2.1.1 Aufzeichnung des Nutzerverhaltens – Erhebungstechniken und Datenformate

1 Die Kommunikationssituation im WWW

Bestimmend für die Beobachtung von Nutzerverhalten im WWW ist die dort herrschende, zunächst technisch bedingte Kommunikationssituation, die festlegt, was prinzipiell wie beobachtet werden kann.

Zu den Grundelementen der WWW-Kommunikation gehört die Identifizierung der teilnehmenden Rechner über eine numerische Adresse, Internet-Protokoll-Adresse (IP-Adresse) genannt, die aus derzeit vier (mit Version 6 des IP-Protokolls in Zukunft 16) Zahlen zwischen 0 und 255 besteht, z.B. 129.13.122.19. Ein spezieller Internetdienst namens Domain Name Service (DNS) ordnet numerische Adressen und Rechnernamen einander zu; so entspricht die obige IP-Adresse beispielsweise dem Rechnernamen viror.wiwi.uni-karlsruhe.de.

Drei Eigenschaften der Kommunikationssituation, die sich durch die Schlagworte Netzwerkstruktur, Verbindungslosigkeit und Funktionsschichten umreißen lassen, sind in unserem Zusammenhang von besonderer Bedeutung.

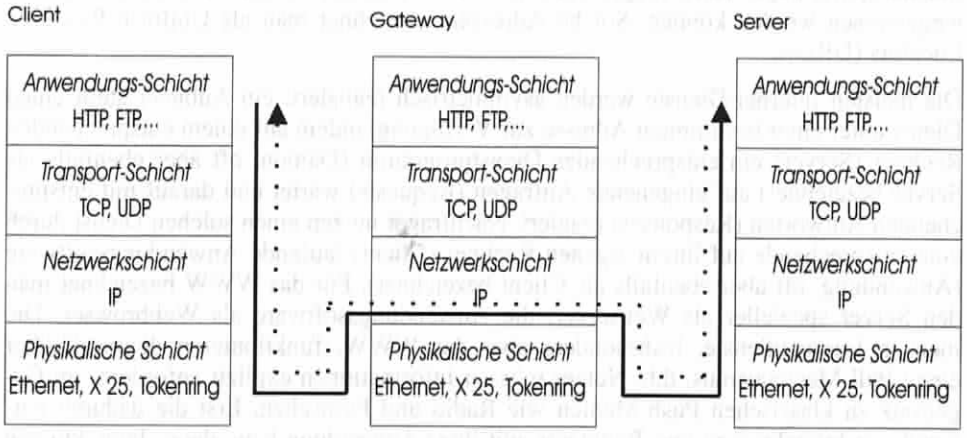
Die *Netzwerkstruktur* des Internet bedeutet, dass am Internet beteiligte Rechner miteinander in einem Netzwerk verbunden sind. Da naheliegenderweise nicht jeder Rechner direkt mit jedem anderen verbunden sein kann, kommunizieren Rechner nicht direkt miteinander, sondern i.d.R. vermittelt über eine Reihe weiterer Rechner im Netzwerk. Die Nachrichtenweiterleitung geschieht dabei in einer hierarchischen Art und Weise: jeder Rechner kann direkt mit unmittelbar benachbarten Rechnern kommunizieren (z.B. mit anderen Rechnern derselben Abteilung eines Unternehmens) und kennt einen Ansprechpartner für alle anderen Zieladressen. Ein solcher Ansprechpartner wird als *Router* oder *Gateway* bezeichnet. Ein Router verfügt über eine Tabelle, die für bestimmte Adressbereiche angibt, an welche Rechner eine Nachricht jeweils weitergeleitet werden soll. Nachrichten, die an keinen direkt mit dem Router verbundenen Rechner weitergeleitet werden können, werden an einen übergeordneten Router delegiert, usw. Der Laufweg einer Nachricht muss dabei nicht eindeutig und über die Zeit konstant sein. Um die Zuverlässigkeit des Netzwerkes zu erhöhen, können Nachrichten auf verschiedenen Wegen (d.h. über verschiedene intermediäre Router) weitergeleitet werden, so dass bei Ausfall eines Routers zwei Rechner in von diesem Router verbundenen Teilen des Netzwerkes über einen anderen Leitungsweg miteinander kommunizieren können (falls ein solcher besteht).

Die *Verbindungslosigkeit* des WWW besagt, dass Rechner nicht über feststehende Kanäle kommunizieren (wie z.B. das Telefon), sondern diskrete Nachrichten austauschen (wie z.B. Briefe per Post). Im Bereich des Internet heißen diese Nachrichten Pakete. Aufgrund der Verbindungslosigkeit müssen Nachrichten, die zum selben Kommunikationspartner gehören, erst als solche identifiziert werden.

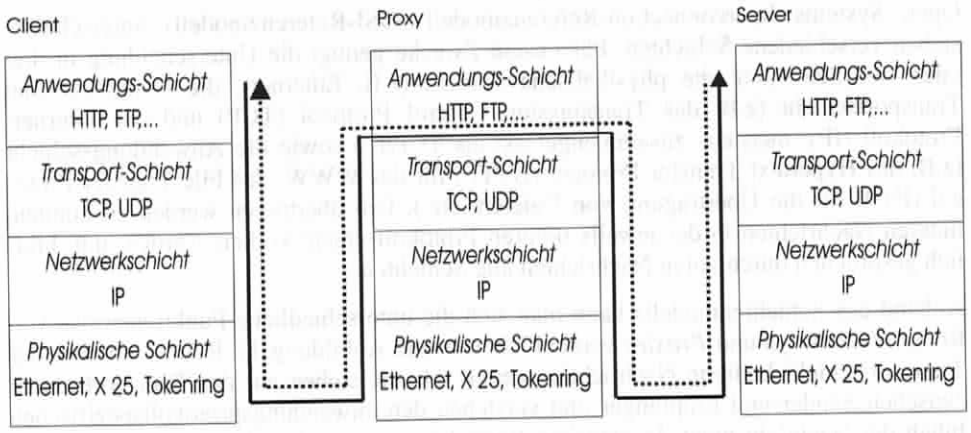
Unter *Funktionsschichten* der WWW-Kommunikation versteht man schließlich verschiedene Abstraktionsebenen, in denen eine Nachricht beschrieben werden kann. Das

Open Systems Interconnection-Referenzmodell (OSI-Referenzmodell) unterscheidet sieben verschiedene Schichten. Für unsere Zwecke genügt die Unterscheidung in drei oder vier Schichten: die physikalische Schicht (z.B. Ethernet), die Netzwerk- und Transportschicht (z.B. das Transmission Control Protocol (TCP) und das Internet-Protokoll (IP), meistens zusammengefasst als TCP/IP) sowie die Anwendungsschicht (z.B. das Hypertext Transfer Protocol (HTTP) für das WWW, das File Transfer Protocol (FTP) für die Übertragung von Dateien, etc.). Um übertragen werden zu können, müssen Nachrichten in der jeweils tieferen Protokollschicht kodiert werden, d.h. bildlich gesprochen durchlaufen Nachrichten alle Schichten.

