Universität Koblenz-Landau
Fachbereich Informatik
Institut für Softwaretechnik

Diplomarbeit

Konzeption eines Web-Usage-Mining-Systems für High-Traffic-Anwendungen

Timo Paulwitz <paulwitz@chamaeleon.de>
September 2006

Erstgutachter:	Prof. Dr. Kurt Lautenbach	<laut@uni-koblenz.de>
Zweitgutachter:	Alexander Pinl	<apinl@uni-koblenz.de>
Betreuer:	Prof. Dr. Kurt Lautenbach	<laut@uni-koblenz.de>
	Alexander Pinl	<apinl@uni-koblenz.de>
	Martin Hümmerich	<huemmerich@chamaeleon.de>

Inhaltsverzeichnis

1 Hintergrund **1**
- 1.1 Web-Usage-Mining . . . 2
- 1.2 Schwachstellen existierender Systeme . . . 2
- 1.3 Ziele dieser Arbeit . . . 4
  - 1.3.1 Präzision . . . 5
  - 1.3.2 Situative Relevanz . . . 5
  - 1.3.3 Komplexität riesiger Datenmengen . . . 6

2 Termini dieser Arbeit **7**
- 2.1 Domänen . . . 7
- 2.2 Konkrete Mengen . . . 8
- 2.3 Entitäten . . . 9
- 2.4 Weitere Konstrukte . . . 9
- 2.5 Anmerkungen zu $\mathcal{A} \subseteq \mathbb{A}$ . . . 10

3 Messwerte **13**
- 3.1 Hits . . . 15
  - 3.1.1 Eigenschaften . . . 15
  - 3.1.2 Bedeutung . . . 16
- 3.2 Verweilzeit . . . 18
  - 3.2.1 Eigenschaften . . . 18
  - 3.2.2 Bedeutung . . . 19
- 3.3 Besucher . . . 21
  - 3.3.1 Eigenschaften . . . 21
  - 3.3.2 Bedeutung . . . 21
- 3.4 Besuche . . . 24
  - 3.4.1 Eigenschaften . . . 24
  - 3.4.2 Bedeutung . . . 25
- 3.5 Vorgänger und Eingangspfade . . . 26
  - 3.5.1 Eigenschaften . . . 26
  - 3.5.2 Bedeutung . . . 26

3.6 Nachfolger und Zielpfade 29
3.6.1 Eigenschaften 29
3.6.2 Bedeutung 29
3.7 Durchschnittliche Pfadlänge 30
3.7.1 Bedeutung 30
3.8 Session-Starter 31
3.8.1 Eigenschaften 31
3.8.2 Bedeutung 31
3.9 Session-Killer 33
3.9.1 Eigenschaften 33
3.9.2 Bedeutung 33
3.10 Kookurrenz 35
3.10.1 Eigenschaften 35
3.10.2 Bedeutung 36
3.11 Klickpfade 37
3.11.1 Eigenschaften 37
3.11.2 Bedeutung 38
3.12 Suchbegriffe 40
3.12.1 Eigenschaften 40
3.12.2 Bedeutung 41

4 Benutzeridentifikation 43
4.1 Host-Tracking 44
4.1.1 Eigenschaften 44
4.1.2 Bedeutung 44
4.2 Session-Tracking 47
4.2.1 Eigenschaften 47
4.2.2 Bedeutung 49
4.3 Authentication-Tracking 50
4.3.1 Eigenschaften 51
4.3.2 Bedeutung 52

5 Datenerfassung 53
5.1 Web-Server-Protokolle 54
5.1.1 Common Log Format (CLF) 54
5.1.2 Extended Common Log Format (ECLF) 56
5.1.3 Proprietäre Log-Formate 57
5.1.4 Eigenschaften 57
5.1.5 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$ 58
5.1.6 Bedeutung 61
5.2 Redirector 63

5.2.1 Clientseitige Weiterleitung 63
5.2.2 HTTP-Weiterleitung 64
5.2.3 Quell-Ziel-Analyse mit Redirectoren 65
5.2.4 „Redirector“-Script für ECLF-Erzeugung 65
5.2.5 Eigenschaften 66
5.2.6 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$ 67
5.2.7 Bedeutung 67
5.3 Ping á la WhatWG 69
5.3.1 Benachrichtigung via Ping 69
5.3.2 Eigenschaften 69
5.3.3 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$ 70
5.4 Cookie-Tracking 71
5.4.1 Eigenschaften 71
5.4.2 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$ 71
5.4.3 Bedeutung 72
5.5 Snap-Shot-Technik 73
5.5.1 73
5.5.2 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$ 76
5.5.3 Bedeutung 78

6 Datennormalisierung **79**
6.1 Koppeln der Eingangsströme 80
6.2 Zusammenführen der Daten 81
6.2.1 Sammeln von CLF-Daten 82
6.2.2 Sammeln von ECLF-, Redirector und Ping-Daten . . . 85
6.2.3 Berücksichtigung der Cookie-Erweiterung 87
6.2.4 Sammeln von Snap-Shots 87

7 Vorberechnung und Abfrage **90**
7.1 Hits 94
7.1.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ 94
7.1.2 Abfrage des Werts der Tage $d_1 \ldots d_n$ für Bereich $\mathcal{O} \in \Omega$ 94
7.2 Verweilzeit 95
7.2.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ 95
7.2.2 Abfrage des Werts der Tage $d_1 \ldots d_n$ für Bereich $\mathcal{O} \in \Omega$ 95
7.3 Besucher 96
7.3.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ 96
7.3.2 Abfrage des Werts der Tage $d_1 \ldots d_n$ für Bereich $\mathcal{O} \in \Omega$ 100
7.4 Besuche 101
7.4.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ 101
7.4.2 Abfrage des Werts der Tage $d_1 \ldots d_n$ für Bereich $\mathcal{O} \in \Omega$ 103

7.5 Eingangs- und Zielpfade 104
7.5.1 Abfrage des Werts der Tage $d_1 \dots d_n$ für Bereich $\mathcal{O} \in \Omega$ 104
7.6 Pfadlänge, Session-Starter und Session-Killer 105
7.6.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ 105
7.6.2 Abfrage des Werts der Tage $d_1 \dots d_n$ für Bereich $\mathcal{O} \in \Omega$ 107
7.7 Kookurrenzen 108
7.7.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ 109
7.7.2 Abfrage des Werts der Tage $d_1 \dots d_n$ für Bereich $\mathcal{O}_1, \mathcal{O}_2 \in \Omega$ 110
7.8 Klickpfad 111
7.8.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ 111
7.8.2 Abfrage des Werts der Tage $d_1 \dots d_n$ 113

8 Abschließende Betrachtungen 114
8.1 Aspekte zur Benutzerführung 114
8.2 Authorisierter Zugriff 116
8.3 Stetiges Anwachsen des Datenvolumens 116
8.4 Fazit 117

Kapitel 1

Hintergrund

Das Internet ist erwachsen geworden. So oder so ähnlich könnte man die aktuelle Situation beschreiben.

Während vor einigen Jahren noch jede Firma darauf zielte, überhaupt eine Präsenz im Internet zu besitzen, um nicht den Anschluss an die Konkurrenz zu verlieren, erkennen heute immer mehr Unternehmen, dass es nicht ausreicht, einfach nur ein paar HTML-Seiten irgendwo abzulegen und den Dingen ihren Lauf zu lassen.

In zunehmendem Maße wird heute erkannt, dass Präsenzen bzw. Online-Anwendungen ständig wechselnden Anforderungen unterliegen. Dabei spielt insbesondere die Informationsüberflutung im Internet eine zentrale Rolle, denn durch das Überangebot wird es zunehmend schwerer, potentielle Interessenten auf gebotene Inhalte zu lenken.

Es wird außerdem verstärkt wahrgenommen, wie Konkurrenten sich die erweiterten Möglichkeiten im Internet zu Nutzen machen, um sich dadurch Wettbewerbsvorteile zu sichern.

Aber auch die Benutzeranforderungen im Internet wachsen stetig an: Während das Internet vor einigen Jahren als Ergänzung zu den „Gelben Seiten“ diente, übernimmt es mittlerweile die Aufgaben vollständiger Warenhäuser und organisiert zentrale Geschäftsbereiche. Besucher erwarten daher mittlerweile von Web-Seiten wesentlich mehr Funktionalität als dies vor ein paar Jahren der Fall war.

Für Unternehmen ergibt sich damit die Anforderung der stetigen Neuausrichtung an diesen wechselnden Ansprüchen. Anwendungsfunktionen müssen

permanent den Vorlieben und Erwartungen der Besucher genügen, dürfen diese aber auch nicht überfordern. Um die Richtung notwendiger Anpassungen zu erkennen und rechtzeitig auf Anforderungsänderungen zu reagieren, ist die stetige Kontrolle betroffener Anwendungen unumgänglich.

1.1 Web-Usage-Mining

Das Web-Usage-Mining versucht zu diesem Zwecke, Nutzungsanalysen von Internet-Präsenzen oder -Anwendungen zu liefern und damit auch Antworten auf zentrale Kontrollfragestellungen zu geben, wie beispielsweise:

1. Welche Inhalte werden überhaupt abgefragt?
2. Wie häufig und wie lange werden die Inhalte besucht?
3. Wer besucht die Präsenz?
4. An welchen Stellen verlassen Besucher ungewollt die Präsenz?
5. Welche Ziele verfolgen die Besucher?
6. Wie effektiv sind durchgeführte Marketing-Aktionen?

Im Zentrum steht hier die Untersuchung einer u.U. großen, heterogenen Besucherschaft möglicherweise weltweit agierender Internetnutzer. Das Informationsverhalten dieser Nutzer und damit auch die kognitive Ergonomie [31] (also die Frage nach Informationsaufnahmeverhalten) und die Interaktion mit unterschiedlichen Web-Entitäten soll dabei aufgedeckt werden. Ziel ist das Finden von Hinweisen zur Evaluation, Adaption und Optimierung von Web-Inhalten.

