

2. the identification of user communities, and
3. the identification of critical situations (anomalies, security attacks, high traffic) in which the information system could be placed.

The first task enables both the customization and construction of adaptive Web sites and recommender systems, as well as the quality analysis of Web applications [9]. The second one is relevant for customer relationship management and business applications (e.g, e-commerce). The third one is essential for the management of computer security for a reliable and efficient information system and its Web front-end.

The paper is organized as follows. Section 2. provides the rationale and the background that justify the proposed approach. Section 3. describes the process of generation of conceptual logs. Section 4. provides the KDD scenarios made of queries and for each of them describes the obtained results. Finally Section 5. draws the conclusions.

2. Rationale and Background

Current Web applications are very complex and highly sophisticated software products, whose quality can heavily determine their success or failure. A number of methods have been proposed for evaluating Web application quality. In particular, Web usage mining methods are employed to analyze how users exploit the information provided by the Web site. For instance, showing those navigation patterns which correspond to high Web usage, or those which correspond to early leaving [11].

Web usage mining is mainly performed on the server side, and therefore is based on information found in log files, containing data about single user page access requests, represented in one of the available standard formats [3, 15]. Accordingly, Web usage mining approaches rely heavily on the preprocessing of log data as a way to obtain high level information regarding user navigation patterns and ground such information into the actual data underlying the Web application [16, 8, 4].

Preprocessing generally includes four main steps.

- *Data cleaning*, for removing information that is useless for mining purposes (requests for graphical contents, such as banners or logos; webspiders navigations).
- *Identification of user sessions*, for extracting full users navigation paths. This step can be very demanding [16], especially due to the adoption of proxy servers by applications, which do not allow the unique identification of users.
- *Content and structure information retrieval*, for mapping users page requests into the actual information of visited pages [1].
- *Data formatting*, for preparing data obtained through the previous three steps for being processed by the actual mining algorithms.

Notwithstanding the preprocessing efforts, in most of the cases the information extracted is usually insufficient and with much loss of the knowledge that is embedded in the application design. For example, among the previous steps, the retrieval of content structure requires the adoption of expensive methods for Web Structure Mining [7]: since any assumption can be made on how the Web application was designed, these methods consist of navigating the Web site to analyze the content and the structure of all the pages found, so as to exploit this information during the Web log mining step [6]. Even more, such approaches are ineffective on Web applications which are heavily based on dynamically created Web pages, since the navigation provides only a static view of site. Accordingly, although these projects have demonstrated the usefulness of exploiting the knowledge about the structure and content organization of a Web application [14, 5], the necessary preprocessing activities are rarely taken, as proved by the limited results reported in the literature (see [8] for a review).

Some efforts have been recently undertaken for enriching Web Log files, using Semantic Web techniques. In [13], authors exploit RDF annotations of static pages for mapping page URLs into a set of ontological entities. Within dynamic applications, the same mapping is achieved by analyzing the query strings enclosed within page URLs.

3. Integrating Web Conceptual Modeling and Web Usage Mining

The approach we present in this paper has the advantage of integrating Web usage mining goals directly into the Web application development process. Thanks to the adoption of a conceptual modelling method for Web application design, and of its supporting case tool, the generated Web applications embed a logging mechanism able to produce semantically enriched Web log files - the *conceptual log* [9]. Therefore, no extra effort is needed during or after the application development, for the extraction of meta-data about page semantics, or even for the construction of a Web site ontology. Furthermore, additional information included in the conceptual logs refers also to the identifier of the user crawling session, the identifier of specific instances that are published within dynamic pages, as well as to some data concerning the topology of the hypertext.

In the rest of this section, we will shortly illustrate the main features of the adopted design model, WebML (Web Modeling Language), and of the rich logs that WebML-based applications are able to produce.

3.1 The WebML Model and its Supporting CASE Tool

WebML (Web Modeling Language) is a visual language for specifying the content structure of a Web application, as well as the organization and presentation of such a content in a hypertext [2]. It mainly consists of the Data Model and the Hypertext Model.