Anhand des Schichtenmodells kann man sich die unterschiedliche Funktionsweise von Routern/Gateways und *Proxies* verdeutlichen (siehe Abbildung 1). Router leiten in der Transportschicht kodierte Nachrichten weiter, d.h. sie stehen auf der Transportschicht zwischen Sender und Empfänger und verstehen den anwendungsprotokollspezifischen Inhalt der Nachricht nicht. In manchen Bereichen ist man aber daran interessiert, dass Nachrichten je nach Inhalt gefiltert oder besonders behandelt werden, z.B. um Viren aus einem Firmennetzwerk herauszuhalten oder um häufig angefragte Resourcen zwischenzuspeichern und dann aus dem Zwischenspeicher zu bedienen (Cache). Hierfür werden Proxies eingesetzt. Proxies verstehen nur bestimmte Anwendungsprotokolle, man spricht dann entsprechend von HTTP-Proxies, FTP-Proxies etc. Sie decodieren Nachrichten, extrahieren den eigentlichen Zielrechner, stellen dann selbst die Anfrage an diesen Zielrechner (Stellvertreter-Funktion) und leiten dessen Antwort an die Rechner weiter, von denen die Anfrage ursprünglich stammte. Proxies stehen demnach auf der Anwendungsschicht zwischen Sender und Empfänger. Im Gegensatz zu Gateways hat ein Empfänger in der Regel keine Möglichkeit, den eigentlichen Sender festzustellen. (Die Stellvertreter-Funktion kommt teilweise auch bei Routern zum Einsatz, wenn diese als Firewalls fungieren und Rechner eines internen Netzes gegen Zugriffe von außen abschirmen sollen; man bezeichnet dies im Unterschied zum Proxy als *Masquerading*.)



a) Funktionsweise eines Routers/Gateways



b) Funktionsweise eines Proxies

Abbildung 1: Router/Gateways und Proxies im Vergleich

Um mehrere Dienste auf einem Rechner anbieten zu können, verwendet man sogenannte Ports: es handelt sich hierbei um Nummern, die an die IP- oder DNS-Adresse angehängt werden. Generische Adressen von Internet-Diensten lassen sich dann in der Form

[Protokoll]: //[IP-/DNS-Adresse]: [Port]/[Resource]

beschreiben, z.B. für das WWW als

http: //viror.wiwi.uni-karlsruhe.de: 80/webmining/ .

Bestimmten Diensten wie hier dem Webserver (HTTP-Protokoll) sind dabei bestimmte Standardports zugewiesen (sogenannte *well known numbers*), die bei Addressangaben weggelassen werden können. Solche Adressen bezeichnet man als Uniform Resource Locators (URLs).

Die meisten Internet-Dienste werden asymmetrisch realisiert: ein Anbieter stellt einen Dienst unter einer bestimmten Adresse zur Verfügung, indem auf einem entsprechenden Rechner (Server) ein entsprechendes Dienstprogramm (Dämon, oft aber ebenfalls als Server bezeichnet) auf eingehende Anfragen (Requests) wartet und darauf mit entsprechenden Antworten (Responses) reagiert. Nachfrager nutzen einen solchen Dienst durch eine entsprechende auf ihrem eigenen Rechner (Client) laufende Anwendungssoftware (Anwendung, oft aber ebenfalls als Client bezeichnet). Für das WWW bezeichnet man den Server spezieller als Webserver, die Anwendungssoftware als Webbrowser. Die meisten Internetdienste, insbesondere aber das WWW, funktionieren demnach über einen Pull-Mechanismus, d.h. Nutzer müssen Informationen explizit anfordern, im Gegensatz zu klassischen Push-Medien wie Radio und Fernsehen. Erst die dadurch notwendigen Interaktionen von Benutzern mit ihrer Anwendung bzw. deren Interaktionen mit der Serversoftware erlauben die Erfassung und Messung des Nutzerverhaltens.

2 Das Hypertext Transfer Protocol (HTTP)

Die Kommunikation zwischen Webbrowser und Webserver wird durch das WWW-Anwendungsprotokoll, das Hypertext Transfer Protocoll (HTTP) geregelt. Dieses Protokoll wird als versionierter Standard vom World Wide Web Consortium (W3C) verwaltet. Seit 1996 ist HTTP-Version 1.0 (Berners-Lee et al. 1996), seit 1999 parallel dazu HTTP 1.1 (Fielding et al. 1999) in Gebrauch.

Wie viele andere Internet-Protokolle auch beschreibt es die zur Anforderung und Versendung von Ressourcen benötigten Nachrichten in einem menschenlesbaren, zeilenorientierten Textformat. Anfrage und Antwort beginnen jeweils mit einer Anfrage- bzw. Statuszeile, enthalten dann eine beliebige Anzahl sogenannter Kopffelder (header fields), die Metadaten der Anfrage bzw. der gelieferten Resource beschreiben, und getrennt durch eine Leerzeile den Entitätsrumpf (entity body), der im Falle von Antworten die angeforderte Resource, also z.B. die gewünschte HTML-Datei, enthält, bzw. im Falle von Anfragen meistens leer ist (aber z.B. im Spezialfall von Webformularen die Eintragungen des Benutzers in die Formularfelder enthalten kann).

Abbildung 2 zeigt ein Beispiel für eine HTTP-Anfrage sowie eine entsprechende HTTP-Antwort. Wie man sieht, besitzt die Anfragezeile drei Einträge: der erste Eintrag gibt die verwendete HTTP-Methode an (hier i.d.R. meistens GET), der zweite Eintrag gibt die relative URL der nachgefragten Resource an und der dritte die gewünschte Protokoll-Version. Die Statuszeile der HTTP-Antwort besitzt ebenfalls drei Einträge: hier bezeichnet der erste die ausgehandelte Protokoll-Version, der zweite gibt einen Statuscode zurück und der dritte eine Kurzbeschreibung dieses Status. Der Statuscode gibt dem Client Auskunft darüber, ob und inwieweit der Server die Anfrage bearbeiten konnte: 200 signalisiert eine erfolgreiche Antwort, 404 einen Fehler aufgrund der Anfrage einer URL, die auf dem Server nicht vorhanden ist, 401 eine Rückfrage aufgrund der Anfrage einer Resource, für die eine Autorisierung erforderlich ist, etc.

