or similarity among the vectors (see Chap. 4). Transaction clusters obtained in this way can represent user or visitor segments based on their navigational behavior or other attributes that have been captured in the transaction file. However, transaction clusters by themselves are not an effective means of capturing the aggregated view of common user patterns. Each transaction cluster may potentially contain thousands of user transactions involving hundreds of pageview references. The ultimate goal in clustering user transactions is to provide the ability to analyze each segment for deriving business intelligence, or to use them for tasks such as personalization.

One straightforward approach in creating an aggregate view of each cluster is to compute the **centroid** (or the mean vector) of each cluster. The dimension value for each pageview in the mean vector is computed by finding the ratio of the sum of the pageview weights across transactions to

the total number of transactions in the cluster. If pageview weights in the original transactions are binary, then the dimension value of a pageview p in a cluster centroid represents the percentage of transactions in the cluster in which p occurs. Thus, the centroid dimension value of p provides a measure of its significance in the cluster. Pageviews in the centroid can be sorted according to these weights and lower weight pageviews can be filtered out. The resulting set of pageview-weight pairs can be viewed as an "aggregate usage profile" representing the interests or behavior of a significant group of users.

More formally, given a transaction cluster *cl,* we can construct the aggregate profile pr_{cl} as a set of **pageview-weight** pairs by computing the centroid of *cl:*

$$pr_{cl} = \{(p, weight(p, pr_{cl})) \mid weight(p, pr_{cl}) \geq \mu\}, \tag{1}$$

where:

- the significance weight, $weight(p, pr_{cl})$, of the page p within the aggregate profile pr_{cl} is given by

$$weight(p, pr_{cl}) = \frac{1}{|cl|} \sum_{\mathbf{s} \in cl} w(p, \mathbf{s}); \tag{2}$$

- $|cl|$ is the number of transactions in cluster *cl*;
- $w(p, \mathbf{s})$ is the weight of page p in transaction vector $\mathbf{s}$ of cluster *cl*; and
- the threshold μ is used to focus only on those pages in the cluster that appear in a sufficient number of vectors in that cluster.

Each such profile, in turn, can be represented as a vector in the original n-dimensional space of pageviews. This aggregate representation can be used directly for predictive modeling and in applications such as recommender systems: given a new user, u, who has accessed a set of pages, P_u, so far, we can measure the similarity of P_u to the discovered profiles, and recommend to the user those pages in matching profiles which have not yet been accessed by the user.

As an example, consider the transaction data depicted in Fig. 12.11 (left). For simplicity we assume that feature (pageview) weights in each transaction vector are binary (in contrast to weights based on a function of pageview duration). We assume that the data has already been clustered using a standard clustering algorithm such as *k*-means, resulting in three clusters of user transactions. The table on the right of Fig. 12.11 shows the aggregate profile corresponding to cluster 1. As indicated by the pageview weights, pageviews B and F are the most significant pages characterizing the common interests of users in this segment. Pageview C, however, only appears in one transaction and might be removed given a filtering thresh-

		A	B	C	D	E	F
Cluster 0	user 1	0	0	1	1	0	0
	user 4	0	0	1	1	0	0
	user 7	0	0	1	1	0	0
Cluster 1	user 0	1	1	0	0	0	1
	user 3	1	1	0	0	0	1
	user 6	1	1	0	0	0	1
	user 9	0	1	1	0	0	1
Cluster 2	user 2	1	0	0	1	1	0
	user 5	1	0	0	1	1	0
	user 8	1	0	1	1	1	0

Aggregated Profile for Cluster 1	
Weight	Pageview
1.00	B
1.00	F
0.75	A
0.25	C

Fig. 12.11. Derivation of aggregate profiles from Web transaction clusters

old greater than 0.25. Such patterns are useful for characterizing user or customer segments. This example, for instance, indicates that the resulting user segment is clearly interested in items B and F and to a lesser degree in item A. Given a new user who shows interest in items A and B, this pattern may be used to infer that the user might belong to this segment and, therefore, we might recommend item F to that user.

Clustering of pages (or items) can be performed based on the usage data (i.e., starting from the user sessions or transaction data), or based on the content features associated with pages or items (keywords or product attributes). In the case of content-based clustering, the result may be collections of pages or products related to the same topic or category. In usage-based clustering, items that are commonly accessed or purchased together can be automatically organized into groups. It can also be used to provide permanent or dynamic HTML pages that suggest related hyperlinks to the users according to their past history of navigational or purchase activities.