1.2 Schwachstellen existierender Systeme

Nutzungsanalysen im Internet bilden keinen neuen Forschungsbereich. Es gibt mittlerweile eine Fülle von Werkzeugen zur Analyse von Web-Log-Dateien. Dabei wird ausgenutzt, dass Web-Server ohne das Wissen ihrer Besucher (nicht-reaktiv) deren Web-Benutzung aufzeichnen. Diese Daten liefern Informationen über die Interaktionen der Besucher mit der Web-Site und geben

dadurch Eindrücke über die Nutzung. Dabei wird bewusst vernachlässigt, dass Web-Server-Protokolle einzig die Sicht des Web-Servers wiederspiegeln. Neben frühzeitig ausgewerteten Anfragen, die von zwischenliegenden Proxy-Caching-Servern ausgewertet wurden, erreichen die Informationen auch zum Teil verändert den Web-Server, sodass beispielsweise Benutzer nicht anhand ihrer IP-Adresse identifiziert werden können (vgl. Abschnitt 4.1 auf Seite 44).

Des Weiteren ist anzumerken, dass viele Werkzeuge direkt auf „rohen“ Log-Daten aufsetzen, ohne ein echtes „Data-Ware-House“ zu erzeugen. Dazu zählen unter anderem:

- WebTrends (http://www.webtrends.com/)
- AnaLog (http://www.analog.cx/)
- ClickTracks (http://www.clicktracks.at/)
- AWStats (http://www.awstats.net/)
- FastStats (http://mach5.com/products/analyzer/)
- u.a.m.

Diese Systeme werten die von Web-Servern gesammelten Zugriffsprotokolle aus und führen dabei eine vordefinierte Menge von Analysen aus, um insbesondere Extremstellen aufzudecken. Die Analyseergebnisse dokumentieren sie in statischen Berichten, die von Analysten interpretiert und beurteilt werden können.

Da die Grundlage der Analyse in diesen Fällen statisch generierte Berichte sind, ist es Aufgabe der genannten Werkzeuge die zentralen Informationen der Zugriffsprotokolle in einer angemessenen Granularität zusammenzustellen. Dabei versuchen die Werkzeuge, in der Regel mittels generischer Data-Mining-Techniken, wichtige Muster zu erkennen.

Die zum Einsatz kommenden Clustering-Verfahren besitzen eine Berechnungskomplexität, die pro erfassten Seitenzugriff von der Cluster-Kardinalität abhängt. Es ist daher für große Systeme nicht praktizierbar, die Cluster dynamisch zu aktualisieren, sodass jederzeit ein aktueller, vollständiger Bericht zur Verfügung steht. Für verzögerungssensitive Umgebungen sind zu große Verzögerungen jedoch andererseits nicht akzeptabel. [3]

Außerdem werden in der Regel Extremstellen bezogen auf den Gesamtdatenbestand ermittelt. Bei der Analyse von größeren Systemen ist es aber von

zentraler Bedeutung auch Teile eines Systems flexibel aufzubereiten, bzw. unabhängig vom Rest analysieren zu können. Die Systeme selbst abstrahieren im Wesentlichen von einer bereichsorientierten Herangehensweise. Die typische Vorgehensweise zur Analyse von Teilen besteht darin, vor der Berichterzeugung das jeweilige Zugriffsprotokoll zu filtern.

Dies führt dazu, dass zu jeder Filterung eine vollständige Neuberechnung des Berichts vorgenommen werden muss, deren Berechnungskomplexität von der Anzahl auszuwertender Daten abhängt und bei High-Traffic-Systemen zu hohen Laufzeiten führt. [3]

Aufgrund des hohen Zeitaufwands, der mit jeder Anpassung verbunden ist, ist es nicht praktikabel mehrere Anpassungen während einer Analyse vorzunehmen. Gerade für die Analyse komplexer Inhalte ist dies aber ein Muss. Schließlich kennen Analysten in der Regel im Vorfeld nicht die Ergebnisse ihrer Analyse und müssen sich diesen erst schrittweise annähern.

Eine Vorfilterung erfordert zudem spezielles technisches Know-How, sodass an der Analyse einer Präsenz häufig mehrere Akteure beteiligt werden. Systemadministratoren einer IT-Abteilung sind dann beispielsweise für das Vorfiltern und Erzeugen statischer Berichte zuständig, die von Analysten einer Marketing-Abteilung angefordert und interpretiert werden. Die Effektivität einer Analyse hängt damit auch vom Zusammenspiel der beteiligten Akteure ab.

Dieses Vorgehen ist zwar theoretisch vollständig, erfordert aber in der Praxis eine enge Kooperation der beteiligten Akteure. So müssen zur Vorfilterung häufig spezielle Umkonfigurationen an den Analysewerkzeugen vorgenommen werden, sodass die Analysten zur Analyse spezieller Bereiche auf die Unterstützung der jeweiligen Systemadministratoren angewiesen sind.

1.3 Ziele dieser Arbeit

Gegenstand der vorliegenden Arbeit ist die Konzeption eines Systems, welches Analysten den effizienten Zugang zu relevanten Informationen gewährt. Dabei setzt dieses System an der Stelle an, an der die meisten existierenden Systeme fundamentale Schwachstellen besitzen:

> Die angemessene Verarbeitung **präziser** Daten unter Berücksichtigung der **situativen Relevanz** und der **Komplexität riesiger Datenmengen**.

1.3.1 Präzision

Um ein höheres Maß an Datenpräzision zu erlangen, als dies mit der einfachen Protokollauswertung der Fall ist, versucht die Arbeit durch zusätzliche Kombination von speziellen Erweiterungen und alternativen Messformen eine Informationsgenauigkeit zu gewinnen, die die einzelnen Verfahren nicht bieten können. Sie stellt dazu drei grundlegend unterschiedliche Verfahren zur Datengewinnung vor und zeigt, wie die Daten in einen homogenen Datenstrom zusammengeführt werden können.

1.3.2 Situative Relevanz

Wie bereits erwähnt verwenden die meisten Anwendungen zum Usage-Mining Clustering-Algorithmen aus dem Data-Mining. Sie versuchen hiermit aus der Gesamtheit der Daten zentrale Aspekte herauszuziehen. Diese Aspekte beziehen sich allerdings auf die Gesamtheit aller Daten und nicht auf bestimmte Bereiche.

Für die Analyse komplexer Systeme halte ich es für notwendig, Analysten die effektive Abfrage von Aspekten aus speziellen Bereichen zu ermöglichen. Um dieses Ziel zu erreichen, möchte ich zentral ansetzen, dass Analysten sich ausgehend von einer grob-granularen Sicht zu einer fein-granularen Sicht „heran-zoomen“.

An dieser Stelle setze ich voraus, dass die geforderte Granularität im Wesentlichen über die beiden Dimensionen „Betrachtungszeitraum“ und „Betrachteter Inhaltsbereich“ festgelegt werden kann (vgl. Kapitel 7 ab Seite 91).

Mit Hilfe dieser Dimensionen kann der Analyst einerseits den Zeitraum festlegen, dessen Zugriffsdaten er betrachten möchte und andererseits einen mengenartigen inhaltlichen Bereich, der den zu analysierenden Teil einer Web-Site oder Web-Anwendung beschreibt. Die beiden Begrenzungen sind dabei frei wählbar und bewirken im Usage-Mining-System die Berechnung geeigneter Aggregationsdaten für die Analyse.

Ein wesentliches Ziel dieser Arbeit ist es demnach, eine Basis für die Entwicklung eines Usage-Mining-Systems zu bauen, welches es einem Analysten ermöglicht, sich nach dem „Zoom“-Prinzip schrittweise an zentrale Daten anzunähern, ohne dabei auf die Unterstützung anderer Personen (wie z.B. Systemadministratoren) angewiesen zu sein.

1.3.3 Komplexität riesiger Datenmengen

Das „Zoom“-Prinzip geht als zentraler Aspekt in die Gesamtkonzeption ein. Durch geeignete Vorberechnungen wird es möglich, aktuelle Informationen für große Zugriffsdatenmengen performant beliebig detailliert bzw. aggregiert abzufragen.

Um eine optimale Interaktivität bei der Analyse zu erhalten, stellt das Konzept statistische Werte vor, die in ihrer Komplexität unabhängig von der Anzahl analysierter Datensätze sind. Auf diese Weise wird für die Analyse von High-Traffic-Anwendungen ein hohes Interaktionsniveau erreicht.

Da das Heranzoomen an zentrale Aggregat-Informationen dann den wesentlichen Analyseprozess darstellt, entfällt zudem die in anderen Systemen herrschende Komplexität von Clustering-Algorithmen [40]. Stattdessen ist lediglich die Vorberechnung geeigneter Aggregatinformationen effizient umzusetzen. Dazu werden sämtliche Vorberechnungen inkrementell ausgeführt, sodass stetig aktuelle Informationen verfügbar sind.

Das Konzept stellt dabei sicher, dass auch zur inkrementellen Erweiterung der Statistiken nur kleine Ausschnitte der Gesamtdaten benötigt werden, sodass auch der Aktualisierungsprozess im laufenden Betrieb unabhängig von der Gesamtzahl erfasster Zugriffe ist.

Kapitel 2

Termini dieser Arbeit

Diese Arbeit verwendet eine Vielzahl von Termini bzw. Symbolen, um mathematische Sachverhalte auszudrücken. Im Folgenden seien diese Symbole kurz erklärt:

2.1 Domänen

$\mathbb{A}$ Menge aller Zugriffsprotokolleinträge im Sinne dieses Systems.

$\mathbb{C}$ Menge aller Protokolleinträge im CLF-Format. (vgl. hierzu Abschnitt 5.1.1 ab Seite 54)

$\mathbb{E}$ Menge aller Protokolleinträge im ECLF-Format. (vgl. hierzu Abschnitt 5.1.2 ab Seite 56)

$\mathbb{N}_0$ Menge der natürlichen Zahlen inklusive der Zahl Null.