The *WebML Data Model* adopts the Entity-Relationship primitives for representing the organization of the application data. The *WebML Hypertext Model* then offers some primitives for describing how data specified in the data model are published within the application hypertext, and consists of:

- a *composition model*, concerning the definition of pages and their internal organization in terms of elementary pieces of publishable content, called *content units*. Content units offer alternative ways of arranging data dynamically extracted from entities and relationships of the data schema.
- A *navigation model*, describing links between pages and content units, which have to be provided to facilitate information location and browsing.
- A *content management model*, which consists of a set of units for specifying operations for creating and updating content.

Besides the visual representation, WebML primitives are also provided with an XML-based representation, suitable to specify those properties that would not be conveniently expressed by a graphical notation. Figure 1 reports a simplified XML specification of a hypertext page, named *Research Area*, taken from the WebML specification of the application we analyze here (see Section 4.1).

The *Research Area* page publishes the description of a DEI research area, and the list of the current research fields covered by the area. Among others, it includes two content units. The first one is a *data unit* (lines 3-12) publishing the title (**att51**) and the textual description (**att57**) of a single instance of the **ResearchArea** entity of the data schema. The instance is retrieved from the database according to a *selector condition* (lines 7-11). The second content unit is an *index unit*. It lists some instances of the entity **ResAreaField** (lines 14), extracted from the database according to a condition (lines 18-22) based on the data schema relationship **Area2Field** that associates each research area with a set of correlated research fields.

WebRatio is the CASE tool supporting the WebML-based development¹. It is able to process WebML conceptual schemas, by translating their visual specification into concrete page templates. The core of WebRatio is a code generator, based on XML and XSL technologies, which is able to generate automatically the application code to be deployed on J2EE or .NET platforms. It covers all the relevant dimensions of a dynamic application: queries for data extraction from the data sources, code for managing the business logic, and page templates for the automatic generation of the front-end. The Web applications generated by WebRatio feature a three-tier architecture, whose application server enables an efficient execution of business components and the construction of dynamic pages.

¹<http://www.webratio.com>

```

01 <PAGE id="page3" name="Research Area">
02 <CONTENTUNITS>
03   <DATAUNIT id="dau84" name="Research Area"
04     entity="ent4" entity_name="Research_Area">
05     <DISPLAYATTRIBUTE attribute="att51" name="Area Title"/>
06     <DISPLAYATTRIBUTE attribute="att57" name="Area Description"/>
07     <SELECTOR>
08       <SELECTORCONDITION attributes="att58" att_name="OID"
09         id="cond90" sel_name="Area Selection"
10         predicate="eq" value="Selected_Area_OID"/>
11     </SELECTOR>
12   </DATAUNIT>
13   <INDEXUNIT id="inu9" name="Research Fields"
14     entity="ent19" entity_name="Res_Area_Field" >
15     <SORTATTRIBUTE attribute="att60" name="Field Title"
16       order="ascending"/>
17     <DISPLAYATTRIBUTE attribute="att60" name="Field Title"/>
18     <SELECTOR>
19       <SELECTORCONDITION relationship="rel17" rel_name="Area2Field"
20         id="cond40" sel_name="Fields_Selection"
21         predicate="in"/>
22     </SELECTOR>
23   </INDEXUNIT>
24   ... ..
25 </PAGE>

```

Figure 1. Simplified XML representation of the WebML schema of a page in the DEI Web application.

3.2 WebML Conceptual Logs

Conceptual logs [9] are standard log files enriched with information available in the WebML conceptual schema of the Web application, and with knowledge of accessed data. They are generated thanks to some modules, developed as extensions to the WebML/WebRatio framework, that are responsible for extracting and integrating logs from the Application Server, the WebML application runtime, and the application conceptual schema.

The use of conceptual logs introduces many advantages over the approaches usually followed in Web Usage Mining. First of all, they offer reach information that is not available with most traditional approaches. Also, they eliminate the typical Web Usage Mining preprocessing phase completely. In fact, we note that according to our approach:

- data cleaning is mainly encapsulated within the procedure that integrates the different log sources;
- the identification of user session is done by the WebML runtime, through the management of session IDs;
- the retrieval of content and structure information is unnecessary since all these information are available from the WebML conceptual schema.

Finally, since mining methods are applied specifically to a type of rich log files, it is possible to tailor these methods to improve their effectiveness in this particular context.