A variety of stochastic methods have also been proposed recently for clustering of user transactions, and more generally for user modeling. For example, recent work in this area has shown that **mixture models** are able to capture more complex, dynamic user behavior. This is, in part, because the observation data (i.e., the user-item space) in some applications (such as large and very dynamic Web sites) may be too complex to be modeled by basic probability distributions such as a normal or a multinomial distribution. In particular, each user may exhibit different "types" of behavior corresponding to different tasks, and common behaviors may each be reflected in a different distribution within the data.

The general idea behind mixture models (such as a **mixture of Markov models**) is as follow. We assume that there exist k types of user behavior (or k user clusters) within the data, and each user session is assumed to be generated via a generative process which models the probability distributions of the observed variables and hidden variables. First, a user cluster is chosen with some probability. Then, the user session is generated from a Markov model with parameters specific to that user cluster. The probabilities of each user cluster is estimated, usually via the **EM** [127] algorithm, as well as the parameters of each mixture component. Mixture-based user models can provide a great deal of flexibility. For example, a mixture of first-order Markov models [76] not only can probabilistically cluster user sessions based on similarities in navigation behavior, but also characterize each type of user behavior using a first-order Markov model, thus capturing popular navigation paths or characteristics of each user cluster. A mixture of hidden Markov models was proposed in [580] for modeling clickstream of Web surfers. In addition to user-based clustering, this approach can also be used for automatic page classification. Incidentally, mixture models have been discussed in Sect. 3.7 in the context of naïve Bayesian classification. The EM algorithm is used in the same context in Sect. 5.1.

Mixture models tend to have their own shortcomings. From the data generation perspective, each individual observation (such as a user session) is generated from one and only one component model. The probability assignment to each component only measures the uncertainty about this assignment. This assumption limits this model's ability of capturing complex user behavior, and more seriously, may result in overfitting.

Probabilistic Latent Semantic Analysis (PLSA) provides a reasonable solution to the above problem [240]. In the context of Web user navigation, each observation (a user visiting a page) is assumed to be generated based on a set of unobserved (hidden) variables which "explain" the user-page observations. The data generation process is as follows: a user is selected with a certain probability, next conditioned on the user, a hidden variable is selected, and then the page to visit is selected conditioned on the chosen hidden variable. Since each user usually visits multiple pages, this data generation process ensures that each user is explicitly associated with multiple hidden variables, thus reducing the overfitting problems associated with the above mixture models. The PLSA model also uses the EM algorithm to estimate the parameters which probabilistically characterize the hidden variables underlying the co-occurrence observation data, and measure the relationship among hidden and observed variables.

This approach provides a great deal of flexibility since it provides a single framework for quantifying the relationships between users, between items, between users and items, and between users or items and hidden

variables that "explain" the observed relationships [254]. Given a set of n user profiles (or transaction vectors), $UP = \{u_1, u_2, \ldots, u_n\}$, and a set of m items (e.g., pages or products), $I = \{i_1, i_2, \ldots, i_m\}$, the PLSA model associates a set of unobserved factor variables $Z = \{z_1, z_2, \ldots, z_q\}$ with observations in the data (q is specified by the user). Each observation corresponds to a weight $w_{u_k}(i_j)$ for an item i_j in the user profile for a user u_k. This weight may, for example, correspond to the significance of the page in the user transaction or the user rating associated with the item. For a given user u and a given item i, the following joint probability can be derived (see [254] for details of the derivation):

$$\Pr(u,i) = \sum_{k=1}^{q} \Pr(z_k) \Pr(u \mid z_k) \Pr(i \mid z_k). \tag{3}$$

In order to explain the observations in (UP, I), we need to estimate the parameters $\Pr(z_k)$, $\Pr(u|z_k)$, and $\Pr(i|z_k)$, while maximizing the following likelihood $L(UP, I)$ of the observation data:

$$L(UP,I) = \sum_{u \in UP} \sum_{i \in I} w_u(i) \log \Pr(u,i). \tag{4}$$

The Expectation–Maximization (EM) algorithm is used to perform maximum likelihood parameter estimation. Based on initial values of $\Pr(z_k)$, $\Pr(u|z_k)$, and $\Pr(i|z_k)$, the algorithm alternates between an expectation step and maximization step. In the expectation step, posterior probabilities are computed for latent variables based on current estimates, and in the maximization step the re-estimated parameters are obtained. Iterating the expectation and maximization steps monotonically increases the total likelihood of the observed data $L(UP, I)$, until a local optimal solution is reached. Details of this approach can be found in [254].

Again, one of the main advantages of PLSA model in Web usage mining is that using probabilistic inference with the above estimated parameters, we can derive relationships among users, among pages, and between users and pages. Thus this framework provides a flexible approach to model a variety of types of usage patterns.