HTTP-Anfrage (Client → Server):

```
GET /webmining/script/1/HTTP-5.xml HTTP/1.1
accept-charset: Any;q=1.0, *;q=0.9, utf-8;q=0.8
accept-language: en_US, en
host: viror.wiwi.uni-karlsruhe.de
accept: text/*;q=1.0, image/png;q=1.0, image/jpeg;q=1.0, image/gif;q=1.0, image/*;q=0.8, */*;q=0.5
user-agent: Mozilla/5.0 (compatible; Konqueror/2.1.2; X11)
referer: http://viror.wiwi.uni-karlsruhe.de/webmining/script/1/HTTP-6.xml
connection: Keep-Alive
accept-encoding: x-gzip; q=1.0, gzip; q=1.0, identity
```

HTTP-Antwort (Server → Client):

```
HTTP/1.1 200 OK
Date: Fri, 17 Aug 2001 07:48:25 GMT
Server: Apache/1.3.12 (Unix) (SuSE/Linux) ApacheJServ/1.1.2
Last-Modified: Fri, 17 Aug 2001 07:48:05 GMT
ETag: "e5819-33-3b7ccc35"
Accept-Ranges: bytes
Content-Length: 51
Connection: close
Content-Type: text/html

<html><body>
Eine HTML-Resource...
</body></html>
```

Erzeugter Logfile-Eintrag (eine Zeile, aus Platzgründen in drei Zeilen gebrochen):

```
129.13.122.23 - - [17/Aug/2001:07:48:25 +0200] "GET /webmining/script/1/HTTP-5.xml"
  200 51 "http://viror.wiwi.uni-karlsruhe.de/webmining/script/1/HTTP-6.xml"
  "Mozilla/5.0 (compatible; Konqueror/2.1.2; X11)"
```

Abbildung 2: HTTP-Anfrage und -Antwort sowie daraus erzeugter Logfile-Eintrag.

Die Kopfzeilen der Anfrage enthalten Informationen über die bevorzugte Sprache einer Resource (falls sie in verschiedenen Sprachen verfügbar ist, Kopffeld accept-language), der Webbrowser, mit dem die Anfrage getätigt wurde (user-agent), die URL der Seite, die den Link auf die angefragte Resource enthielt (referer), etc. Die Kopfzeilen der Antwort enthalten Meta-Informationen wie den Zeitpunkt der Bearbeitung der Anfrage (date), das Format der gesendeten Resource (content-type, im Beispiel eine HTML-Seite), die Länge der gesendeten Resource in Bytes (content-length) etc. Getrennt durch eine Leerzeile folgt dann im Entitätskörper schließlich die angefragte Resource, hier eine HTML-Datei. Die möglichen Kopffelder werden durch den HTTP-Standard festgelegt. Die Verwendung und Beachtung der meisten Felder ist jedoch optional und eine korrekte Verwendung steht zur Disposition der jeweiligen Software. Die Information der Kopffelder des eigenen Browsers kann man sich z.B. bei Schmidt-Thieme (2001) anzeigen lassen.

Im Prinzip könnte man Informationen über die Nutzung einer Website festhalten, indem man die HTTP-Anfragen und entsprechenden Antworten protokolliert. Aber wie man schon an dem kleinen Beispiel in Abbildung 2 sieht, fallen dabei sehr viele Informationen an, die in der Regel nicht von Belang sind. Deshalb wird nur ein Auszug der wichtigsten Informationen in einem sogenannten Logfile festgehalten.

Logfiles können verschiedene Formate besitzen. Die beiden wichtigsten sind das Common Logfile Format (Luotonen 1995) und das Combined Logfile Format, das ersteres um zwei weitere Einträge pro Anfrage erweitert. Um die Übernahme zusätzlicher Informationen in Logfiles einfacher zu gestalten, wurde das Extended Logfile Format entwickelt, das eine variable Protokollierung von Informationen erlaubt (Hallam-Baker/Behlendorf 1996). Sowohl Common als auch Combined Logfile Format sind Spezialfälle dieses Formats.

In allen diesen Fällen wird für jede Anfrage eine Zeile in das Logfile geschrieben. Einzelne Einträge werden durch Leerzeichen getrennt. In Tabelle 1 ist der Aufbau der beiden wichtigsten Logfile-Formate beschrieben. (Der Name des Referrer-Feldes wurde in Version 1.0 des HTTP-Standards versehentlich als referer geschrieben; in Version 1.1 des HTTP-Standards wird auf diesen Umstand hingewiesen, die falsche Schreibweise aus Kompatibilitätsgründen jedoch beibehalten.) In Abbildung 2 ist der für das Beispiel-Anfrage/Antwort-Paar erzeugte Logfile-Eintrag abgebildet. Die IP/DNS-Adresse des Clients stammt aus der TCP/IP-Schicht, nicht aus der HTTP-(Protokoll-)Schicht. Das Feld remoteuser ist für die Benutzerkennung vorgesehen, die bei einem clientseitigen Identifikationsdienst erfragt werden kann (StJohns 1993); da dieser jedoch nur innerhalb geschlossener Netzwerke vertrauenswürdig ist, der Dienst von den meisten Clientrechnern nicht angeboten wird und durch die Rückfrage eine unnötige Verzögerung entsteht, wird auf eine solche Rückfrage meistens verzichtet und das Feld bleibt leer. (Bei den meisten Webservern ist die Rückfrage beim Identifikationsdienst standardmäßig deaktiviert.) Im Feld authuser wird eine HTTP-Benutzerkennung festgehalten, falls auf entsprechende nicht-öffentliche Resourcen zugegriffen wird.

Die meisten Webserver bieten die Möglichkeit an, Nutzungsinformationen statt in einem Logfile in einer analog aufgebauten Tabelle einer Datenbank abzulegen. Einige Webserver können neben den Standardformaten zusätzliche proprietäre Formate schreiben; da die Menge der verfügbaren Informationen jedoch technisch durch die Kommunikationssituation bedingt ist und nicht durch das Format der Datenhaltung, ergeben sich weder neue Möglichkeiten noch andere Probleme.

Common Logfile Format

remotehost remoteuser authuser[date] „request“ status length

Combined Logfile Format

remotehost remoteuser authuser[date] „request“ status length “referrer” “agent”

Logfileeintrag	Beschreibung	Herkunft der Information
remotehost	Remote IP-Nummer (oder hostname, falls DNSLookup aktiviert)	TCP/IP-Socket
remoteuser	Benutzerkennung aus Identifikationsservice	Rückfrage bei clientseitigem Informationsdienst
authuser	Die vom Benutzer angegebene Benutzerkennung	Benutzereingabe auf Nachfrage des Servers (Statuscode 401)
date	Datum und Uhrzeit der Anfrage	Erzeugt (wie im Kopffeld date: der http-Antwort)
request	Die Anfragezeile in genau der Form, in der sie vom Client kam	Anfragezeile des HTTP-Anfrage-Headers
status	Der HTTP- Status- Code, der an den Client zurückgegeben wurde	Erzeugt (wie in der Statuszeile des HTTP-Antwort- Headers)
lenght	Die Länge des transferierten Inhalts in Bytes	Erzeugt (wie im Kopffeld content-length: der HTTP- Antwort)
referrer	URL der Resource, die den Hyperlink enthielt, dem der Benutzer zur angefordeten Resource gefolgt ist	Kopffeld referer: der HTTP-Anfrage
agent	Bezeichnung des Browsers, mittels dessen der Benutzer die Anfrage tätigt	Kopffeld user-agent:

Tabelle 1: Aufbau und Informationen von Logfile- Einträgen

3 Probleme bei der Beobachtung von Nutzerverhalten

Nutzungsinformationen stehen zunächst prinzipiell nur dem Betreiber einer Website nur für seine eigene Website zur Verfügung. Informationen zur Nutzung anderer Websites, beispielsweise seiner Konkurrenten, besitzt er in der Regel nicht.