4. Mining Conceptual Logs

In this Section we describe the typology of information contained in the Web logs we processed and analyzed, and the KDD scenarios, i.e., the sequences of queries in a constraint-based mining language (MINE RULE) which allowed us to obtain interesting and actionable patterns for Web administrators, application designers and analysts.

4.1 DEI Web Application Conceptual Logs

The Web logs of the DEI Web site², record accesses on a very large application collecting one fourth of the overall clickstream directed to Politecnico di Milano, Italy. We collected the Web logs for the first consecutive 3 months in 2003. The original Web log stored by the Web server (**Apache**) was 60 MBytes large and is constituted by a relation that has the following information.

RequestID: the identifier of the request made by the user of a Web page;

IPcaller: IP address from which the request is originated; very often it is a proxy IP, that masks the real identification of the user.

Date: date of the request,

TS: time stamp of the request,

Operation: the kind of operation request (for instance, **get** or **put**)

Page URL: URL of the page to be transferred as a consequence of the request,

Protocol: transfer protocol used (such as TCP/IP),

Return Code: code returned by the Web server to the user,

Dimension: dimension in bytes of the page,

Browser: name of the browser from which the request is originated,

OS Type: type of the Operating System.

The additional data deriving from the WebML application design and from the application runtime debugging module include the following items:

Jsession: identifier of the user crawling session that spans over the single page requests. User crawling sessions are identified by an enabled Java browser by the Java thread identifier of a Web crawling.

Page: identifier of the page generated by the application server. Very often a page is generated dynamically but this identifier is always the same for each page.

UnitID: identifier of an atomic piece of information contained in a page. This identifier gives information on the type of content of a page.

OID: identifier of an object (for instance, a professor, a course, a publication) whose content is shown in a page. This object identifier is used by the application server to instantiate in a different way dynamic pages according to the object itself. For instance, all professor pages obey to the same template that shows personal data, photo, description of the curriculum vitae of the person and of its research area. Instead, the real information that is shown for each person changes accordingly to the professor, and therefore to the OID parameter that identifies the person.

Order: ordering number in which content units are presented in the page.

The Web Log contained almost 353 thousands user sessions for a total of more than 4.2 millions of page requests. The total number of pages (dynamic, instantiated by means of OIDs) was 38554. Each user session was constituted by an average of 12 page requests.

²<http://www.elet.polimi.it>

4.2 MINE RULE

MINE RULE is an SQL-like operator for mining association rules in relational databases. A detailed description can be found in [12]. This operator extracts a set of association rules from the database and stores them back in the database in a separate relation.

Let us explain the operator with the aid of a simple example on `WebLogTable`, containing the information of the conceptual log described in Section 4.1. The following MINE RULE statement extracts rules that aim to provide a description of the situations that generate frequently an error in the Web server (a favorite situation for attacks). `WebLogTable` has been grouped by `RequestId`; requested rules associate values of $\langle Operation, Browser, PageURL \rangle$ with values of `Returncode`. Selected rules will have a value of returned code corresponding to an error (WHERE clause). Rules will have a support and a confidence greater than the minimum requested values (respectively 0.2 and 0.4).

```
MINE RULE SituationsRuturnCodes AS
SELECT DISTINCT 1..n Operation, Browser, Page Url AS BODY,
                1..n Return Code AS HEAD, SUPPORT, CONFIDENCE
WHERE HEAD.Return Code LIKE '%error%'
FROM WebLogTable
GROUP BY RequestId
EXTRACTING RULES WITH SUPPORT:0.2, CONFIDENCE:0.4
```

This statement extracts each rule as an association of attribute values occurring within single tuples. In other statement cases, rule elements are constituted by values of the same attribute (e.g., Page URL) occurring in different tuples (e.g., requests) of the same group (e.g., date).