12.3.3 Association and Correlation Analysis

Association rule discovery and statistical correlation analysis can find groups of items or pages that are commonly accessed or purchased together. This, in turn, enables Web sites to organize the site content more efficiently, or to provide effective cross-sale product recommendations.

Most common approaches to association discovery are based on the Apriori algorithm (see Sect. 2.2). This algorithm finds groups of items (pageviews appearing in the preprocessed log) occurring frequently together in many transactions (i.e., satisfying a user specified minimum support threshold). Such groups of items are referred to as **frequent itemsets**. Association rules which satisfy a minimum confidence threshold are then generated from the frequent itemsets.

Recall an association rule is an expression of the form $X \rightarrow Y$ [*sup, conf*], where X and Y are itemsets, *sup* is the support of the itemset $X \cup Y$ representing the probability that X and Y occur together in a transaction, and *conf* is the confidence of the rule, defined by $sup(X \cup Y) / sup(X)$, representing the conditional probability that Y occurs in a transaction given that X has occurred in that transaction. More details on association rule discovery can be found in Chap. 2.

The mining of association rules in Web transaction data has many advantages. For example, a high-confidence rule such as

$$\text{special-offers/, /products/software/} \rightarrow \text{shopping-cart/}$$

might provide some indication that a promotional campaign on software products is positively affecting online sales. Such rules can also be used to optimize the structure of the site. For example, if a site does not provide direct linkage between two pages A and B, the discovery of a rule, $\text{A} \rightarrow \text{B}$, would indicates that providing a direct hyperlink from A to B might aid users in finding the intended information. Both association analysis (among products or pageviews) and statistical correlation analysis (generally among customers or visitors) have been used successfully in Web personalization and recommender systems [236, 389].

Indeed, one of the primary applications of association rule mining in Web usage or e-commerce data is in recommendation. For example, in the collaborative filtering context, Sarwar et al. [474] used association rules in the context of a *top-N* recommender system for e-commerce. The preferences of the target user are matched against the items in the antecedent X of each rule, and the items on the right hand side of the matching rules are sorted according to the confidence values. Then the top N ranked items from this list are recommended to the target user (see Sect. 3.5.3).

One problem for association rule recommendation systems is that a system cannot give any recommendations when the dataset is sparse (which is often the case in Web usage mining and collaborative filtering applications). The reason for this sparsity is that any given user visits (or rates) only a very small fraction of the available items, and thus it is often difficult to find a sufficient number of common items in multiple user profiles. Sarwar et al. [474] relied on some standard dimensionality reduction tech-

niques to alleviate this problem. One deficiency of this and other dimensionality reduction approaches is that some of the useful or interesting items may be removed, and therefore, may not appear in the final patterns. Fu et al. [187] proposed two potential solutions to this problem. The first solution is to rank all the discovered rules based on the degree of intersection between the left-hand side of each rule and the user's active session and then to generate the top k recommendations. This approach will relax the constraint of having to obtain a complete match with the left-hand-side of the rules. The second solution is to utilize **collaborative filtering**: the system finds "close neighbors" who have similar interest to a target user and makes recommendations based on the close neighbors' histories.

Lin et al. [337] proposed a **collaborative recommendation** system using association rules. The proposed mining algorithm finds an appropriate number of rules for each target user by automatically selecting the minimum support. The system generates association rules among users (user associations), as well as among items (item associations). If a user minimum support is greater than a threshold, the system generates recommendations based on user associations, else it uses item associations.

Because it is difficult to find matching rule antecedent with a full user profile (e.g., a full user session or transaction), association-based recommendation algorithms typically use a sliding window w over the target user's active profile or session. The window represents the portion of user's history that will be used to predict future user actions (based on matches with the left-hand sides of the discovered rules). The size of this window is iteratively decreased until an exact match with the antecedent of a rule is found. A problem with the naive approach to this algorithm is that it requires repeated search through the rule-base. However, efficient trie-based data structure can be used to store the discovered itemsets and allow for efficient generation of recommendations without the need to generate all association rules from frequent itemsets [389]. Such data structures are commonly used for string or sequence searching applications. In the context of association rule mining, the frequent itemsets are stored in a directed acyclic graph. This **frequent itemset graph** is an extension of the lexicographic tree used in the tree projection mining algorithm of Agarwal, et al. [2]. The graph is organized into levels from 0 to k, where k is the maximum size among all frequent itemsets. Each node at depth d in the graph corresponds to an itemset, X, of size d and is linked to itemsets of size d+1 that contain X at level d+1. The single root node at level 0 corresponds to the empty itemset. To be able to search for different orderings of an itemset, all itemsets are sorted in lexicographic order before being inserted into the graph. If the graph is used to recommend items to a new

target user, that user's active session is also sorted in the same manner before matching with itemsets.