$\mathbb{O}$ Menge aller Informationsobjekte, deren Nutzung das Usage-Mining-System überwacht. Informationsobjekte sind dabei eindeutig identifizierbare Elemente, die Bereichen zugeordnet sind. Im System selbst werde ich die Differenzierung von Objekten nicht vertiefen, da der Schwerpunkt der Arbeit in der Nutzungsanalyse und nicht im Dokumenten-Management besteht. Stattdessen setze ich voraus, dass es eine Funktion $objectOf : URL \rightarrow \mathbb{O}$ gibt, die

zu einer gegebenen URL eine zugehörige Objektidentität ermittelt und diese zurückgibt. Dabei ist es durchaus zulässig, dass diese Methode einfach die URL selbst als Identitätsmerkmal zurückgibt, wobei dies bedeuten würde, dass jedes Objekt nur an genau einer Stelle im Netz abrufbar ist.

$\mathbb{T}$ Die Menge aller Zeitpunkte, wobei hier sowohl Datumswerte als auch Zeitwerte gesehen werden. Im Späteren wird insbesondere eine Funktion $date : \mathbb{T} \rightarrow \mathbb{T}$ verwendet, die aus einem Zeitpunkt den jeweiligen Datumsteil extrahiert.

$\mathbb{U}$ Menge aller Benutzer. Benutzer des analysierten Systems besitzen hier eine Identität, die anhand verschiedenster Merkmale festgelegt wird (vgl. Kapitel 4 ab Seite 43). Die Menge $\mathbb{U}$ ist in dieser Arbeit zu verstehen als eine Menge über Benutzeridentitäten, die bijektiv auf die Menge aller das analysierte System nutzenden Benutzer abbildbar ist.

2.2 Konkrete Mengen

$\mathcal{A} \subseteq \mathbb{A}$ Ein zentraler Teil dieser Arbeit beschäftigt sich mit dem Erzeugen eines homogenen Zugriffsprotokolls. Dieses wird in der Arbeit mit dem Symbol $\mathcal{A}$ erwähnt und ist zu verstehen als eine konkrete Teilmenge aller denkbaren Zugriffsprotokolleinträge.

$\mathcal{O} \subseteq \mathbb{O}$ Das konzipierte System versucht einem Analysten die Möglichkeit zu bieten, ganze Bereiche zusammenhängend zu analysieren. Dabei werden in dieser Arbeit Bereiche als Mengen von Objekten verstanden. Das Symbol $\mathcal{O}$ entspricht demnach einem konkreten Bereich.

$\Omega \subseteq P(\mathbb{O})$ Das Analysesystem geht für Vorberechnungen (vgl. Kapitel 7 ab Seite 90) davon aus, dass im System eine konkrete Menge vordefinierter Bereiche existiert, die von häufiger Relevanz für Analysen sind. Diese Konkrete Bereichsmenge wird entsprechend mit einem Ω dargestellt.

$\mathcal{C} \subseteq \mathbb{C}$ Ein konkretes Zugriffsprotokoll im Common Log Format.

$\mathcal{E} \subseteq \mathbb{E}$ Ein konkretes Zugriffsprotokoll im Extended Common Log Format.

$\mathcal{S}$ In Kapitel 7 (ab Seite 90) werden Vorberechnungen für konkrete Messwerte (vgl. Kapitel 13) vorgenommen. Die Hauptstatistik für jeden Messwert wird dabei jeweils in einer speziellen Relation festgehalten, die in dieser Arbeit mit dem Symbol $\mathcal{S}$ adressiert werden.

$\mathcal{K}$ Neben der Hauptstatistik $\mathcal{S}$ gibt es für einige Messwerte eine Korrekturrelation, in der zusätzliche Informationen hinterlegt werden, um Messwerte für größere Zeiträume berechnen zu können. Diese Relationen werden jeweils mit dem Symbol $\mathcal{K}$ angesprochen.

2.3 Entitäten

$a \in \mathbb{A}$ Ein konkretes Zugriffsprotokollelement.

$o \in \mathbb{O}$ Ein konkretes Objekt.

$t \in \mathbb{T}$ Ein konkreter Zeitpunkt.

$u \in \mathbb{U}$ Ein konkreter Benutzer.

ϵ Dies ist das NULL-Element aller Domänen und wird in Situationen verwendet, in denen konkrete Werte nicht verfügbar sind.

2.4 Weitere Konstrukte

τ Dieser Wert ist eine Konstante, die Laufzeitverzögerungen beim Laden von Seiten und Aufbereiten von Inhalten im Web abdecken soll. Dieser Schwellenwert liegt dabei in der Größenordnung weniger Sekunden.

Γ_δ Das System nutzt an verschiedenen Stellen Caching-Strategien die Daten für einen Zeitraum δ vorhalten. Sofern nicht anders erwähnt umspannt δ dabei einen Zeitraum von einigen Tagen.

2.5 Anmerkungen zu $\mathcal{A} \subseteq \mathbb{A}$

$\mathcal{A} \subseteq \mathbb{A}$ bezeichnet in dieser Arbeit eine Relation, die ein ideales Zugriffsprotokoll abbildet. Ideal bedeutet an dieser Stelle, dass das Zugriffsprotokoll frei von Messfehlern ist und die Benutzer bidirektional auf reale Personen (bzw. elektronische Agenten) abbildbar sind. Die Tupel aus $\mathbb{A}$ bilden Zugriffe von Benutzern auf ein konkretes Objekt ab, wobei zu jedem Zugriff erfasst wird, von wann bis wann das jeweilige Objekt für den Benutzer dargestellt wurde.

Da im folgenden häufig die Komponenten der Tupel aus $\mathcal{A}$ benötigt werden, werden für die Komponenten folgende Bezeichner benutzt:

1. $a \in \mathbb{A} \Rightarrow a[user] \in \mathbb{U}$ Die Komponente "user" beschreibt den Benutzer, der den jeweiligen Zugriff durchgeführt hat.
2. $a \in \mathbb{A} \Rightarrow a[object] \in \mathbb{O}$ Die Komponente "object" beschreibt das Objekt, welches der Benutzer aufgerufen hat.
3. $a \in \mathbb{A} \Rightarrow a[access] \in \mathbb{T}$ Die Komponente "access" beschreibt den Zeitpunkt, an dem der Benutzer das Objekt aufgerufen hat. Dabei wird an dieser Stelle von Ladezeiten, Rendering-Zeiten usw. abstrahiert. Die Komponente entspricht somit dem Zeitpunkt der Benutzeraktion, die zum unmittelbaren Aufruf des Objekts geführt hat. „Unmittelbar" bedeutet hier, dass der Benutzer keine weiteren zwischenliegenden Aktionen zum Aufruf des Objekts getätigt hat.
4. $a \in \mathbb{A} \Rightarrow a[exit] \in \mathbb{T}$ Die Komponente "exit" beschreibt den Zeitpunkt, an dem der Benutzer zu einem anderen Objekt gewechselt oder die Präsenz/Anwendung verlassen hat. Wie auch bei der Komponente "access" wird hier der Zeitpunkt verstanden, an dem der Benutzer die entsprechend unmittelbare Aktion zum Verlassen des Objekts getätigt hat.

Neben diesen Komponenten werde ich an einigen Stellen den Protokolleinträgen zusätzliche Attribute verleihen, die allerdings nur temporär existieren und nur im jeweiligen Kontext von Bedeutung sind.

Beziehungen

In dieser Arbeit möchte ich den temporalen Fakt ausnutzen, dass dem Verlassen eines Objekts immer der Zugriff darauf vorangehen muss. Da für die Analyse Zugriffe unbedeutend sind, bei denen der Benutzer überhaupt keine Zeit investiert hat, können solche Zugriffe in Vorphasen eliminiert werden. Für ein beliebiges Zugriffsprotokoll $\mathbb{A}$ gilt daher in dieser Arbeit:

$$\forall a \in \mathbb{A} : a[access] < a[exit] \tag{2.1}$$

Von Parallelitätsaspekten beim Zugriff auf Inhalte möchte ich in dieser Arbeit weitestgehend abstrahieren und davon ausgehen, dass reale Personen nicht gleichzeitig mehrere unabhängige Aktionen durchführen und elektronische Agenten seriell arbeiten. Der hierdurch entstehende Fehler für die Beobachtung natürlicher Personen kann sicherlich vernachlässigt werden, da disjunkt-paralleles Handeln gegen die Natur des Menschen ist und Ausnahmen entsprechend selten sind. Für den Fehler in Verbindung mit elektronischen Agenten ist anzumerken, dass es nicht Gegenstand dieser Arbeit ist, das Verhalten synthetischer Werkzeuge zu studieren, sondern dass es darum geht Benutzerprofile realer Personen zu erlangen. Zweitgenannter Fehler ist daher ebenso vernachlässigbar. Da ich bei der Beschreibung der Messwerte von in sich stimmigen Zugriffsprotokollen ausgehe und somit annehmen kann, dass sich der Übergang von einem Objekt zu einem anderen Objekt im Zugriffsprotokoll durch ein gleichzeitiges Verlassen des erstgenannten und Aufrufen des zweitgenannten Objekts zeigt, kann eine Relation direkter Vorgänger $\prec$ bzw. direkter Nachfolger $\succ$ zwischen zwei Zugriffsprotokolleinträgen wie folgt definiert werden:

$$a_1 \prec a_2 \Leftrightarrow a_1 \neq a_2 \wedge a_1[user] = a_2[user] \wedge a_1[exit] = a_2[access] \tag{2.2}$$

$$a_1 \succ a_2 \Leftrightarrow a_1 \neq a_2 \wedge a_1[user] = a_2[user] \wedge a_1[access] = a_2[exit] \tag{2.3}$$