The main features of MINE RULE are:

- *Selection of the relevant set of data* for a data mining process. This feature is applied at different granularity levels, (row level or at the group level, with the *group condition*).
- Definition of the *structure of the rules* (single or multi-dimensional association rules) and cardinality of the rule body and head.
- Definition of *constraints applied at different granularity levels*. Constraints belong to different categories: constraints applied at the rule level (*mining conditions* instantiated by a WHERE clause), constraints applied at a group level (instantiated by a HAVING predicate) and constraints applied at the *cluster level* (*cluster conditions*). For lack of space we will not make use of cluster condition in this paper.
- Definition of *rule evaluation measures*. Practically, the language allows to define support and confidence thresholds.³ Support of a rule is computed on the total number of groups in which it occurs and satisfies the given constraints. Confidence is analogously computed (ratio between the rule support and the support of the body satisfying the given constraints).

4.3 Analysis of Conceptual Logs with MINE RULE

We have imported into a relational DBMS (MySQL) conceptual logs obtaining a table named `WebLogTable`. In this Section we describe in detail the KDD scenarios, composed of a sequence of pre-processing, mining and post-processing queries that we have designed for discovery of useful patterns in the Web logs. These queries can be conceived as a sort of *template* that can be used to gather descriptive patterns from Web logs useful to solve some frequent, specific or critical situations.

³Theoretically, also other measures, based on body and head support, could be used.

4.3.1 Analysis of Users that Visit the Same Pages.

This analysis aims at discovering Web communities of users on the basis of the pages that they frequently visited.

Pre-processing Query. The mining request could be preceded by a pre-processing query selecting only those page requests that occurred frequently (above a certain threshold) thus allowing to neglect the rare page requests.

Mining Query. This query finds the associations between sets of users (IP addresses) that have all visited a certain number of same pages. In particular this number of pages is given in terms of support of the rules. (In this example, support is computed over the requested pages, since grouping is made according to the requested pages). It is an open issue whether the discovered regularities among IP addresses occur because these IP addresses have been commonly used by the same users in their pages crawling. Indeed, this phenomenon could put in evidence the existence of different IP addresses dynamically assigned to the same users.

```
MINE RULE UsersSamePages AS
SELECT DISTINCT 1..n IPcaller AS BODY, 1..n IPcaller AS HEAD, SUPPORT, CONFIDENCE
FROM WebLogTable
GROUP BY Page Url
EXTRACTING RULES WITH SUPPORT:0.2, CONFIDENCE:0.4
```

In practice, in our experiments we discovered that the most frequently co-occurring IP addresses belong to web crawlers engines or big entities, such as universities. In the immediately lower support association rules we discovered (of course) the members of the various research groups. A similar query would occur if we wish to discover user communities which share the same user profile in terms of usage of the network resources. In this case, we would add constraints (in the mining condition, for instance) on the volume of the data transferred as a consequence of a user request. Examples of discovered patterns are the requests of frequent download of materials for courses, or documentation provided in user home pages.

Post-processing Query. As a post-processing query instead we could be interested in finding those pages that have been all visited most frequently by certain sets of users. This is a query that crosses-over extracted patterns and original data. With this request we could discard from the discovered patterns those ones belonging to web crawlers engines.

The following two query scenarios aim at performing Web structure mining.

4.3.2 Most Frequent Crawling Paths

Mining Query. This query returns sequences of pages (ordered by date of visit) frequently visited.

```
MINE RULE FreqSeqPages AS
SELECT DISTINCT 1..n Page Url AS BODY, 1..n Page Url AS HEAD, SUPPORT, CONFIDENCE
WHERE BODY.Date < HEAD.Date
FROM WebLogTable
GROUP BY IPcaller
EXTRACTING RULES WITH SUPPORT:0.3, CONFIDENCE:0.4
```

You can notice that in this case we grouped by user (`IPcaller`) and searched for sets of pages frequently occurring in the visits of a sufficient number of users (support). Notice also that we used a mining condition to constrain the temporal ordering between pages in antecedent and consequent of rules, thus ensuring the discovery of sequential patterns. In practice, examples of resulting patterns showed that requests of a research center page, or research expertise area, were later followed by the home page of a professor. This pattern was later used by Web administrators as a hint for restructuring the Web site access paths.