As an example, suppose that in a hypothetical Web site with user transaction data depicted in the left table of Fig. 12.12. Using a minimum support (minsup) threshold of 4 (i.e., 80%), the Apriori algorithm discovers the frequent itemsets given in the right table. For each itemset, the support is also given. The corresponding frequent itemset graph is depicted in Fig. 12.13.

A recommendation engine based on this framework matches the current user session window with the previously discovered frequent itemsets to find candidate items (pages) for recommendation. Given an active session window w and a group of frequent itemsets, the algorithm considers all the frequent itemsets of size $|w| + 1$ containing the current session window by performing a depth-first search of the Frequent Itemset Graph to level $|w|$. The recommendation value of each candidate is based on the confidence of the corresponding association rule whose consequent is the singleton containing the page to be recommended. If a match is found, then the children of the matching node n containing w are used to generate candidate recommendations. In practice, the window w can be incrementally decreased until a match is found with and itemset. For example, given user active session window $<B, E>$, the recommendation generation algorithm, using the graph of Fig. 12.13, finds items A and C as candidate recommendations. The recommendation scores of item A and C are 1 and 4/5, corresponding to the confidences of the rules, $B, E \rightarrow A$ and $B, E \rightarrow C$, respectively.

A problem with using a single global minimum support threshold in association rule mining is that the discovered patterns will not include "rare" but important items which may not occur frequently in the transaction data. This is particularly important when dealing with Web usage data, it is often the case that references to deeper content or product-oriented pages oc-

Transactions
A, B, D, E
A, B, E, C, D
A, B, E, C
B, E, B, A, C
D, A, B, E, C

Size 1		Size 2		Size 3		Size 4	
Itemset	Supp.	Itemset	Supp.	Itemset	Supp.	Itemset	Supp.
A	5	A,B	5	A,B,C	4	A,B,C,E	4
B	5	A,C	4	A,B,E	5		
C	4	A,E	5	A,C,E	4		
E	5	B,C	4	B,C,E	4		
		B,E	5				
		C,E	4				

Fig. 12.12. Web transactions and resulting frequent itemsets (minsup = 4)

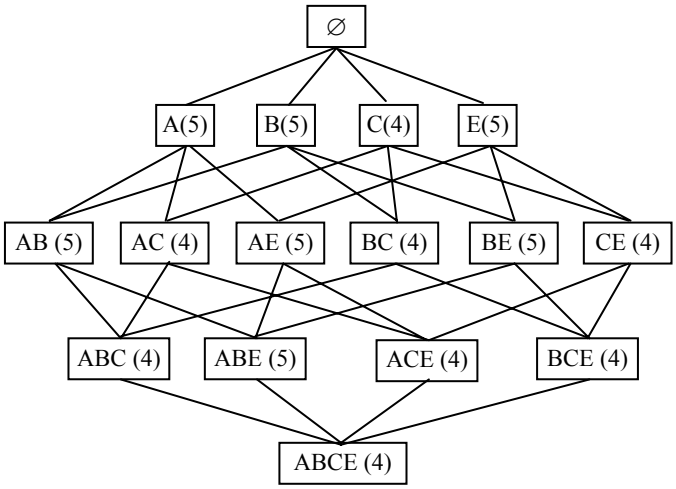


Fig. 12.13. A frequent itemset graph.

cur far less frequently than those of top level navigation-oriented pages. Yet, for effective Web personalization, it is important to capture patterns and generate recommendations that contain these items. A mining method based on **multiple minimum supports** is proposed in [344] that allows users to specify different support values for different items. In this method, the support of an itemset is defined as the minimum support of all items contained in the itemset. For more details on mining using multiple minimum supports, see Sect. 2.4. The specification of multiple minimum supports thus allows frequent itemsets to potentially contain rare items which are deemed important. It has been shown that the use of multiple support association rules in the context of Web personalization can dramatically increase the coverage (or recall) of recommendations while maintaining a reasonable precision [389].

12.3.4 Analysis of Sequential and Navigational Patterns

The technique of sequential pattern mining attempts to find inter-session patterns such that the presence of a set of items is followed by another item in a time-ordered set of sessions or episodes. By using this approach, Web marketers can predict future visit patterns which will be helpful in placing advertisements aimed at certain user groups. Other types of temporal analysis that can be performed on sequential patterns include trend analysis, change point detection, or similarity analysis. In the context of Web

usage data, **sequential pattern mining** can be used to capture frequent navigational paths among user trails.