Als abkürzende Notation werde ich die transitive Vorgängerrelation mit $\overset{+}{\prec}$ und die transitive Nachfolgerrelation mit $\overset{+}{\succ}$ darstellen. Es gilt:

$$a_1 \overset{+}{\prec} a_2 \Leftrightarrow a_1 \prec a_2 \vee \exists a_3 : a_1 \overset{+}{\prec} a_3 \prec a_2 \tag{2.4}$$

$$a_1 \overset{+}{\succ} a_2 \Leftrightarrow a_1 \succ a_2 \vee \exists a_3 : a_1 \overset{+}{\succ} a_3 \succ a_2 \tag{2.5}$$

Aus der Vorgängerrelation lässt sich die folgende "Sitzungsanfangsrelation" $\overset{Start}{\prec}$ festlegen, die zu zwei Zugriffsprotokolleinträgen angibt, ob erster den Anfangseintrag in der zur zweiten gehörenden Sitzung darstellt. Die Relation ist wie folgt festgelegt:

$$a_1 \overset{Start}{\prec} a_2 \Leftrightarrow (\nexists a_3 : a_3 \prec a_1) \wedge ((a_1 = a_2) \vee (a_1 \overset{+}{\prec} a_2)) \tag{2.6}$$

Analog zur Sitzungsanfangsrelation lässt sich eine "Sitzungsenderelation" $\overset{Ende}{\succ}$ festlegen, die angibt, ob ein Zugriffsprotokolleintrag der letzte Eintrag einer zu einem anderen Zugriffsprotokolleintrag gehörenden Sitzung bildet. Die Relation ist wie folgt festgelegt:

$$a_1 \overset{Ende}{\succ} a_2 \Leftrightarrow (\nexists a_3 : a_3 \succ a_1) \wedge ((a_1 = a_2) \vee (a_1 \overset{+}{\succ} a_2)) \tag{2.7}$$

Es sei an dieser Stelle angemerkt, dass weder Vorgänger-, Nachfolger-, Sitzungsanfangs- noch Sitzungsenderelation eindeutig sind, denn aufgrund der Tatsache, dass die Protokoll-Komponenten "access" und "exit" den Zeitpunkt der Benutzeraktionen zum Aufruf bzw. Verlassen eines Objekts angeben, ist es durchaus möglich, dass beispielsweise durch ein und die gleiche Benutzeraktion ein HTML-Frameset mit mehreren HTML-Seiten aufgerufen wird.

Kapitel 3

Messwerte

Wesentliche Aufgabe des vorgestellten Systems ist es, verschiedenartige Messdaten über die Nutzung einer Web-Site bzw. Web-Anwendung zu liefern. Im vorliegenden Kapitel möchte ich diese Messwerte vorstellen und ihre Einsatzbereiche aufzeigen. Zentrale Aufgabe eines dieses Konzept implementierenden Systems ist es, diese Daten unter Verwendung geeigneter graphischer Aufbereitungen (Balkendiagramme, Tortendiagramme, Flächendiagramme u.a.m.) zu visualisieren.

Neben einer einleitenden kurzen Definition zu jedem Messwert führe ich an dieser Stelle eine naive Berechnung zu jedem Messwert auf, die mittels relationalem Tupelkalkül [41] und den in Abschnitt 2 aufgeführten Konzepten die logisch-mathematische Bedeutung des jeweiligen Messwerts zum Ausdruck bringt. Diese Berechnungen zeigen, dass jeder Messwert anhand einer einfachen Relation ermittelt werden kann. Diese Relation kann als ideales Zugriffsprotokoll verstanden werden und besitzt den folgenden Aufbau:

	AccessEntry	$a \in \mathbb{A}$
$\mathbb{U}$:	User	$a[user]$
$\mathbb{O}$:	Object	$a[object]$
$\mathbb{T}$:	AccessTime	$a[access]$
$\mathbb{T}$:	ExitTime	$a[exit]$

Abbildung 3.1: Zugriffsprotokoll-Eintrag im Basis-Protokoll

Jeder Zugriffseintrag $a \in \mathcal{A}$ eines Zugriffsprotokolls $\mathcal{A} \subseteq \mathbb{A}$ hält demnach für einen Benutzer $a[user]$ fest, dass dieser ein Objekt $a[object]$ der Anwendung im Zeitraum $a[access]$ bis $a[exit]$ aufgerufen/betrachtet hat. Da diese

Relation aus logisch-mathematischer Sicht zur Berechnung aller Messwerte ausreicht, wird in diesem Kapitel lediglich die Existenz eines derartigen konkreten Zugriffsprotokolls vorausgesetzt.

Bei der Berechnung des jeweiligen Messwerts wird an dieser Stelle vom Bezugszeitraum des Analysten abstrahiert, da für eine zeitraumbezogene Berechnung logisch gesehen für alle Messwerte einfach die Bezugsprotokollmenge $\mathcal{A}$ auf den jeweiligen Zeitraum $[t_1, t_2] \subseteq \mathbb{T}$ eingegrenzt werden muss. Ist daher $f : P(\mathbb{A}) \times P(\mathbb{O}) \Rightarrow \mathbb{N}_0$ eine Funktion zur Berechnung eines konkreten Messwerts, so ergibt sich der entsprechende Messwert bezogen auf den Zeitraum $[t_1, t_2]$ wie folgt:

$$f_{t_1,t_2}(\mathcal{A}, \mathcal{O}) = f(\{a \in \mathcal{A} | t_1 \leq a[access] \leq t_2\}, \mathcal{O}) \tag{3.1}$$

Auf die Berechnung folgt ein Abschnitt, der die Eigenschaften und Merkmale des jeweiligen Messwerts erörtert. Dabei werden technische Eigenschaften zum jeweiligen Messwert diskutiert. Es werden an dieser Stelle Hinweise auf die Ursachen von Messfehlern gegeben und algorithmische Probleme bei der Erfassung erläutert.

Abschließend wird zu jedem Messwert ein Überblick zur Interpretation gegeben. Dabei wird die Bedeutung des alleinstehenden Messwerts wiedergegeben und auf seine betriebswirtschaftliche Relevanz eingegangen. Außerdem wird gezeigt, wie dieser Messwert mit anderen Messwerten kombiniert werden kann, um weitere Informationen zu gewinnen.

3.1 Hits

Der Messwert „Hits“ gibt die Anzahl der Zugriffe auf ein Objekt in einem gegebenen Zeitintervall an. Im Internet wird an dieser Stelle meist die Anzahl der Aufrufe einzelner Seiten, Bilder oder sonstiger Dateien verstanden. Die meisten Messtechniken erfassen dabei zu jedem Objektaufruf den entsprechenden Aufrufzeitpunkt. Im Nachhinein kann dann für einen beliebigen Zeitraum gezählt werden, wie viele Aufrufe ein konkretes Objekt in diesem Zeitraum besaß. [40]

Berechnung

$$hits(\mathcal{A}, \mathcal{O}) = |\{a \in \mathcal{A} | a[object] \in \mathcal{O}\}| \tag{3.2}$$

3.1.1 Eigenschaften

Der „Hits“-Messwert ist wohl einer der am weitesten verbreiteten Messwerte im Bereich der Web-Seiten-Analyse. Dies kommt vor allem daher, dass die meisten Web-Server geeignete Protokolle zur Verfügung stellen, aus denen sich die Information leicht entnehmen lässt. Allerdings ist zu berücksichtigen, dass dieser Messwert oft verfälscht ist, da diverse Caching-Strukturen im Netz vorzeitig Benutzeraufrufe auswerten und so der eigentliche Quellserver wiederholte Aufrufe oft nicht mitbekommt.

Dennoch bietet der Messwert einen grundlegenden Einblick in die technische Nutzungslast von Objekten und ermöglicht dadurch das Aufspüren potentieller Flaschenhälse. So zeigt ein hoher „Hits“-Wert an, dass ein Objekt sehr häufig aufgerufen wird und somit die gegebene Infrastruktur entweder sehr zielgerichtet auf dieses Objekt hinleitet oder die Benutzer selbst aufgrund ihrer bisherigen Erkenntnisse zielgerichtet dieses Objekt aufsuchen. Auf diese Weise wird erkennbar, an welchen Stellen die höchste Belastung entsteht und wo mögliche Übertragungs- und Verarbeitungsengpässe entstehen. Im Falle einer Systemüberbelastung sollte im technischen Sinne geprüft werden, welche Objekte sehr häufig aufgerufen werden, um diese mit geeigneten algorithmischen Verbesserungen oder Systemseitigen Caching-Verfahren auszustatten.

3.1.2 Bedeutung

Während das gleichförmige Anwachsen oder Fallen des „Hits“-Werts meist ein entsprechendes Ausdehnen bzw. Schrumpfen des Bekanntheitsgrades für ein Objekt wiederspiegelt, sind sprunghafte Änderungen meist auf den Erfolg von gezielten Marketing-Aktionen oder auch Störungen zurückzuführen. Eine Überwachung der „Hits“ in einem gewissen Zeitraum birgt daher zentrale Informationen über die Effektivität von Marketing-Aktionen.

Ein hoher Wert ist hier jedoch nicht immer nur im “guten“ Sinne zu verstehen. In Extremfällen ist vielmehr zu hinterfragen, ob nicht irgendwelche Fehlsituationen diesen Wert bewirken. Ist der Wert beispielsweise unerwartet hoch, so kann dies auch ein Anzeichen dafür sein, dass Benutzer nicht willentlich das Objekt aufrufen, sondern vielmehr festhängen, sich im Kreise drehen und gezwungen sind, ein Objekt immer wieder aufzurufen.

Für den Fall, dass der Wert einem Null-Wert entspricht, kann mit Sicherheit gesagt werden, dass das Objekt überhaupt nicht genutzt wurde, denn schließlich wurde es nicht aufgerufen. Für alle Werte ungleich Null lässt eine alleinstehende Betrachtung nur Spekulationen über mögliche betriebswirtschaftliche Auswirkungen zu. So kann zum Beispiel nicht erkannt werden, ob es immer die gleichen Benutzer sind, die auf ein Objekt zugreifen oder ob ein Objekt möglicherweise sehr häufig von irgendwelchen Internet-Robots abgefragt wird, aber es nie von realen Benutzern erreicht wird.