Um die Nutzung einer Website durch einen individuellen Nutzer zu analysieren, wäre es optimal, wenn für jede Anfrage ersichtlich wäre, von welchem Nutzer sie stammt. Ne-

ben dem oben bereits angesprochenen Identifikationsdienst, der von den meisten Clients nicht angeboten wird und bei nicht kontrollierten Clients ohnehin nicht zuverlässig wäre, käme als Quelle für eine solche identifizierende Information noch das HTTP-Kopffeld from in Frage, in der Clients die Email-Adresse ihres Nutzers übermitteln können; alle gängigen Webbrowser lassen dieses Kopffeld jedoch leer. Das Grundproblem bei der Beobachtung von Nutzerverhalten besteht also darin, dass die Nutzung des WWW normalerweise anonym erfolgt und somit identifizierende Informationen nicht zur Verfügung stehen.

Selbst wenn Nutzer nicht absolut identifiziert werden können, wäre zumindest die Identifizierung von Nutzersitzungen (sessions) wünschenswert, d.h. die Identifizierung aller Anfragen, die zur selben Sitzung eines Nutzers gehören. Aufgrund der Verbindungslosigkeit des HTTP-Protokolls ist eine solche Zuordnung nicht gegeben (Nachrichten erreichen den Empfänger nicht über einen Kanal zwischen ihm und dem Sender, so dass alle Nachrichten, die aus demselben Kanal stammen, demselben Absender zugeordnet werden könnten, sondern Nachrichten werden als diskrete Nachrichten (Pakete) verschickt). Wir bezeichnen die beiden genannten Grundprobleme als *a) fehlende Nutzeridentifizierung* bzw. *b) fehlende Identifizierung von Sitzungen.*

Im anonymen Szenario stehen zur Identifizierung von Sitzungen bzw. Benutzern lediglich die IP-Adresse des Clients sowie dessen Browserkennung zur Verfügung. Als naive Grundregel werden alle Zugriffe vom gleichen IP-/Browserpaar dem gleichen Benutzer zugeordnet. Zur weiteren Unterteilung der Zugriffe eines Benutzers in einzelne Sitzungen bedient man sich meistens eines simplen Timeouts. (Meistens verwendet man basierend auf Pitkow/Kehoe 1996 ein Timeout von 30 Minuten.)

Zu den beiden genannten Grundproblemen treten jedoch eine Reihe weiterer Detailprobleme, die erstere in ihrer Auswirkung gravierend verschärfen. Zunächst ist es möglich, dass *c) Anfragen verschiedener Benutzer von der gleichen IP-Adresse* kommen, da beispielsweise alle Anfragen von Benutzern eines Proxies von diesem Proxy zu kommen scheinen. Das Problem kann häufig an sprechenden DNS-Namen von Proxies (z.B. http-proxy.uni-karlsruhe.de) bzw. an proxy-typischen Zusätzen in der Browserkennung erkannt werden. Benutzer hinter einem Proxy können eventuell an verschiedenen Browserkennungen (die sich i.d.R. schon bei verschiedenen Browserversionen unterscheiden) bzw. aufgrund verschiedener Pfade auf der Website auseinandergehalten werden.

Ebenso können *d) Anfragen eines Benutzers von verschiedenen IP-Adressen* stammen. Der einfachere Fall liegt bei dynamisch vergebenen IP-Adressen vor, wie sie beispielsweise bei Internet Service Providern (ISPs) üblich sind: die Rechner ihrer Kunden bekommen keine feste IP-Adresse zugeordnet, sondern erhalten bei jeder Einwahl dynamisch eine IP-Adresse aus einem Adresspool zugewiesen, d.h. zu verschiedenen Zeitpunkten/Einwahlen besitzt der gleiche Rechner eine unterschiedliche IP-Adresse. Einfacher ist der Fall deshalb, weil der Rechner zumindest für die Dauer einer Sitzung eine feste IP-Adresse besitzt, so dass hierdurch lediglich Probleme für die Nutzer-, nicht aber für die Sitzungsidentifikation entstehen.

Der schwerwiegendere Fall liegt vor, wenn die IP-Adresse nicht pro Sitzung/Einwahl wechselt, sondern pro Anfrage. Ähnlich wie Anbieter stark frequentierter Webangebote diese nicht mittels eines Rechners umsetzen können, sondern die Last durch Vorschalten eines Verteilers (Load Balancer) auf mehrere Rechner umlegen müssen, können auch Betreiber stark frequentierter Proxy-Dienste (wie z.B. große ISPs) diesen nicht über einen Rechner abwickeln, sondern verteilen die Last dynamisch auf mehrere Proxies. Während der den Proxy-Dienst in Anspruch nehmende Client einen festen Ansprechpartner hat (den Proxy-Verteiler), scheint für den angesprochenen Webserver jede Anfrage von einem der dynamisch wechselnden Proxies zu kommen. Betreiber solcher Proxycluster wie z.B. AOL publizieren i.d.R. deren IP-Adressen, so dass man alle diese IP-Adressen zu einer Gruppe zusammenfassen kann; man spricht hier von host munging (Pirolli/Pitkow 1999).

Bei der Beschreibung von Nutzerverhalten ist man nur an expliziten Nutzeraktionen interessiert, d.h. an solchen Zugriffen, die der Nutzer selbst explizit veranlasst hat. Die Struktur von Webseiten (HTML-Seiten) sowie die Funktionsweise der Webbrowser bringt es jedoch mit sich, dass auch *e) nicht explizit vom Nutzer veranlasste Anfragen* protokolliert werden. HTML-Seiten bestehen neben der Hauptresource, dem eigentlichen HTML-Textgerüst, i.d.R. aus einer Vielzahl weiterer Hilfsresourcen wie Icons, Stylesheets, etc. Auf Ebene des HTTP-Protokolls wird nicht zwischen diesen Resourcen unterschieden, jede muss für sich angefordert werden. Webbrowser unterstützen ihre Benutzer, indem sie diesen Vorgang automatisieren. Sie fordern zunächst die Hauptresource an, bestimmen die URLs aller dort referenzierten Hilfsresourcen, fordern dann diese Hilfsresourcen an, bauen die Seite entsprechend auf und präsentieren sie schließlich dem Benutzer. Während der Benutzer eine Aktion (click) ausführt und dafür eine Seite angezeigt bekommt (pageview), registriert der Webserver eine Vielzahl abgerufener Resourcen (hits). Bei der Analyse von Nutzerverhalten müssen diese nicht explizit mit Nutzeraktionen in Verbindung zu bringenden Anfragen getilgt bzw. in die explizite Anfrage der Hauptresource aggregiert werden (data cleaning).