Sequential patterns (SPs) in Web usage data capture the Web page trails that are often visited by users, in the order that they were visited. Sequential patterns are those sequences of items that frequently occur in a sufficiently large proportion of (sequence) transactions. A sequence $\langle s_1 s_2 \ldots s_n \rangle$ occurs in a transaction $t = \langle p_1, p_2, \ldots, p_m \rangle$ (where $n \leq m$) if there exist n positive integers $1 \leq a_1 < a_2 < \ldots < a_n \leq m$, and $s_i = p_{a_i}$ for all i. We say that $\langle cs_1\ cs_2 \ldots cs_n \rangle$ is a **contiguous sequence** in t if there exists an integer $0 \leq b \leq m - n$, and $cs_i = p_{b+i}$ for all $i = 1$ to n. In a **contiguous sequential pattern** (CSP), each pair of adjacent items, s_i and s_{i+1}, must appear consecutively in a transaction t which supports the pattern. A normal sequential pattern can represent non-contiguous frequent sequences in the underlying set of sequence transactions.

Given a sequence transaction set T, the support (denoted by $sup(S)$) of a sequential (respectively, contiguous sequential) pattern S in T is the fraction of transactions in T that contain S. The confidence of the rule $X \rightarrow Y$, where X and Y are (contiguous) sequential patterns, is defined as:

$$conf(X \rightarrow Y) = sup(X \circ Y) / sup(X),$$

where $\circ$ denotes the concatenation operator.

In the context of Web usage data, CSPs can be used to capture frequent navigational paths among user trails [497]. In contrast, items appearing in SPs, while preserving the underlying ordering, need not be adjacent, and thus they represent more general navigational patterns within the site. Note that sequences and sequential patterns or rules discussed here are special cases of those defined in Sect. 2.9.

The view of Web transactions as sequences of pageviews allows for a number of useful and well-studied models to be used in discovering or analyzing user navigation patterns. One such approach is to model the navigational activities in the Web site as a Markov model: each pageview (or a category) can be represented as a state and the transition probability between two states can represent the likelihood that a user will navigate from one state to the other. This representation allows for the computation of a number of useful user or site metrics. For example, one might compute the probability that a user will make a purchase, given that she has performed a search in an online catalog. **Markov models** have been proposed as the underlying modeling machinery for link prediction as well as for Web prefetching to minimize system latencies [132, 473]. The goal of such approaches is to predict the *next* user action based on a user's previous surfing behavior. They have also been used to discover high probability user navigational trails in a Web site [57]. More sophisticated statistical learn-

ing techniques, such as mixtures of Markov models, have also been used to cluster navigational sequences and perform exploratory analysis of users' navigational behavior in a site [76].

More formally, a Markov model is characterized by a set of states $\{s_1, s_2, \ldots, s_n\}$ and a transition probability matrix, $[Pr_{i,j}]_{n \times n}$, where $Pr_{i,j}$ represents the probability of a transition from state s_i to state s_j. Markov models are especially suited for predictive modeling based on contiguous sequences of events. Each state represents a contiguous subsequence of prior events. The order of the Markov model corresponds to the number of prior events used in predicting a future event. So, a kth-order Markov model predicts the probability of next event by looking the past k events. Given a set of all paths R, the probability of reaching a state s_j from a state s_i via a (non-cyclic) path $r \in R$ is the product of all the transition probabilities along the path and is given by $Pr(r) = \prod Pr_{m,m+1}$, where m ranges from i to $j - 1$. The probability of reaching s_j from s_i is the sum of these path probabilities over all paths: $Pr(j|i) = \sum_{r \in R} Pr(r)$.

As an example of how Web transactions can be modeled as a Markov model, consider the set of Web transaction given in Fig. 12.14 (left). The Web transactions involve pageviews A, B, C, D, and E. For each transaction the frequency of occurrences of that transaction in the data is given in the table's second column (thus there are a total of 50 transactions in the data set). The (absorbing) Markov model for this data is also given in Fig. 12.14 (right). The transitions from the "start" state represent the prior probabilities for transactions starting with pageviews A and B. The transitions into the "final" state represent the probabilities that the paths end with the specified originating pageviews. For example, the transition probability from the state A to B is 16/28 = 0.57 since out of the 28 occurrences of A in transactions, in 16 cases, B occurs immediately after A.