In Verbindung mit dem Messwert „Verweilzeit“ (s. Abschnitt 3.2) lässt sich erkennen, ob Benutzer die gebotenen Inhalte wirklich aufnehmen oder die Objektausgabe eher ignorieren. So gibt der Quotient aus Verweilzeit und Hits an, wie lange die Benutzer sich im Mittel pro Aufruf mit einer Objektausgabe auseinandergesetzt haben. Ist der Wert extrem klein, so kann darauf geschlossen werden, dass die Seite nicht wirklich gelesen wird. Ist sie hingegen erwartungsgemäß groß, so drückt dies aus, dass die Seite entsprechend den Erwartungen betrachtet wird. Dabei muss allerdings berücksichtigt werden, dass die Verweilzeit in der Regel nur indirekt gemessen werden kann, sodass der Messwert sehr starken Messungenauigkeiten unterliegt (s. Abschnitt 3.2).

In Kombination mit dem Messwert „Besucher“ (s. Abschnitt 3.3) kann durch einfache Quotientenbildung ermittelt werden, wie häufig ein Objekt im Mittel pro Benutzer aufgerufen wurde. Dieser Mittelwert spiegelt die Rückkehr-Rate von Benutzern zum aktuellen Objekt wider. In der Regel kann davon ausgegangen werden, dass Seiten, die von Benutzern als hochwertig und wertvoll erachtet werden, häufiger wiederholt aufgerufen werden, als Seiten, denen sie

nur geringen Wert beimessen. Natürlich muss dabei berücksichtigt werden, dass Benutzer möglicherweise ungewollt auf eine Seite geleitet werden. In diesen Fällen kann davon ausgegangen werden, dass sie versuchen die Seiten möglichst schnell wieder zu verlassen, was sich in kurzen Verweilzeiten niederschlägt. Bei einer Auswertung sollten somit die Werte „Hits“, „Verweilzeit“ und „Besucher“ in Kombination betrachtet werden.

Auch in Verbindung mit dem Messwert „Besuche“ lassen sich interessante Informationen ableiten. Wird die Seite pro Besuch mehrfach aufgerufen, so beutet dies, dass Benutzer entweder immer wieder zum Objekt zurückgeführt werden ober dass sie das aktuelle Objekt immer wieder als Ausgangspunkt für unterschiedliche Aktionen nutzen. Wird dieses Verhältnis sehr groß, so ist dies ein Anzeichen dafür, dass die Navigationsstruktur einer Präsenz nicht den Benutzerbedürfnissen entspricht.

Es lassen sich aber auch Hinweise auf betriebswirtschaftlich interessantere Fragen ableiten. So kann anhand des Messwerts die prinzipielle Erreichbarkeit einer Seite erkannt werden. Ein kleiner Wert deutet eine schlechte Erreichbarkeit oder Auffindbarkeit für ein Objekt an. Strebt der Wert selbst für einen großen Betrachtungszeitraum gegen Null, so ist dies ein direkter Hinweis darauf, dass das Objekt überhaupt nicht im Web zur Verfügung steht.

Der Hits-Messwert spiegelt die grundlegende Nutzung von Objekten wieder. Sofern der Messwert gegen Null strebt, ist dies ein klares Anzeichen dafür, dass ein Objekt aufgrund der gegebenen Infrastruktur nicht erreichbar ist. Ein hoher Wert besitzt alleine betrachtet keinen großen Aussagegehalt, da hier die Interpretationsmöglichkeiten sehr weit gefächert sind. In Verbindung mit den Messwerten „Verweilzeit“, „Besucher“ und „Besuche“ lassen sich jedoch äußerst nützliche Informationen herleiten.

Eine Variante des „Hits“-Messwerts bildet der Messwert “Page Impression“, der nicht die Zugriffe einzelner technischer Dateien misst, sondern den Zugriff auf zusammenhängende Objekte. Bilder und andere Dateien, die Bestandteil spezieller Seiten sind, werden dabei nicht mehr alleinstehend betrachtet, sondern immer im Kontext der jeweiligen Seite, sodass die Anzahl der Seitenkonstellationen, die Besucher zu Gesicht bekamen, gezählt werden und nicht die Zugriffe einzelner Dateien. Vorteilhaft an diesem Messwert ist, dass die Anzahl der resultierenden Seiten beurteilt werden und nicht die Zugriffe auf elementare Seitenbestandteile. [31] Da diese Variante des Messwerts durch eine geeignete Bereichswahl anhand des „hits“-Werts ermittelt werden kann und somit implizit gegeben ist, wird er hier nicht genauer betrachtet.

3.2 Verweilzeit

Unter Verweilzeit versteht man die Zeit, die Benutzer mit der Nutzung oder Auswertung einer Ausgabe verbringen. In der Regel kann dieser Wert nicht direkt ermittelt werden, da ein Programm kaum entscheiden kann, ob ein Benutzer gerade eine Ausgabe liest oder diese nur im Hintergrund angezeigt wird. Näherungsweise kann man jedoch davon ausgehen, dass die Verweilzeit der Zeit entspricht, die zwischen dem Aufrufzeitpunkt und dem Verlassenszeitpunkt eines Objekts vergeht, wobei Ausreißerwerte ignoriert werden müssen.

Berechnung

$$dwell(\mathcal{A}, \mathcal{O}) = \sum^{a \in \mathcal{A} | a[object] \in \mathcal{O}} a[exit] - a[access] \tag{3.3}$$

3.2.1 Eigenschaften

Wie bereits erwähnt, kann eine präzise Nutzungszeit mit den, in der Web-Technologie zur Verfügung stehenden Mitteln, nicht erfasst werden. So ist es zum Beispiel unmöglich zu unterscheiden, ob ein Benutzer seit Stunden den Inhalt einer Web-Seite durchliest oder ob er mittlerweile in die Mittagspause verschwunden ist.

Dennoch lassen sich in einigen Fällen gute Näherungswerte aus anderen Daten herleiten, indem die Zeitdifferenz zwischen zwei Objektaufrufen als „Verweilzeit auf dem ersten Objekt“ interpretiert wird. Bei einfach strukturierten Web-Sites führt diese Vorgehensweise meist zu sehr guten Ergebnissen. Allerdings ist es schwierig die Verweilzeit an den Endpunkten zu messen, da hier keine Nachfolgeraufrufe erfasst wurden und somit auch keine Werte für das Verweilende zur Verfügung stehen.

Eine alternative Möglichkeit zum Erfassen von Verweilzeiten bietet die SnapShot-Technik (siehe Abschnitt 5.5 auf Seite 73). Dabei werden periodisch Informationen vom Client zum Server geschickt, sodass die andauernde Sichtbarkeit von Inhalten erfasst werden kann. Allerdings sei an dieser Stelle angemerkt, dass auch hier Messungenauigkeiten entstehen, wobei die Messungenauigkeit von der gewählten Periode zur Übertragung von Snapshot-Informationen abhängt.

Die Verweilzeit drückt in der Regel aus, wie lange ein Objekt im Client dargestellt wird wodurch auch die potentielle Belastung des Clients sichtbar wird.

3.2.2 Bedeutung

Die Zeit, die ein Benutzer mit einer Objektausgabe verbringt, spiegelt im Idealfall wieder wie lange sich der Benutzer mit dieser Ausgabe auseinandergesetzt hat.

Verbringen die Benutzer lange Zeit mit dem Betrachten einer konkreten Ausgabe, kann dies verschiedene Ursachen haben. So ist es möglich, dass der gebotene Inhalt die Benutzer anspricht und diese beispielsweise die Anzeige eines Photos genießen. Da an dieser Stelle ein Aggregat (z.B. die Summe) aller Verweilzeiten betrachtet wird, kann jedoch anhand der alleinstehenden Betrachtung des Werts nicht erkannt werden, ob wenige Benutzer viel Zeit oder sehr viele Benutzer kaum Zeit mit einem Objekt verbringen.

Eine sehr lange Gesamtverweilzeit kann aber auch bedeuten, dass der dargestellte Inhalt sehr umfangreich ist und ein Durcharbeiten viel Zeit benötigt oder dass der Inhalt zwar nicht mengenmäßig umfangreich aber dafür entsprechend komplex und schwer aufnehmbar ist, sodass Nutzer gezwungen sind, sich intensiv mit dem Inhalt auseinanderzusetzen. Außerdem ist denkbar, dass die Benutzer sich nicht zurechtfinden und aus der Unwissenheit heraus, was als nächstes zu tun ist, verharren.

Eine kurze Gesamtverweilzeit kann wiederum bedeuten, dass der gebotene Inhalt die Benutzer nur wenig oder überhaupt nicht interessiert beispielsweise, dass der gebotene Inhalt so zielgerichtet aufgebaut ist, dass der Benutzer auf Anhieb weiß, was er als nächstes tun muss, um ein von ihm selbst gestecktes Ziel zu erreichen.

In Verbindung mit den Messwerten „Hits“, „Besucher“ sowie „Besuche“ lassen sich durchschnittliche Verweilzeiten ermitteln, die Rückschlüsse auf die wahrgenommene Komplexität einer Seite oder die Benutzerkonfidenz zulassen.

Wenngleich Levene u.a. [28] zeigen, dass Verweilzeiten einer Zipf-Verteilung unterliegen und daher irreführend sind wenn lange Zugriffe die Relevanz kürzerer verdecken, drücken lange durchschnittliche Verweilzeiten dennoch aus, dass Benutzer die Bearbeitung eines Inhaltes als entsprechend komplex wahrnehmen.