4.3.3 Units that Occur Frequently Inside Users Crawling Sessions.

The following query extracts associations between two sets of content units that appeared together in at least a certain number of crawling sessions.

```
MINE RULE UnitsSessions AS
SELECT DISTINCT 1..n UnitID AS BODY, 1..n UnitID AS HEAD, SUPPORT, CONFIDENCE
FROM WebLogTable
GROUP BY Jsession
EXTRACTING RULES WITH SUPPORT:0.05, CONFIDENCE:0.4
```

With this query we discovered patterns that helped Web designers to redesign the Web application. In fact, we obtained that the units that most frequently co-occurred in visits are the structural components of the Web site (indexes, overview pages, and so on).

4.3.4 Anomalies Detection.

This query tries to determine the associations between pages and users that caused a bad authentication error when making access to those pages. Therefore, this query wants to determine those pages that could be effectively used by callers as a way to enter illegally into the information system.

Pre-processing Query. The mining request was preceded by a pre-processing query selecting only those page requests that occurred a sufficient number of times. This discards those requests that have been mistakenly submitted by the user (a wrongly typed password), that if not repeated many times, cannot be considered an intrusion attempt.

```
MINE RULE BadAuthentication AS
SELECT DISTINCT 1..1 IPcaller AS BODY, 1..n Page Url AS HEAD, SUPPORT, CONFIDENCE
WHERE BODY.IPcaller = HEAD.IPcaller
FROM WebLogTable WHERE Return Code='bad authentication'
GROUP BY Date
EXTRACTING RULES WITH SUPPORT:0.03, CONFIDENCE:0.4
```

In this query we grouped source data by date, thus identifying patterns (association of users to page requests) that are frequent in time. Notice that mining condition `WHERE BODY.IPcaller = HEAD.IPcaller` ensures that page requests (head) effectively were originated by the callers associated to them (body). Examples of most retrieved patterns are attempts to change passwords, or downloading some reserved information.

4.3.5 High Traffic Users

Pre-processing query. Similarly to previous queries, also this data mining query could be preceded by a pre-processing step, selecting only the frequent page requests. (Indeed, rare page requests can be neglected).

Mining query. This query returns the associations between two sets of user IP addresses from which a request of pages is characterized by a large volume of data. This constraint is enforced by means of a preprocessing predicate `WHERE dimension>=1024` that selects only those requests generating high volume of traffic on the network.

```
MINE RULE HighTrafficUsers AS
SELECT DISTINCT 1..n IPcaller AS BODY, 1..n IPcaller AS HEAD, SUPPORT, CONFIDENCE
FROM WebLogTable
WHERE dimension>=1024
GROUP BY date
EXTRACTING RULES WITH SUPPORT:0.03, CONFIDENCE:0.4
```

Notice that we grouped the input relation by date thus identifying the users that request high volume pages frequently in time.

Post-processing query. A cross-over query can discover those pages originating the frequently occurring page requests.

As examples of discovered patterns there are the requests of frequent download of materials for courses from remote, or documentation provided in user home pages.

4.3.6 Errors Correlated to the Usage of an Operating System.

This query returns associations between the operating system and the error code frequently returned by the Web server.

```
MINE RULE OSErrors AS
SELECT DISTINCT 1..1 OStype AS BODY, 1..n Return Code AS HEAD, SUPPORT, CONFIDENCE
WHERE BODY.OStype=HEAD.OStype
FROM WebLogTable WHERE Return Code LIKE '%error%'
GROUP BY Date
EXTRACTING RULES WITH SUPPORT:0.01, CONFIDENCE:0.4
```

Notice the pre-processing predicate (`WHERE Return Code ..`) that selects only the page requests that result in some errors. This query is similar to query named `BadAuthentication` for the discovery of anomalies. We lunched also another similar query, requesting associations between a set of pages to an error and a browser. Both of them can be useful to test the reliability and robustness of a new Web application.

4.3.7 Users that Visit Frequently Certain Pages.

This request aims at discovering if recurrent requests of a set of pages from a certain IP exist. This puts in evidence the *fidelity* of the users to the service provided by the web site.

```
MINE RULE UsersPages AS
SELECT DISTINCT 1..1 ipaddress AS BODY, 1..2 Page Url AS HEAD, SUPPORT, CONFIDENCE
WHERE BODY.ipaddress = HEAD.ipaddress
FROM WebLogTable
GROUP BY requestId
EXTRACTING RULES WITH SUPPORT:0.01, CONFIDENCE:0.4
```

Examples of patterns we discovered are provided by the pages that allow the download of material (course slides, research reports).