Ein Spezialfall nicht explizit vom Nutzer stammender Zugriffe sind Zugriffe automatischer Clients, sogenannter Web Robots, die nicht von einem menschlichen Nutzer gesteuert werden. Solche Clients kommen beispielsweise zur Indizierung von Websites für Suchmaschinen, zur Extraktion von Produkt- und Preisinformationen für automatische Preisvergleicher etc. zum Einsatz. Da Web Robots technisch über den gleichen Mechanismus wie gewöhnliche Webbrowser zugreifen, das HTTP-Protokoll, sind sie zunächst nicht als solche zu erkennen. Man unterscheidet hier *gutartige* von *bösartigen* Robots: erstere geben sich in ihrer Agentkennung als Robots zu erkennen (z.B. googlebot für den Indizierer der Suchmaschine Google) und halten sich an den Robots Exclusion-Standard (Koster 1994), der Sitebetreibern die Möglichkeit einräumt, Robots nur Zugriff auf bestimmte Bereiche ihrer Site zu gewähren. Da dies siteweit über eine Resource mit der URL /robots.txt (oder per Webseite über das HTML-Metaelement) gesteuert wird, lassen sich gutartige Robots so auch anhand eines Zugriffs auf die /robots.txt-Resource identifizieren. Zu Robots gehörende Agentkennungen erkennt man entweder an einem sprechenden Namen (Bestandteile wie crawler, bot, etc.), durch Identifikation der Software oder anhand konstanter Verhaltensmuster wie etwa daran,

dass die meisten Robots nie etwas kaufen. Viele Anbieter von Web Mining-Software verwalten Listen mit zu Robots gehörenden Agentkennungen.

Da sowohl das Senden einer korrekten Agentkennung als auch die Einhaltung des Robots Exclusion-Standards dem Robot (bzw. seinem Betreiber) anheim gestellt sind, können bösartige Robots nur anhand ihres Navigationsverhaltens erkannt werden. Besonders rücksichtslose Robots kann man an einer extrem schnellen Zugriffsfolge erkennen (rapid fire). Da sich diese negativ auf die anderen Nutzern zur Verfügung stehende Leistung auswirkt, wird diese durch den Einbau von Verzögerungen zwischen je zwei Zugriffen vermieden; dadurch erfolgen die Zugriffe häufig in konstanten Zeitabständen, ein weiteres Erkennungsmerkmal für Web Robots neben ihrem sonstigen Browsingverhalten wie etwa dem vollständigen Abgriff einer Site oder einer Breitensuche (Tan/Kumar 2000).

Das vielleicht bekannteste Problem bei der Beschreibung von Nutzerverhalten sind *f) Anfragen, die nicht zum Server durchgereicht werden* und folglich dort auch nicht protokolliert werden können. Verantwortlich hierfür sind sogenannte Caches, die zuvor abgerufene Resourcen zwischenspeichern und bei erneutem Abruf nicht nochmals beim Server nachfragen, sondern die zwischengespeicherte Version liefern; Ziel eines Caches ist es, die Netzlast zu drosseln und die Antwortzeiten zu vermindern. Die harmlosere Variante ist der Browsercache, der jeweils den Nutzer eines Browsers dahingehend unterstützt, die Navigation durch zuvor gesehene Seiten zu beschleunigen. Zugriffe fehlen hier per Sitzung systematisch. Gravierender sind Caches in Proxies, die von mehreren Benutzern verwendet werden. Ein Proxycache kann Resourcen aus der Sitzung eines Benutzers speichern, um sie für Anfragen anderer Benutzer verfügbar zu machen. Zugriffe fehlen hier folglich unsystematisch. Cachebar sind grob gesagt zunächst nur statische Resourcen.

Das HTTP-Protokoll bietet Möglichkeiten, Caches anzuweisen, bestimmte Resourcen nicht zu cachen (Kopffelder expires und cache-control); da damit jedoch die Vorteile eines Caches, ein Performance-Gewinn und eine flüssigere Navigation der Website, verlorengehen, ist dies oft nicht wünschenswert. Fehlende Zugriffe sind oft an Pfadsprüngen erkennbar, d.h. an aufeinanderfolgenden Zugriffen desselben IP-/Browserpaars auf nicht direkt verlinkte Resourcen. Wahlweise können fehlende Zugriffe durch entsprechende Heuristiken ergänzt (Pfadvervollständigung) oder spezielle Verfahren für fehlende Werte eingesetzt werden (Gaul/Schmidt-Thieme 2000).

Aufgrund der Vielzahl genannter Probleme erfordert die Analyse von Web-Nutzungsdaten stets eine geeignete Aufbereitung (für eine ausführliche Darstellung der erforderlichen Schritte der Datenaufbereitung s. Kapitel 1.3 dieses Bandes).

4 Zusätzliche Techniken zur Benutzer-/ Sitzungsidentifizierung

Die naive Grundregel, das IP-/Browserpaar als Grundlage zur Identifizierung von Sitzungen oder gar Benutzern zu verwenden, ist nur in sehr speziellen Kontexten einsetzbar, etwa Websites im Intranet oder Websites mit B2B-Angeboten, wo Nutzer oft bereits aufgrund ihrer IP-Adresse identifiziert werden können, oder Websites mit sehr geringem Traffic, bei denen selten zwei Benutzer des gleichen Proxies aktiv sind. In allen typischen Kontexten jedoch, wie etwa größeren Informations- und E-Commerce-Angeboten, müssen zusätzliche Maßnahmen zur Benutzer-/Sitzungsidentifikation ergriffen werden.

Die zuverlässigste Art der Benutzeridentifikation geschieht über *a) explizite Authentifizierung*: Resourcen werden nicht mehr anonym angeboten, sondern nur nach vorheriger Anmeldung jeden Benutzers sowie durch Eingabe einer Benutzerkennung (Login) und eines Passworts. Eine solche Authentifizierung kann beispielsweise direkt über das HTTP-Protokoll abgewickelt werden: Fragt ein Client eine geschützte Resource an, antwortet der Webserver mit einer Aufforderung zur Authentifizierung (Statuscode 401); der Webbrowser fragt beim Benutzer nach einem passenden Login-/Passwortpaar und schickt dies für die Dauer der Sitzung mit jedem Request an den Herkunftsserver (Kopffeld *authorization*); der Webserver vergleicht die angebotene Autorisierungsinformation mit seiner Benutzer-Datenbank und schickt bei Übereinstimmung die angeforderte Resource.