Bei der Auswertung müssen immer die technisch bedingten Messungenauigkeiten berücksichtigt werden. Nicht selten rufen Benutzer eine Web-Site auf und schalten dann zu einer anderen Applikation um, ohne dass das System erkennen kann, dass der Benutzer die Seite nicht mehr nutzt. Verhältnismäßig große Messwerte verlieren somit an Bedeutung.

Setzt man jedoch voraus, dass solche Extremwerte bei der Erfassung herausgefiltert werden und geht man zudem von einem kausalen Benutzerverhalten aus, so kann die durchschnittliche Verweilzeit bedeutend zur Beurteilung der allgemeinen Benutzerkonfidenz beitragen.

So ist davon auszugehen, dass Benutzer nur dann einen Inhalt lange betrachten, wenn ein direktes Interesse an diesem speziellen Inhalt besteht. Dabei gilt es zu unterscheiden, ob der Inhalt selbst dieses Interesse abdeckt oder ob sich die Benutzer bereits von der Bearbeitung einen Mehrwert für ein entferntes Ziel erhoffen. Für den zuletzt genannten Fall kann anhand der Verweilzeit nicht entschieden werden, ob der Inhalt für den Benutzer nützlich oder unnütz ist. Nichts desto trotz kann man jedoch davon ausgehen, dass Benutzer die Bearbeitung irrelevanter Inhalte vorzeitig abbrechen, was sich in kurzen Verweilzeiten niederschlägt.

Aus Marketing-Sicht, bzw. zur optimalen Platzierung von Werbebannern, ist es interessant zu erfahren, wo die Benutzer am meisten Zeit verbringen, denn je länger ein Besucher eine Seite betrachtet, um so höher ist die Wahrscheinlichkeit, dass er einen hier platzierten Werbebanner wahrnimmt. Daher ist das Verhältnis aus „Verweilzeit“ und „Besucher“ von besonderer Bedeutung.

Allgemein kann man sagen, dass die Verweilzeit ein vereinigendes Merkmal für die Identifikation wahrgenommener Komplexität und Interessensabdeckung ist.

3.3 Besucher

Der Messwert Besucher gibt die Anzahl der unterschiedlichen Personen an, die ein Objekt in einem gegebenen Zeitraum aufgerufen haben. Auf diese Weise lässt sich die Bekanntheit eines Objekts messen: Je mehr Benutzer ein Objekt aufrufen, um so größer ist seine Verbreitung.

Berechnung

$$visitors(\mathcal{A}, \mathcal{O}) = |\{u \in \mathbb{U} | \exists a \in \mathcal{A} : a[user] = u \wedge a[object] \in \mathcal{O}\}| \quad (3.4)$$

3.3.1 Eigenschaften

Beim Identifizieren der Benutzer muss berücksichtigt werden, dass eine eindeutige Zuordnung von Personen im Internet keine triviale Aufgabe ist. (Die hier entstehenden Probleme werden in Kapitel 4 ab Seite 43 näher betrachtet.) Auch muss berücksichtigt werden, dass der Messwert, je nach eingesetzter Messtechnik, unterschiedlicher Präzision unterliegt. Beispielsweise können einem Benutzer mehrere Cookies zugeordnet werden, wenn er diese zwischenzeitig zurücksetzt oder Cookies mehrfach genutzt werden weil unterschiedliche Benutzer die gleiche Software nutzen.

Aus Gründen der Speicher- und Berechnungskomplexität bietet es sich an, Benutzerdaten in verdichteter Form vorzuhalten. In einem eigenen Projekt hat es sich als günstig erwiesen, Abfragen für größere Zeiträume (Wochen, Monate oder Jahre) auf Tagesgenauigkeiten zu beschränken. Diese Verfahren werde ich in Abschnitt 7.3 ab Seite 96 genauer betrachten.

3.3.2 Bedeutung

Aus technischer Sicht ist zwar in der Regel der Messwert „Hits“ (auf Seite 15) von höherer Bedeutung als die Besucherzahl, da er die reale Infrastrukturbelastung wiederspiegelt, jedoch lassen sich auch aus der Besucherzahl Hinweise zu technischen Optimierungschancen ableiten. So ist beispielsweise der Einsatz von DNS basiertem Load-Balancing genau dann sinnvoll, wenn eine sehr hohe Besucherzahl auf die verfügbaren Objekte zugreift. Ist die Besucherzahl im Vergleich zum „Hits“-Wert klein, so bedeutet dies, dass die gleichen Besucher immer wieder die selben Inhalte abfragen. In diesem Falle

kann der Einsatz von Client-seitigem Caching die Infrastrukturbelastung verbessern. Erreicht die Besucherzahl nahezu den Wert der „Hits“, so bedeutet dies, dass die Objekte von sehr vielen unterschiedlichen Besuchern aufgerufen werden. Entsprechend liefert hier eine client-seitige Caching-Strategie keinen Nutzen und nur ein Proxy oder ein anderes serverseitiges Caching kann einen höheren Datendurchsatz für den Endnutzer erreichen.

Der Messwert spiegelt zudem die Größe der Personengruppe wieder, die ein Objekt genutzt hat. Für ein typisches Werbemittel kann man sagen, dass dieser Wert überhaupt nicht hoch genug sein kann. Je größer der Wert, um so mehr Personen werden erreicht. Sofern der Wert über einem Durchschnittswert liegt, drückt dies aus, dass eine Konzentration der internen oder externen Infrastruktur auf dieses Objekt vorliegt, da vergleichsweise viele Benutzer auf dieses Objekt geleitet werden. Stellt man im Laufe der Zeit sogar ein Anwachsen des Werts fest, so bedeutet dies, dass der Bekanntheitsgrad des Objekts zunimmt und das Objekt eine dem Zuwachs entsprechende Erreichbarkeit besitzt.

Je kleiner der Wert ist, um so weniger Besucher erreichen dieses Objekt. Das bedeutet, dass die Infrastruktur nicht hinreichend zu diesem Objekt leitet und das Objekt zudem vergleichsweise unbekannt ist. Wird im Laufe der Zeit ein fallender Wert diagnostiziert, so sind entweder bestimmte Pfade zu diesem Objekt nicht mehr existent, bzw. in Vergessenheit geraten oder die Besuchergruppe empfindet das Objekt selbst als nicht hinreichend nützlich für existierende Situationen und kennt bessere Alternativen.

Vergleicht man die Anzahl der Besucher mit dem Messwert Vorgänger bzw. Eingangspfad (siehe Seite 26), so kann auf diese Weise ein Eindruck über die Nutzung der Wege zum aktuellen Objekt gewonnen werden. Ist die Besucherzahl vergleichsweise groß zur Anzahl der Eingangswege, so benutzen die Besucher überwiegend die gleichen Wege. Ist die Besucherzahl hingegen vergleichsweise klein, so gelangen sie über unterschiedliche Wege zum aktuellen Objekt. Im Extremfall geht sogar jeder Benutzer seinen eigenen Weg. Eine derartige Erkenntnis kann entscheidend für die Auswahl von Werbe-Bannern sein, da hiermit die Effektivität von Links gemessen werden kann.

Ein vergleichender Blick auf die Nachfolger- bzw. Zielpfade (siehe Seite 29) lässt analog Rückschlüsse auf die Bedeutung des aktuellen Objekts als “Wegbereiter“ zu anderen Objekten zu.

Wie auch für die „Hits“ erlaubt es dieser Wert, direkte Informationen über die Objekterreichbarkeit abzuleiten: Wächst die Besucherzahl im Laufe der Zeit stetig an, kann abgeleitet werden, dass die Objekte recherchierbar und

gut erreichbar sind. Stagniert das Anwachsen des Werts, so bedeutet dies, dass keine weiteren Benutzer das Objekt finden und lediglich Besucher, die ein Objekt bereits kennen, dieses erneut finden. Erreicht der Wert für ein gegebenes Zeitintervall den Wert Null, so wurde das Objekt in diesem Zeitraum nicht besucht.

Der Wert spiegelt so auf direkte Weise die Größe des betrachteten Klientels wieder: Je größer der Wert, um so größer ist die Menge der Personen, die dieses Objekt aufgerufen haben.

Die Besucherzahl gibt keinerlei direkte Informationen über die Besucherkonfidenz. Dennoch kann in Verbindung mit dem Messwert „Hits“ (siehe Seite 15) die wiederholte Nutzung von Objekten durch Besucher erkannt werden.

Die wiederholte Nutzung eines Objekts lässt zudem auf eine entsprechende Benutzerzufriedenheit schließen. Wird ein Objekt von einer großen Besucherzahl häufig wieder verwendet, so kann davon ausgegangen werden, dass es vielfältig einsetzbar ist. Wird es von einer großen Besucherzahl selten wiederverwendet, so drückt dies aus, dass es spezielle Situationen gibt, die die Nutzung dieses Objekts erfordern. Ist die Anzahl wiederkehrender Besucher im Laufe der Zeit rückläufig, so impliziert dies, dass entweder die entsprechenden Nutzungssituationen selbst immer weniger Bedeutung erlangen oder den Benutzern bessere Alternativen zur Verfügung stehen.

3.4 Besuche

Ein häufig genannter Indikator für die Intensität der Benutzung einer Web-Site und somit der Popularität und dem Erfolg der Site, ist die Anzahl der Web-Sitebesuche /in der Literatur auch "visit" oder "session" genannt). Der Messwert spiegelt die Häufigkeit der zusammenhängenden Aktionen bezogen auf einen größeren Dokumentenraum wieder.