One of the main advantages gained by the conceptual web logs is the knowledge of the information content of the pages. These content units can give us a more precise information on the reasons why certain pages are frequently requested by the users. For this purpose we have also considered another query (omitted here) retrieving associations between users and content units placed in pages frequently visited by those users. Patterns resulting from this request confirm that the most recurrent requests are download requests of materials from the Web site.

5. Conclusions

We presented a case study, the analysis of Web log of DEI Department, in which we applied and evaluated the usability and expressive power of a mining query language – MINE RULE. The Web log was a conceptual log, obtained by integration of standard (ECFL) Web server logs with information on web design application and web pages content information. The adoption of this case study had also the purpose to verify and experiment

the suitability of some KDD scenarios developed for inductive databases, and proved the possibility to employ them and the mining query languages in practice to solve real case problems.

Obtained patterns can be exploited for the definition of effective navigation and composition of hypertext elements to be adopted for improving the Web site quality. We also obtained some concrete examples of interesting or suspicious event that are useful to the end-users (web and system administrators).

The examples of query we provided show that the mining language is powerful, and at the same time versatile because its operational semantics seems to be the basic one. Indeed these experiments allow us to claim that Mannila and Imielinski's initial view on inductive databases [10] was correct: "There is no such thing as real discovery, just a matter of the expressive power of the query languages".

References

- [1] B. Berendt, A. Hotho, and G. Stumme. Towards semantic Web Mining. In *Proc. of the Semantic Web Conference, ISWC 2002, Sardinia, Italy, 2002*, LNCS, pages 264–278. Springer Verlag, June 2002.
- [2] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2002.
- [3] Apache Cocoon. Cocoon. <http://xml.apache.org/cocoon/>.
- [4] R. Cooley. *Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data*. PhD thesis, University of Minnesota, 2000.
- [5] R. Cooley, P.N. Tan, and J. Srivastava. *Discovery of Interesting Usage Patterns from Web Data*. LNCS/LNAI. Springer Verlag, 2000.
- [6] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.
- [7] Oren Etzioni. The world-wide web: Quagmire or gold mine? *Communications of the ACM*, 39(11):65–68, 1996.
- [8] Federico Michele Facca and Pier Luca Lanzi. Mining interesting knowledge from weblogs: A survey. Technical Report 2003.15, Dipartimento di Elettronica e Informazione. Politecnico di Milano., April 2003.
- [9] P. Fraternali, M. Matera, and A. Maurino. Conceptual-level log analysis for the evaluation of web application quality. In *Proceedings of LA-Web'03, Santiago, Chile, November 2003*. IEEE Computer Society, 2003.
- [10] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, November 1996.
- [11] R. Kohavi and R. Parekh. Ten supplementary analyses to improve e-commerce web sites. In *Proceedings of the Fifth WEBKDD Workshop: Webmining as a premise to effective and intelligent Web Applications*, ACM SIGKDD, Washington, DC, USA, 2003. Springer-Verlag.
- [12] R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Journal of Data Mining and Knowledge Discovery*, 2(2), 1998.
- [13] D. Oberle, B. Berendt, A. Hotho, and J. Gonzales. Conceptual User Tracking. In *Proc. of the First International Atlantic Web Intelligence Conference, AWIC 2003, Madrid, Spain, 2003*, LNAI 2663, pages 142–154. Springer Verlag, May 2003.
- [14] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the web. In *Proc. of CHI 96 Conference*. ACM Press, April 1996.

- [15] John R. Punin, Mukkai S. Krishnamoorthy, and Mohammed J. Zaki. Logml: Log markup language for web usage mining. In R. Kohavi, B. Masand, M. Spiliopoulou, and J. Srivastava, editors, *WEBKDD 2001 - Mining Web Log Data Across All Customers Touch Points, Third International Workshop, San Francisco, CA, USA, August 26, 2001. Revised Papers*, volume 2356 of *Lecture Notes in Computer Science*, pages 88–112. Springer, 2002.
- [16] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.