Dem großen Vorteil einer eindeutigen Benutzeridentifikation steht der sehr hohe Interaktionsgrad als Nachteil gegenüber. Eine Benutzeregistrierung ist nur dann sinnvoll einsetzbar, wenn dem Benutzer über einen längeren Zeitraum ein Dienst zur Verfügung gestellt wird, für den er bereit ist, sich registrieren zu lassen und sich seine Zugangsdaten zu merken. Selbst aus Informations- und Zusatzangeboten geschnürte Pakete wie etwa Suns Java Developer Connection bieten die meisten Informationen ohne Registrierung an und erfordern erst für Downloads etc. eine Registrierung. Infolge des hohen Grades an Benutzerinteraktion ist die Authentifizierung für viele Bereiche nicht sinnvoll anwendbar, z.B. im E-Commerce.

Eine andere Art der Benutzeridentifizierung bietet die *b) Verwendung von Cookies*. Cookies sind kleine Datenpakete, die Clients ähnlich wie die Autorisierungsinformation bei jeder Anfrage an einen Server mitschicken (Kopffeld *cookie*). Diese Datenpakete enthalten ein Variable-/Wertpaar, das auf dem Client für die Dauer einer Sitzung (session cookie) oder auch darüber hinaus für einen bestimmten Zeitraum (Lebensdauer; persistent cookie) per Zielserver und partieller Ziel-URL gespeichert ist. Ein solcher Cookie kann auf verschiedene Art und Weise gesetzt werden: entweder im Rahmen einer Client-/Serverinteraktion innerhalb eines Kopffeldes (*set-cookie*) oder clientseitig durch Ausführen clientseitiger Skripten. Im ersten Fall kann man nochmals unterschei-

den zwischen dem Setzen von Cookies in dynamischen Resourcen, z.B. erzeugt von Servlets, Java Server Pages, etc. – jede Servertechnik bietet hier entsprechende High Level-Konstrukte zum Setzen solcher Cookies an – sowie für statische Resourcen, z.B. statische HTML-Seiten; letzteres wird vom Webserver erledigt, die genauen Einstellungen hängen im Detail von der jeweiligen Implementation ab (für Apache kann man beispielsweise das Modul usertrack verwenden). Clientseitige Cookies hingegen werden in der Regel mittels Javascript gesetzt. Die Cookie-Kopffelder sind nicht Bestandteil des HTTP-Kernstandards, sondern einer Erweiterung (Kristol/Montulli 1997).

Zur Benutzeridentifikation können Cookies eingesetzt werden, indem man als Wert eine künstliche eindeutige Kennzeichnung speichert, im einfachsten Fall z.B. eine fortlaufende Nummer (in der Praxis verwendet man komplexere Kennzeichnungen, damit Kennzeichnungen aktiver Sitzungen nicht erraten und zugehörige Sitzungen nicht entführt werden können). Diese Cookie-IDs können nun als zusätzliches Feld in einem Logfile (z.B. im Extended Logfile Format) abgelegt werden. Zugriffe mit gleicher Cookie-ID stammen vom gleichen Benutzer.

Der Hauptvorteil von Cookies besteht darin, dass keinerlei Benutzer-Interaktion von Nöten ist; das Setzen und Senden von Cookies geschieht oft völlig unbemerkt vom Nutzer. Zu den Nachteilen von Cookies zählen, dass verschiedene Browser Cookies nicht teilen, so dass Benutzer mit mehreren Browsern als verschiedene Benutzer erscheinen, sowie dass Cookies per Benutzeraccount abgelegt werden, so dass Benutzer desselben Accounts als derselbe Benutzer erscheinen (problematisch beispielsweise in Internet-Cafes). Der gravierendste Nachteil besteht jedoch darin, dass das Senden von Cookies im Belieben des Browsers bzw. seines Nutzers steht, d.h. eine Kooperation des Webbrowsers notwendig ist. Das Deaktivieren von Cookies war lange Zeit nur durch besondere Einstellungen und in eher grobschlächtiger Art und Weise vorzunehmen, so dass dies einer kleinen Minderheit erfahrener Nutzer vorbehalten war. Moderne Browser verfügen jedoch über eine feinkörniges Cookie-Management sowie oft über (für den Benutzer) sinnvolle Grundeinstellungen, so dass ähnlich wie bei der Authentifizierung damit gerechnet werden muss, dass Benutzer nur noch Cookies für Websites aktivieren, von denen sie sich einen Mehrwert versprechen.

Als besonders innovativ muss in diesem Zusammenhang die Umsetzung im Mozilla/Netscape-Browser erwähnt werden (http: //mozilla.org), der diesen Vorgang für Benutzer noch weiter automatisiert, indem er Cookie-Einstellungen von Angaben in der Privacy Policy der Website abhängig macht. Privacy Policies werden durch den W3C-Standard *Platform for Privacy Preferences Project* (P3P; Cranor et al. 2001) definiert. Hier teilt der Betreiber einer Website seinen Besuchern in computerlesbarer Form mit, welche Nutzungsdaten er zu welchen Zwecken erfasst. Benutzer können Cookies in Abhängigkeit von diesen Privacy Policies automatisch gestatten oder blocken.

Obwohl Cookies derzeit die am weitesten verbreitete Technik zum Identifizieren von Benutzern darstellen, werfen die immanenten, zum Teil aber auch erst durch Missbrauch (siehe Abschnitt 5) entstandenen Risiken für den Schutz der Privatsphäre mittelfristig die Frage auf, wie lange diese Technik von der Mehrheit der Benutzer noch akzeptiert und damit mit Erfolg einsetzbar bleibt.

Ein weniger invasives Mittel der Identifizierung von Sitzungen ist die *c) Verwendung sitzungspezifischer URLs*. Darunter versteht man, dass der Webserver bei der ersten Anfrage einer Resource von einem Client eine eindeutige Kennzeichnung erzeugt (Session-ID) und alle zurückgelieferten HTML-Seiten mittels dieser Kennzeichnung instrumentiert. Hyperlinks instrumentierter Seiten beinhalten neben der ursprünglichen Information, welche Resource geliefert werden soll, noch die Session-ID; solche URLs können z.B. wie folgt aussehen:

/webmining/? id=wz562812673,

wobei /webmining/ die nachgefragte Resource bezeichnet und wz562812673 die als Parameter spezifizierte Session-ID. Der Hauptvorteil dieses Verfahrens liegt darin, dass es rein serverseitig ablaufen kann und nicht auf eine aktive Browser-Kooperation angewiesen ist. Nachteile sind ein gewisser Performance-Verlust durch die Instrumentierung (dynamische Resourcen) und die Nicht-Cachebarkeit sowie die dadurch entstehenden, für den Benutzer sichtbaren hässlichen URLs. Die Instrumentation von Links gehört zu den typischen Aufgaben eines Applikationsservers.

Eine Variante dieses Verfahrens ist die *d) Ablage von Session-IDs in verborgenen Eingabefeldern* von HTML-Formularen. Diese Technik kommt im wesentlichen bei Präsentation mehrseitiger Online-Fragebögen zum Einsatz. Sie hat (bei Verwendung der HTTP-Methode POST) den zusätzlichen Vorteil, dass die Session-IDs für Benutzer nicht sichtbar sind, allerdings erkauft durch den Nachteil, dass zur Navigation HTML-Formulare verwendet werden müssen und manche Browser ihre Nutzer darauf aufmerksam machen, dass dabei Informationen an den Server übertragen werden.