Higher-order Markov models generally provide a higher prediction accuracy. However, this is usually at the cost of lower coverage (or recall) and much higher model complexity due to the larger number of states. In order to remedy the coverage and space complexity problems, Pitkow and Pirolli [446] proposed all-kth-order Markov models (for coverage improvement) and a new state reduction technique, called **longest repeating subsequences** (LRS) (for reducing model size). The use of all-kth-order Markov models generally requires the generation of separate models for each of the k orders: if the model cannot make a prediction using the kth order, it will attempt to make a prediction by incrementally decreasing the model order. This scheme can easily lead to even higher space complexity since it requires the representation of all possible states for each k. Deshpande and Karypis [132] proposed selective Markov models, introducing several schemes in order to tackle the model complexity problems

Transaction	Frequency
A, B, E	10
B, D, B, C	4
B, C, E	10
A, B, E, F	6
A, D, B	12
B, D, B, E	8

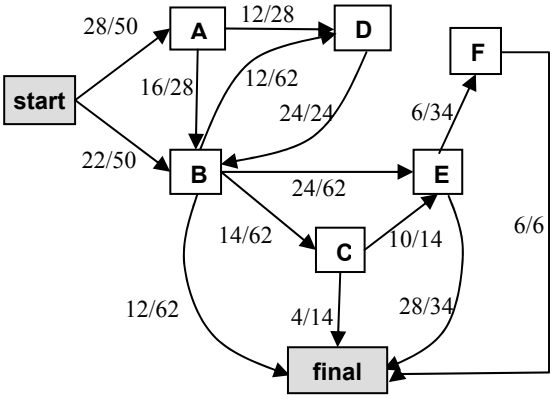


Fig. 12.14. An example of modeling navigational trails as a Markov

with all-*k*th-order Markov models. The proposed schemes involve pruning the model based on criteria such as support, confidence, and error rate. In particular, the support-pruned Markov models eliminate all states with low support determined by a minimum frequency threshold.

Another way of efficiently representing contiguous navigational trails is by inserting each trail into a *trie* structure. A good example of this approach is the notion of aggregate tree introduced as part of the WUM (Web Utilization Miner) system [497]. The aggregation service of WUM extracts the transactions from a collection of Web logs, transforms them into sequences, and merges those sequences with the same prefix into the aggregate tree (a trie structure). Each node in the tree represents a navigational subsequence from the root (an empty node) to a page and is annotated by the frequency of occurrences of that subsequence in the transaction data (and possibly other information such as markers to distinguish among repeat occurrences of the corresponding page in the subsequence). WUM uses a mining query language, called MINT, to discover generalized navigational patterns from this trie structure. MINT includes mechanisms to specify sophisticated constraints on pattern templates, such as wildcards with user-specified boundaries, as well as other statistical thresholds such as support and confidence. This approach and its extensions have proved useful in evaluating the navigational design of a Web site [496].

As an example, again consider the set of Web transactions given in the previous example. Figure 12.15 shows a simplified version of WUM's aggregate tree structure derived from these transactions. Each node in the tree represents a navigational subsequence from the root (an empty node) to a page and is annotated by the frequency of occurrences of that subsequence in the session data. The advantage of this approach is that the search for

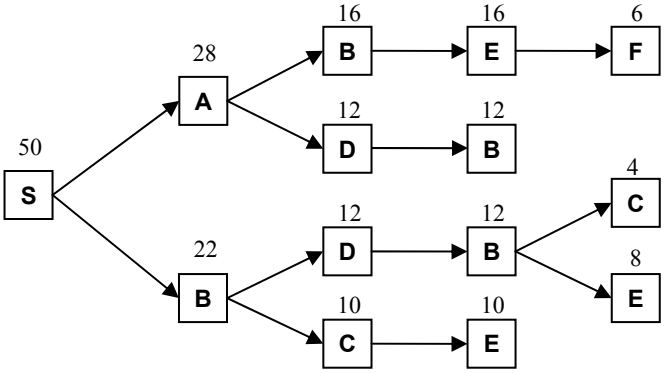


Fig. 12.15. An example of modeling navigational trails in an aggregate tree

navigational patterns can be performed very efficiently and the confidence and support for the navigational patterns can be readily obtained from the node annotations in the tree. For example, consider the contiguous navigational sequence <A, B, E, F>. The support for this sequence can be computed as the support of the last page in the sequence, *F*, divided by the support of the root node: 6/50 = 0.12, and the confidence of the sequence is the support of *F* divided by the support of its predecessor, E, or 6/16 = 0.375. If there are multiple branches in the tree containing the same navigational sequence, then the support for the sequence is the sum of the supports for all occurrences of the sequence in the tree and the confidence is updated accordingly. For example, the support of the sequence <D, B> is (12+12)/50 = 0.48, while the confidence is the aggregate support for B divided by the aggregate support for D, i.e., 24/24 = 1.0. The disadvantage of this approach is the possibly high space complexity, especially in a site with many dynamically generated pages.