Berechnung

$$\begin{aligned} visits(\mathcal{A},\mathcal{O}) = \; & |\{(u,t) \in \mathbb{U} \times \mathbb{T} | (\exists a_1 \in \mathcal{A} : a_1 \overset{Start}{\prec} a_1 \wedge a_1[user] = u \wedge \\ & a_1[access] = t \wedge a_1[object] \in \mathcal{O}) \\ & \wedge (\nexists a_2, a_3 \in \mathcal{A} : a_2 \overset{Start}{\prec} a_3 \wedge a_1 \overset{Start}{\prec} a_3 \\ & \wedge a_1[access] > a_2[access])\}| \end{aligned} \tag{3.5}$$

3.4.1 Eigenschaften

Besuche lassen sich i.d.R. leicht ermitteln. Über sogenannte „Session" Cookies lassen sich Sitzungskennungen umherreichen, die auf dem Server protokolliert werden können. Um im Verlauf der Zeit die Anzahl der Besuche ermitteln zu können, braucht lediglich die Anzahl unterschiedlicher Sitzungskennungen gezählt werden. Da Besuche in der Regel nur relativ kurze Zeiträume in Anspruch nehmen, ist eine triviale Verdichtung der Daten ohne größeren Informationsverlust möglich, sodass beispielsweise Tag für Tag gezählt werden kann, wie viele Sitzungen an dem jeweiligen Tag das Objekt berücksichtigten. Die Anzahl der Besuche in einem größeren Zeitraum (z.B. Monate oder Jahre) kann dann durch die Summe der Besuche der jeweiligen Tage berechnet werden.

Ein Web-Sitebesuch setzt sich aus Transaktionen einzelner Besucher zusammen. Problematisch bei der Bewertung von Besuchsmessungen ist die Tatsache, dass bei der Analyse zeitliche Festlegungen getroffen werden müssen. Zum Beispiel wann beginnt ein Web-Sitebesuch und wann endet er. Im Gegensatz zu kommerziellen Datenbankrecherchen, bei denen sich der Recherchierende am Host an- und abmeldet, gibt es bei öffentlichen Web-Sites in der Regel kein "Logoff" von Web-Sitesitzungen. Das bedeutet, dass bei Web-Analysen für die Bestimmung der Besuche ein spezieller Timeout gesetzt

werden muss. Die Untersuchung von Oldenburg zeigt dabei, dass die Anzahl der Besuche durch die Länge des gewählten Timeouts bestimmt wird [34].

3.4.2 Bedeutung

Dieser Messwert ist in vielerlei Hinsicht verwandt mit dem Messwert Besucher (siehe Seite 21), da er ebenfalls einen benutzerbezogenen Blick auf die Objektnutzung wiederspiegelt. Je nach eingesetztem Verfahren zur Benutzeridentifikation sind die beiden Werte bedingt durch die entstehenden Messfehler sogar identisch. In der idealisierten Welt wendet dieser Wert jedoch einen anderen Blickwinkel auf die Nutzung an. Es wird hier davon ausgegangen, dass ein Benutzer, der innerhalb ein und derselben Sitzung ein Objekt mehrfach besucht, dieses nicht unbedingt inhaltlich mehrfach durcharbeitet, sondern lediglich eine unterbrochene Arbeitshandlung fortsetzt oder bereits ermitteltes Wissen anwendet. Entsprechend spiegelt der Messwert wieder, in wieviel unterschiedlichen Situationen das Objekt benutzt wurde.

Eine hohe Besuchszahl drückt somit aus, dass das Objekt in vielen unterschiedlichen Sitzungen aufgerufen wurde. Sofern es Benutzer bewusst aufgerufen haben, spiegelt dieser Häufigkeitswert wieder, dass die Benutzer dem Objekt eine hohe Bedeutung beimessen und es in vielen Situationen als geeignetes Hilfsmittel ansehen.

Betrachtet man den Quotienten aus dem Messwert und dem Messwert Besucher (siehe Seite 21), so erhält man eine durchschnittliche Rückkehrrate der Besucher. Die gewonnene Information drückt aus, ob die Besucher eher Stammkunden oder Einmalbesucher sind. Stammkunden kennzeichnet wiederum eine gewisse Zufriedenheit, sodass dieses Verhältnis implizite Aussagen über die Benutzerkonfidenz ermöglicht.

3.5 Vorgänger und Eingangspfade

In den meisten Fällen rufen Benutzer Objekte über Querverweise aus anderen Objekten auf. Die Messwerte "Vorgänger" bzw. "Eingangspfade" spiegeln die Anzahl der Objekte wieder, von denen ausgehend ein Benutzer direkt bzw. indirekt zu einem konkreten Objekt gelangt ist.

Berechnung

$$\begin{aligned} predecessors(\mathcal{A}, \mathcal{O}) &= |\{(o) \in \mathbb{O} | \exists a_1, a_2 \in \mathcal{A} : a_1[object] = o \\ &\quad \wedge a_2[object] \in \mathcal{O} \wedge a_1 \prec a_2\}| \\ inpaths(\mathcal{A}, \mathcal{O}) &= |\{(o) \in \mathbb{O} | \exists a_1, a_2 \in \mathcal{A} : a_1[object] = o \\ &\quad \wedge a_2[object] \in \mathcal{O} \wedge a_1 \overset{+}{\prec} a_2\}| \end{aligned} \tag{3.6}$$

3.5.1 Eigenschaften

Bei der Erfassung von Vorgängerobjekten gilt es vor allem die technische Schwierigkeit zu meistern, ein und dasselbe Objekt intelligent wiederzuerkennen. Häufig wird an dieser Stelle die URL des Vorgängerobjekts als Identifikationsmerkmal genutzt. Während man jedoch für eigene Objekte leicht aus der URL geeignete Objektidentifikationsmerkmale entnehmen kann, ist dies für beliebig fremdverwaltete URLs eine nahezu unmögliche Aufgabe, zumal URLs nicht ausschließlich Positionsbeschreibungen beinhalten, sondern häufig auch Session-Kennungen, Layout-Optionen u.a.m.

In einer praktischen Implementierung bietet es sich daher häufig an, dedizierte Präfixe der URL getrennt zu bewerten (z.B. Domäne, Domäne + Port, Domäne + Port + Pfad, ...)

3.5.2 Bedeutung

Die Anzahl der Vorgänger drückt aus, in welchem Maß ein Objekt von anderen Objekten referenziert wird, wobei nur solche Verweise gezählt werden, die mindestens einmal von einem Benutzer aufgerufen wurden. Stellt man diesem Messwert einen typischen Ranking-Mechanismus von Internet-Suchmaschinen gegenüber, bei dem die Qualität von Inhalten durch die Anzahl existierender Verweise auf dieser Seite beurteilt wird, so fällt auf, dass

die beiden Werte eine hohe Ähnlichkeit besitzen. Für den Fall, dass die Besucher im Laufe der Zeit alle existierenden Querverweise gewählt haben, sind die Werte sogar identisch. Für den Fall, dass Besucher nur eine Teilmenge der tatsächlich existierenden Querverweise gewählt haben gilt, dass andere Querverweise nur wenig Bedeutung besitzen, da ihre schwere Auffindbarkeit auch einen geringen Verknüpfungswert impliziert. Tatsächlich versuchen Suchmaschinen sogar diesen Verknüpfungswert heuristisch zu prognostizieren, um nicht zu sensibel auf Fehlinformationen zu reagieren.

Die Eingangspfadvariante spiegelt die Anzahl der Situationen wieder, in denen Benutzer das aktuelle Objekt als hilfreich erachtet haben. Der Wert gibt somit einen direkten Hinweis auf die wahrgenommene Vielfalt des Objekts aus Benutzersicht. Hat eine Seite einen im Laufe der Zeit kontinuierlich hohen Eingangspfadwert ist anzunehmen, dass intensiv genutzte Web-Seiten auf diese Seite zeigen. Die Verteilung der Nutzungszahlen kann in diesem Fall auf eine bestimmte Art von Webseite hindeuten z.B. Authority-Seiten (vgl. dazu das linkbasierte Konzept hubs & authorities [26]). Beinhaltet eine häufig referenzierte Seite selbst viele Outlinks (Links, die auf eine externe Web-Ressource verweisen), wird die Seite eventuell als "hub page" nach Kleinberg [26] genutzt.

Ist die Anzahl unterschiedlicher Vorgängerobjekte sehr hoch, so drückt dies eine Konzentration der existierenden Infrastruktur auf das aktuelle Objekt aus. Das Objekt wird somit an vielen Stellen als hochwertig verstanden und weiterempfohlen, wobei die Benutzer diese Empfehlungen aufnehmen und verfolgen. Hat eine Seite eine hohe Anzahl direkter Vorgänger dann liegt nahe, dass die Seite einen hohen Bekanntheits- und Etablierungsgrad im Web oder einer Community hat.

Analog drückt eine geringe Vorgängerzahl eine schwere Erreichbarkeit aus. Sofern es kaum Vorgängerobjekte gibt heißt dies sogar, dass das Objekt nur von einer sehr speziellen Kundengruppe durch Direktaufruf gefunden wird.

Ein interessanter Messwert ergibt sich aus dem Verhältnis von ein- und ausgehenden Strömen. Setzt man beispielsweise die Anzahl der Vorgänger ins Verhältnis zur Anzahl der Nachfolger, so drückt dieses Verhältnis aus, ob der Knoten als Durchgangspunkt bzw. Verteiler dient oder ob hier ein inhaltlicher Endpunkt existiert. Gibt es in diesem Sinne nur wenige Vorgänger-Dokumente aber viele Nachfolgerdokumente, so dient der Knoten als Verteilerstelle und leitet Benutzer zielgerichtet nach Interessen weiter. Gibt es hingegen deutlich mehr Vorgängerdokumente als Nachfolger, so heißt dies, dass die Benutzer von sehr vielen Orten herströmen, sich aber vom aktuellen

Knoten aus homogen weiterbewegen. In diesen Fällen sollte eine Klick-Pfad- (siehe Seite 37) oder Kookurrenz-Auswertung (siehe Seite 35) vorgenommen werden, um zu erkennen, wie sich die Benutzer im Detail weiterbewegen und ob möglicherweise überflüssige Zwischenpunkte eliminierbar sind.