Da Session-IDs in URLs trotz fehlender Standardisierung unschwer zu erkennen sind, wäre es sehr einfach möglich, clientseitige Filter zu konstruieren, die sie aus den Links/URLs wieder entfernen. Zu einer aktiven Obstruktion der Sitzungsidentifikation seitens eines Webbrowsers sind uns derzeit jedoch keine Ansätze bekannt und erscheinen uns auch nicht wahrscheinlich, so dass mit der Link-Instrumentierung zumindest zur Identifikation von Sitzungen, wenn auch nicht von Benutzern, auf längere Zeit ein geeignetes Instrument zur Verfügung stehen dürfte.

5 Jenseits von Server-Logfiles

Die zuvor skizzierten Probleme, die von einer Erfassung des Nutzerverhaltens mittels Webserver-Logfiles ausgehen, haben zu einer Vielzahl von Erweiterungen sowie konkurrierender bzw. ergänzender Techniken geführt. Zu den *a) Erweiterungen serverseitiger Techniken* sind sogenannte Webserver-Plugins zu zählen, die im wesentlichen aber Funktionalitäten wie Cookies zur Verfügung stellen, über die moderne Webserver von Hause aus verfügen, das Erfassen von Nutzungsinformationen auf IP-Ebene sowie die Erfassung von Nutzungsinformationen auf Applikationsserver-Ebene.

Das Erfassen von Nutzungsinformationen auf einer tieferen Netzwerkschicht kann unabhängig vom Webserver mittels eines sogenannten Paketsniffers realisiert werden, der IP-Pakete protokolliert und auswertet. Der Vorteil solcher Paketsniffer besteht darin, dass sie an zentraler Stelle Nutzungsinformationen für eine heterogene Serverfarm erfassen können (z.B. verschiedene Dienste oder nur verschiedene Implementationen des gleichen Dienstes; Logfiles unterschiedlicher Dienste müssen normalerweise in geeigneter Weise zusammengeführt werden, wobei weiteren Problemen, wie etwa einer Synchronisierung, begegnet werden muss), was durch den Nachteil erkauft wird, dass sie entsprechende Anwendungsprotokolle nicht verstehen und i.d.R. mit verschlüsselter Information nichts anzufangen wissen.

Mittels Applikationsserver-Logfiles kann man dem Problem dynamischer Resourcen begegnen. Dynamische Resourcen wie etwas Ergebnisse von Suchanfragen, aus Informationen einer Produkt-Datenbank generierte Präsentationsseiten, etc. werden meistens über URL-Parameter gesteuert. Im Rahmen des Preprocessings von Nutzungsdaten müssen solche Parameter in entsprechende Ereignisse etwa der Form „Produktpräsentation X betrachtet" zurückübersetzt werden. Applikationsserver, mittels derer die Erzeugung der genannten dynamischen Seiten i.d.R. realisiert wird, bieten die Möglichkeit, diese High-Level-Ereignisse direkt in einem Applikationsserver-Logfile festzuhalten. Dies bietet, auch im Zusammenhang mit einer zentralen Verwaltung verwendeter serverseitiger Techniken wie Session-IDs, Link-Instrumentierung, Cookies etc., eine deutliche Vereinfachung des Preprocessings (Kohavi 2001). Jedoch muss darauf hingewiesen werden, dass durch den Einsatz von Applikationsservern keine prinzipiell neuen Möglichkeiten entstehen, d.h. die beschriebenen Schwierigkeiten der Benutzer- und Sitzungsidentifikation bleiben bestehen. Als Nachteil ist eine Festlegung auf eine höhere Schicht, die Applikationsserver-Schicht, als zentrale Schicht zu sehen; nur von Diensten, die in den verwendeten Applikationsserver integriert werden können, lässt sich die Nutzung so konsistent erfassen.

Zusätzliche Informationen über den ungefähren geographischen Standort des Clients (bzw. des von ihm benutzten Proxies) lassen sich durch den Einsatz von *b) Geo-Location-Software* gewinnen. Mittels des Internet Control Message Protocols (ICMP) wird der Weg zu jedem Client bestimmt, indem jeder Router auf dem Weg zum Client aufgefordert wird, eine Bestätigung über den Erhalt des Paketes an den Absender zurückzuschicken (traceroute, ICMP-Typ 30). Kennt man den Nachrichtenweg sowie die geographische Position wichtiger Router, erhält man Aufschluss über die geographische Position des Clients. Da das ICMP-Protokoll eigentlich zur Lokalisierung von Netzwerkproblemen seitens von Administratoren ausgelegt ist, stellt seine Verwendung zur Informationsgewinnung möglicherweise einen Missbrauch dar.

Jenseits von Server-Logfiles sind *c) Banner-Netzwerke* anzusiedeln. Auf verschiedenen Websites werden Banner eingeblendet, die von einem zentralen Ad-Server geladen werden (z.B. DoubleClick). Die ursprünglich angeforderte Resource steht als Referrer zur Verfügung (der ursprüngliche Referrer hingegen kann auf dem Ad-Server nicht beobachtet werden). Das Logfile des zentralen Ad-Servers registriert so indirekt alle Zugriffe eines Benutzers auf allen Websites, auf denen entsprechende Banner geschaltet

sind, d.h. Nutzungsinformationen über die Grenzen einer Website hinaus. Insbesondere unter Zuhilfenahme einer starken Identifizierungstechnik wie den Cookies können hiermit von Betreibern großer Bannernetzwerke umfangreiche Nutzungsinformationen gewonnen werden. Hersteller von Webbrowsern haben darauf schon reagiert, indem Cookies, die nicht für den ursprünglich angesprochenen Server gedacht sind (third party cookies), besonders misstrauisch behandelt werden; die Möglichkeit einer konsequenten Deaktivierung der Referrer-Information für solche Websites ist uns indes nicht bekannt.

Mittels sogenannter *d) Wanzen* kann die Erfassung von Nutzungsinformationen größtenteils auf den Client verlagert werden. Unter Wanzen versteht man kurze Javascript-Programme oder Java-Applets, die eingebettet in eine HTML-Seite von Clients geladen werden. Auf dem Client sammeln sie dann verschiedene Daten (z.B. den ursprünglichen Referrer, die Auflösung der Bildschirmdarstellung, die Größe des Browserfensters, etc.) und senden diese, i.d.R. kodiert in eine Anfrage-URL für eine „leere" Resource (1-Pixel-Bild), an einen zentralen Tracking-Server.