12.3.5 Classification and Prediction based on Web User Transactions

Classification is the task of mapping a data item into one of several predefined classes. In the Web domain, one is interested in developing a profile of users belonging to a particular class or category. This requires extraction and selection of features that best describe the properties of given the class or category. Classification can be done by using supervised learning algorithms such as decision trees, naive Bayesian classifiers, *k*-nearest neighbor classifiers, and Support Vector Machines (Chap. 3). It is also

possible to use previously discovered clusters and association rules for classification of new users (Sect. 3.5).

Classification techniques play an important role in Web analytics applications for modeling the users according to various predefined metrics. For example, given a set of user transactions, the sum of purchases made by each user within a specified period of time can be computed. A classification model can then be built based on this enriched data in order to classify users into those who have a high propensity to buy and those who do not, taking into account features such as users' demographic attributes, as well their navigational activities.

Another important application of classification and prediction in the Web domain is that of **collaborative filtering**. Most collaborative filtering applications in existing recommender systems use *k*-nearest neighbor classifiers to predict user ratings or purchase propensity by measuring the correlations between a current (target) user's profile (which may be a set of item ratings or a set of items visited or purchased) and past user profiles in order to find users in the database with similar characteristics or preferences [236]. Many of the Web usage mining approaches discussed earlier can also be used to automatically discover user models and then apply these models to provide personalized content to an active user [386, 445].

Basically, collaborative filtering based on the *k*-nearest neighbor (*k*NN) approach involves comparing the activity record for a target user with the historical records T of other users in order to find the top k users who have similar tastes or interests. The mapping of a visitor record to its neighborhood could be based on similarity in ratings of items, access to similar content or pages, or purchase of similar items. In most typical collaborative filtering applications, the user records or profiles are a set of ratings for a subset of items. The identified neighborhood is then used to recommend items not already accessed or purchased by the active user. Thus, there are two primary phases in collaborative filtering: the neighborhood formation phase and the recommendation phase. In the context of Web usage mining, *k*NN involves measuring the similarity or correlation between the target user's active session $\mathbf{u}$ (represented as a vector) and each past transaction vector $\mathbf{v}$ (where $\mathbf{v} \in T$). The top k most similar transactions to $\mathbf{u}$ are considered to be the neighborhood for the session $\mathbf{u}$. More specifically, the similarity between the target user, $\mathbf{u}$, and a neighbor, $\mathbf{v}$, can be calculated by the **Pearson's correlation coefficient** defined below:

$$sim(\mathbf{u},\mathbf{v}) = \frac{\sum_{i\in C}(r_{\mathbf{u},i} - \bar{r}_{\mathbf{u}})(r_{\mathbf{v},i} - \bar{r}_{\mathbf{v}})}{\sqrt{\sum_{i\in C}(r_{\mathbf{u},i} - \bar{r}_{\mathbf{u}})^2}\sqrt{\sum_{i\in C}(r_{\mathbf{v},i} - \bar{r}_{\mathbf{v}})^2}}, \tag{5}$$

where C is the set of items that are co-rated by $\mathbf{u}$ and $\mathbf{v}$ (i.e., items that

have been rated by both of them), $r_{\mathbf{u},i}$ and $r_{\mathbf{v},i}$ are the ratings (or weights) of some item i for the target user **u** and a possible neighbor **v** respectively, and $\bar{r}_{\mathbf{u}}$ and $\bar{r}_{\mathbf{v}}$ are the average ratings (or weights) of **u** and **v** respectively. Once similarities are calculated, the most similar users are selected.

It is also common to filter out neighbors with a similarity of less than a specific threshold to prevent predictions being based on very distant or negative correlations. Once the most similar user transactions are identified, the following formula can be used to compute the rating prediction of an item i for target user **u**.

$$p(\mathbf{u},i) = \bar{r}_{\mathbf{u}} + \frac{\sum_{\mathbf{v}\in V} sim(\mathbf{u},\mathbf{v}) \times (r_{\mathbf{v},i} - \bar{r}_{\mathbf{v}})}{\sum_{\mathbf{v}\in V} |sim(\mathbf{u},\mathbf{v})|}, \tag{6}$$

where V is the set of k similar users, $r_{\mathbf{v},i}$ are the ratings of those users on item i, and $sim(\mathbf{u}, \mathbf{v})$ is the Pearson correlation described above. The formula in essence computes the degree of preference of all the neighbors weighted by their similarity and then adds this to the target user's average rating, the idea being that different users may have different "baselines" around which their ratings are distributed.

The problem with the user-based formulation of the collaborative filtering problem is the lack of scalability: it requires the real-time comparison of the target user to all user records in order to generate predictions. A variation of this approach that remedies this problem is called **item-based collaborative filtering** [475]. Item-based collaborative filtering works by comparing items based on their pattern of ratings across users. Again, a nearest-neighbor approach can be used. The kNN algorithm attempts to find k similar items that are co-rated by different users similarly. The similarity measure typically used is the **adjusted cosine similarity** given below:

$$sim(i,j) = \frac{\sum_{\mathbf{u}\in U} (r_{\mathbf{u},i} - \bar{r}_{\mathbf{u}})(r_{\mathbf{u},j} - \bar{r}_{\mathbf{u}})}{\sqrt{\sum_{\mathbf{u}\in U} (r_{\mathbf{u},i} - \bar{r}_{\mathbf{u}})^2} \sqrt{\sum_{\mathbf{u}\in U} (r_{\mathbf{u},j} - \bar{r}_{\mathbf{u}})^2}}, \tag{7}$$

where U is the set of all users, i and j are items, $r_{\mathbf{u},i}$ represents the rating of user $\mathbf{u} \in U$ on item i, and $\bar{r}_{\mathbf{u}}$ is the average of the user **u**'s ratings as before. Note that in this case, we are computing the pair-wise similarities among items (not users) based on the ratings for these items across all users. After computing the similarity between items we select a set of k most similar items to the target item (i.e., the item for which we are interested in predicting a rating value) and generate a predicted value of user **u**'s rating by using the following formula

$$p(\mathbf{u}, i) = \frac{\sum_{j \in J} r_{\mathbf{u},j} \times sim(i, j)}{\sum_{j \in J} sim(i, j)}, \tag{8}$$

where J is the set of k similar items, $r_{\mathbf{u},j}$ is the rating of user $\mathbf{u}$ on item j, and $sim(i, j)$ is the similarity between items i and j as defined above. It is also common to ignore items with negative similarity to the target item. The idea here is to use the user's own ratings for the similar items to extrapolate the prediction for the target item.

12.4 Discussion and Outlook

Web usage mining has emerged as the essential tool for realizing more personalized, user-friendly and business-optimal Web services. Advances in data pre-processing, modeling, and mining techniques, applied to the Web data, have already resulted in many successful applications in adaptive information systems, personalization services, Web analytics tools, and content management systems. As the complexity of Web applications and user's interaction with these applications increases, the need for intelligent analysis of the Web usage data will also continue to grow.

Usage patterns discovered through Web usage mining are effective in capturing item-to-item and user-to-user relationships and similarities at the level of user sessions. However, without the benefit of deeper domain knowledge, such patterns provide little insight into the underlying reasons for which such items or users are grouped together. Furthermore, the inherent and increasing heterogeneity of the Web has required Web-based applications to more effectively integrate a variety of types of data across multiple channels and from different sources.

Thus, a focus on techniques and architectures for more effective integration and mining of content, usage, and structure data from different sources is likely to lead to the next generation of more useful and more intelligent applications, and more sophisticated tools for Web usage mining that can derive intelligence from user transactions on the Web.

Bibliographic Notes

Web usage mining as a complete process, integrating various stages of data mining cycle, including data preparation, pattern discovery, and interpreta-

tion, was initially introduced by Cooley et al. [114]. This initial work was later extended by Srivastava, et al. [505].

Proper data preparation is an essential activity that enables the discovery of actionable knowledge from usage data. A complete discussion of the stages and tasks in data preparation for Web usage mining can be found in the paper by Cooley et al. [115]. One of these tasks is that of sessionization of the user activity records in the log data which is generally accomplished through the use of various heuristics. Several heuristics were defined by Cooley et al. [115]. Berendt et al. [46] and Spiliopoulou et al. [498] introduced several additional sessionization heuristics, and developed a comprehensive framework for the evaluation of these heuristics in the context of various applications in Web usage mining. Much of the discussion of Sect. 12.1 is based on these sources.

One of the primary applications of Web usage mining has been in Web personalization and predictive user modeling. Initially, Web usage mining as a tool for personalization was introduced by Mobasher et al. [388]. More recent surveys of issues and techniques related to personalization based on Web usage mining can be found in the papers by Pierrakos et al. [445], Mobasher [386], and Anand and Mobasher [20].

Another important application of Web usage mining is the analysis of customer and visitor behavior in e-commerce and for Web marketing. Web usage mining applied to e-commerce data enables the discovery of important business intelligence metrics such as customer conversion ratios and lifetime values. A good discussion of lessons and challenges in e-business data analysis can be found in the paper by Kohavi et al. [286].