Wanzen haben den Vorteil, dass sie anders als serverseitige Techniken keine Probleme bei eventuellem Caching von Resourcen haben, da die Wanze mit der Webseite gecacht und bei jedem Aufbau der Seite aktiv wird. Zweitens trennt der Einsatz von Wanzen die Webserver- von der Logging-Infrastruktur: der Tracking-Server ist i.d.R. vom Webserver verschieden. Letzteres hat insbesondere dazu geführt, dass eine Erfassung von Nutzungsinformationen mittels Wanzen als Dienstleistung von vielen Drittanbietern angeboten wird.

Als Nachteil steht dem wiederum die Notwendigkeit einer erhöhten Browser-Kooperation entgegen. Das Ausführen von Javascript- oder Java-Programmen gilt nicht nur als potentielle Privacy-, sondern sogar als potentielle Sicherheitslücke, weswegen Nutzer die Ausführung solcher Programme unterbinden können. Für Java steht dafür ein feinkörniges Policy-Konzept zur Verfügung. Berücksichtigt man weiterhin, dass der derzeit am weitesten verbreitete Webbrowser Internet Explorer in der neuesten Fassung standardmäßig kein Java mehr unterstützt, scheint eine Implementation von Wanzen mittels Java (Shahabi et al. 1997) wenig sinnvoll. Solange, wie derzeit selbst bei einigen neueren Browsern, die Kontrolle über die Javascript-Ausführung noch sehr grobschlächtig, d.h. Javascript nur global aktivierbar oder deaktivierbar ist, erscheint aufgrund zunehmender Verwendung von Javascript zur Gestaltung von Webseiten eine Deaktivierung eher unwahrscheinlich, so dass Javascript-Wanzen kurzfristig eine interessante Alternative zu Server-Logfiles darstellen. Mittelfristig hingegen dürfte diese Lücke bald geschlossen sein.

Bei deaktiviertem Javascript können Wanzen beispielsweise auf einen statisch referenzierten Icon rekurrieren, so dass zumindest protokolliert werden kann, welche Seiten abgerufen wurden (auch wenn dann z.B. die Information über den ursprünglichen Referrerer nicht mehr zur Verfügung steht); Wanzen funktionieren in diesem Fall ganz analog wie Banner-Netzwerke.

6 Ausblick

Zum Erfassen von Informationen über die Nutzung von Webangeboten steht mit den Server-Logfiles eine grundsätzlich solide Technologie zur Verfügung. Eine Vielzahl technisch bedingter Probleme jedoch verschlechtert die Güte der Daten dramatisch, insbesondere was die Identifizierung von Benutzern und Sitzungen betrifft. Verschiedene zusätzliche Maßnahmen können ergriffen werden, um eine bessere Identifizierung zu bewerkstelligen, insbesondere die Benutzeridentifikation mittels Cookies sowie die Sitzungsidentifikation mittels Link-Instrumentierung sind hier hervorzuheben. Zusätzlich können weitere Techniken zur Erfassung von Nutzerverhalten wie etwa eine clientseitige Erfassung mittels Wanzen eingesetzt werden. Bei jeder Erfassung von Nutzungsinformationen ist zu berücksichtigen, dass das WWW derzeit als anonymes Medium funktioniert und der Beobachtung von Nutzerverhalten der Schutz der Privatsphäre der Nutzer entgegensteht, rechtlich wie technologisch. Technologisch ist ein starker Trend zu intelligenteren Webbrowsern und entsprechenden Webstandards wie z.B. P3P zu verzeichnen, der auch gewöhnliche Alltagsnutzer in Stand setzt, Informationen nur noch dort zu hinterlassen, wo sie einen entsprechenden Gegenwert dafür erhalten. In diesem Sinne ist ein Umschwung vom Informationssammel-Wildwuchs der letzten Jahre zu stärker an „permission-based marketing" orientierten Ansätzen zu erwarten.

Literatur

Berners-Lee, T.; Fielding, R.; Frystyk, H. (1996): Hypertext transfer protocol – HTTP/1.0. Request for Comments 1945. http://www.w3.org/Protocols/rfc1945/rfc1945.

Cooley, R.; Mobasher, B.; Srivastava, J. (1999): Data preparation for mining world wide web browsing patterns. In: Journal of Knowledge and Information Systems 1/1, S. 5–32.

Cranor, L.; Langheinrich, M.; Marchiori, M.; Presler-Marshall, M.; Reagle, J. (2001): The platform for privacy preferences 1.0 (P3P1.0) specification. W3C Working Draft, 28. September 2001. http: //www.w3.org/TR/P3P/.

Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T. (1999): Hypertext transfer protocol – HTTP/1.1. Request for Comments 2616. http: //www.w3.org/Protocols/.

Gaul, W.; Schmidt-Thieme, L. (2000): Frequent generalized subsequences — a problem from web mining. In: Gaul, W.; Opitz, O.; Schader, M. (Eds.): Data Analysis, Scientific Modeling and Practical Application, Berlin, S. 430–445.

Hallam-Baker, Ph. M.; Behlendorf, B. (1996): Extended log file format. W3C Working Draft WD-logfile-960323. http: //www.w3.org/TR/WD-logfile.

Kristol, D.; Montulli, L. (1997): HTTP state management mechanism. Request for Comments 2109. http: //www.faqs.org/rfcs/rfc2109.html.

Kohavi, R. (2001): Mining e-Commerce data: the good, the bad, and the ugly. In: Provost, F.; Srikant, R.: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Bases (KDD 2001), New York, S. 8–13.

Koster, M. (1994): A standard for robot exclusion. http: // www.robotstxt.org/wc/norobots.html.

Luotonen, A. (1995): The common logfile format. http: //www.w3.org/pub/WWW/Daemon/User/Config/Logging.html.

Pirolli, P.; Pitkow, J. E. (1999): Distributions of surfers' paths through the World Wide Web, Empirical characterization. In: World Wide Web 2/1–2, S. 29-45.

Pitkow, J. E.; Kehoe, C. M. (1996): Emerging trends in the WWW user population. In: Communications of the ACM 39/6, S. 106–108.

Schmidt-Thieme, L. (2001): Web Mining. Vorlesung an der Universität Karlsruhe im Wintersemester 2001. http: //viror.wiwi.uni-karlsruhe.de/webmining/. (Die im Text genannte dynamische Resource zur Visualisierung von Headerfeldern befindet sich unter http: //viror.wiwi.uni-karlsruhe.de/webmining/script/2/HTTP-5.xml.)

Shahabi, C., Zarkesh, A.M., Adibi, J.,& Shah, V. (1997): Knowledge discovery from users' web-page navigation. In: Seventh International Workshop on Research Issues in Data Engineering (RIDE '97), High Performance Database Management for Large-Scale Applications, April 7–8, Birmingham, UK.

StJohns, M. (1993): Identification protocol. Request for Comments 1413. http: //www.faqs.org/rfcs/rfc1413.html.

Tan, P.-N.; Kumar, V. (2000): Modeling of web robot navigational patterns. In: Workshop on Web Mining for E-Commerce — Challenges and Opportunities (WebKDD 2000), August 20, 2000, Boston, MA, USA.

Underhill, P. (2000): Why we buy – the science of shopping. New York.