Sofern Vorgänger erfasst sind, ist die Erreichbarkeit des aktuellen Objekts per Definition gewährleistet. Ist die Menge der Vorgänger groß, so existieren auch viele Wege zum aktuellen Objekt, wobei aufgrund der Eigenschaften des Messwerts auch gewährleistet ist, dass die Benutzer diese Wege kennen.

3.6 Nachfolger und Zielpfade

Dieser Messwert beschreibt die Anzahl der Ziele, die vom aktuellen Knoten aus direkt oder indirekt angesteuert werden. Auf diese Weise erhält der Analyst Hinweise auf die Effektivität der Navigationsnutzung der Präsenz.

Berechnung

$$\begin{aligned} successors(\mathcal{A}, \mathcal{O}) &= |\{(o) \in \mathbb{O} | \exists a_1, a_2 \in \mathcal{A} : a_1[object] = o \\ &\quad \wedge a_2[object] \in \mathcal{O} \wedge a_1 \succ a_2\}| \\ outpaths(\mathcal{A}, \mathcal{O}) &= |\{(o) \in \mathbb{O} | \exists a_1, a_2 \in \mathcal{A} : a_1[object] = o \\ &\quad \wedge a_2[object] \in \mathcal{O} \wedge a_1 \overset{+}{\succ} a_2\}| \end{aligned} \tag{3.7}$$

3.6.1 Eigenschaften

Wie auch bei der Erfassung der Vorgängerobjekte gilt es bei der Ermittlung der Nachfolger bzw. Zielpfade die Schwierigkeit der Objektidentifikation zu meistern.

3.6.2 Bedeutung

Aus der Anzahl der Nachfolgerdokumente bzw. Zielpfade lässt sich ein grundlegender Einblick in die Nutzungsstruktur der Präsenz aus Sicht des Knotens gewinnen.

So drückt eine große Anzahl von Zielpfaden aus, dass der Punkt selbst im Wesentlichen Verteilungscharakter besitzt und das Auffinden entfernter Knoten vermittelt. Ist dabei die Anzahl der direkten Nachfolger bereits groß, wird die Verteilungsentscheidung direkt am aktuellen Knoten getroffen. Ist sie vergleichsweise klein, so findet die Verteilung erst später statt.

Gibt es hingegen für einen Knoten nur wenige Zielpfade, so handelt es sich entweder um einen Zielknoten oder um einen Knoten, der die Benutzer stark gerichtet deligiert. Falls die Session Killer-Information (siehe Seite 33) an dieser Stelle besagt, dass der Knoten nur in wenigen Fällen einen Endpunkt bildet, sollte die allgemeine Relevanz des Knotens überprüft werden, da der Knoten in Verbindung mit kurzen Verweilzeiten (siehe Seite 18) als überflüssig wahrgenommen wird.

3.7 Durchschnittliche Pfadlänge

Die Pfadlänge ist zu verstehen als die Anzahl der Clicks, die Benutzer benötigt haben, um zum aktuellen Objekt zu gelangen.

Berechnung

$$avgClicks(\mathcal{A}, \mathcal{O}) = |\{a_1 \in \mathcal{A} | \exists a_2 \in \mathcal{A} : a_2[object] \in \mathcal{O} \wedge a_1 \overset{+}{\prec} a_2\}| / hits(\mathcal{A}, \mathcal{O}) \tag{3.8}$$

Eigenschaften

Dieser Messwert ist in vielen Situationen leicht ermittelbar. In Verbindung mit Session-Tracking (siehe Seite 47) braucht man in der Session lediglich einen Zähler mitzuführen, der die Anzahl der Klicks zum Erreichen des aktuellen Knotens wiederspiegelt. Schwierig wird es allerdings, wenn man berücksichtigt, dass die Benutzer i.d.R. bereits vor dem Erreichen eines beobachteten Bereichs Aktionen durchführen, die inhaltlich den Wert mitbestimmen würden.

3.7.1 Bedeutung

Die durchschnittliche Pfadlänge betrachtet Positionen in Besuchen. So bildet dieser Messwert ein direktes Indiz für die relative Positionierung des Objekts in einer gewöhnlichen Sitzung. [40] Je kleiner der Wert, um so früher wird das Objekt erreicht. Je größer der Wert, um so mehr zwischenliegende Schritte sind notwendig, um zum Objekt zu gelangen.

3.8 Session-Starter

„Session-Starter“ sind die Knoten einer Sitzung, die den Anfang eines Besuchs bilden. Der Messwert „Session-Starter“ drückt dabei aus, wie häufig ein Objekt als Anfangspunkt vorkam.

Berechnung

$$sessStart(\mathcal{A}, \mathcal{O}) = |\{a \in \mathcal{A} | a[object] \in \mathcal{O} \wedge a \overset{Start}{\prec} a\}| \tag{3.9}$$

3.8.1 Eigenschaften

Die „Session-Starter“-Häufigkeit kann technisch vergleichsweise leicht erfasst werden, sofern Sitzungen in irgendeiner Form erkennbar sind. Letztendlich handelt es sich bei einem „Session-Starter“-Knoten einfach um den zuerst aufgerufenen Knoten einer Sitzung. Messungenauigkeiten entstehen lediglich, wenn Sitzungen nicht präzise erfasst werden oder zusammengehörende Teile nicht als eine Sitzung erkannt werden.

3.8.2 Bedeutung

Eine hohe „Session-Starter“-Häufigkeit ist bei Portalseiten oder Anmeldedialogen zu erwarten, da diese den inhaltlichen Startpunkt einer Site bilden. In der Praxis verweisen Suchmaschinen und externe Sites jedoch meist auf speziellere Dokumente, sodass diese häufig den tatsächlichen Eintrittspunkt in eine Präsenz bilden. Aufgrund der im Internet herrschenden Informationsüberflutung entscheiden sich die Besucher meist direkt auf diesen Seiten, ob sie tiefer in die Präsenz hineinnavigieren oder nicht. Im Sinne des Marketing oder einer Optimierung der Dokumentennavigationsstruktur sind daher solche Punkte von besonderer Bedeutung, da hier der prägende “erste Eindruck“ gewonnen wird.

„Session-Starter“-Punkte sind Punkte einer Präsenz, die Benutzer initial ansteuern. Der Inhalt dieses Knotens birgt daher den meisten Information über die Interessen eines Besuchers, denn hier gelangen sie hin, ohne ihre Interessen an die Restriktionen einer Präsenz angepasst zu haben. Je nachdem, ob die Benutzer den Knoten wirklich als Eintrittspunkt nutzen, also weitere Inhalte

aufrufen oder direkt wieder verlassen, kann erkannt werden, ob der gebotene Inhalt den tatsächlichen Benutzerinteressen entspricht oder nur aufgrund von Mehrdeutigkeit bzw. suchmaschinenbedingter Informationsverdichtung erreicht wurde.

Die Auswertung der „Session-Starter"-Häufigkeit drückt aus, wie häufig externe Knoten den Weg zum aktuellen Knoten bereitet haben. Auf diese Weise können die Auswirkungen von extern liegenden Werbe-Bannern oder auch die Wirksamkeit von ansonsten nicht technisch zuordbaren Werbemitteln beurteilt werden. So könnte beispielsweise auf einem Plakat eine konkrete Internetadresse beworben werden und zur Auswertung der Wirksamkeit dieses Plakates die „Session-Starter"-Häufigkiet für die beworbene Adresse beobachtet werden.

3.9 Session-Killer

„Session-Killer“-Knoten sind Objekte, die den Endpunkt einer Sitzung bilden. Es handelt sich also um solche Punkte, die der Benutzer zuletzt aufgerufen hat, bevor er eine Präsenz verlassen hat. Der an dieser Stelle betrachtete Messwert „Session-Killer“ beschreibt die Häufigkeit, mit der ein Knoten als „Session-Killer“ auftritt.

Berechnung

$$sessKill(\mathcal{A}, \mathcal{O}) = |\{a \in \mathcal{A} | a[object] \in \mathcal{O} \wedge a \overset{End}{\succ} a\}| \tag{3.10}$$

3.9.1 Eigenschaften

Technisch gesehen werden „Session-Killer“ dadurch erkannt, dass keine weiteren Knoten derselben Sitzung folgen.

3.9.2 Bedeutung

Prinzipiell sind „Session-Killer“-Objekte als Knoten zu verstehen, an denen Benutzer das (vorläufige oder prinzipielle) Interessse an einer Seite verloren haben. Dabei gibt es in der Regel drei Gründe für diesen Interessensverlust. So kann eine Tätigkeit (z.B. das Aufsuchen einer Information) erfolgreich abgeschlossen worden sein oder der Benutzer hat die Durchführung abgebrochen, da er seinem Ziel nicht näher kam. Ein dritte Möglichkeit bilden äußere Einflüsse, durch die die aktuelle Tätigkeit als zweitrangig betrachtet und zurückgestellt bzw. abgebrochen wurde.

In der Praxis ist die dritte Variante für hohe „Session-Killer“-Werte sehr unwahrscheinlich, da eine Konzentration von hohen „Session-Killer“-Werten auf ein konkretes Objekt ein stochastisches Indiz für einen Zusammenhang mit dem Objekt selbst bildet. In diesen Fällen gilt daher zu entscheiden, ob der erwähnte Interessensverlust im positiven oder im negativen Sinne zu sehen ist. In einigen Situationen kann diese Entscheidung trivial getroffen werden: Wurde am besagten Knoten beispielsweise ein Bestellungsformular abgeschickt, so kann sicherlich von einem erfolgreichen Abschluss die Rede sein. Auf rein informativen Seiten ist diese Entscheidung jedoch wesentlich schwerer zu treffen. In diesen Situationen bleibt einzig und allein die Option, den Messwert

„Verweilzeit“ zu Rate zu ziehen um auf diese Weise nähere Erkenntnisse zu gewinnen.

Dieser Messwert hilft dabei Entscheidungsstellen in der Präsenz aufzudecken, also Punkte, die die Benutzer als Point-Of-Frustration oder Point-Of-Success werten.

Letztendlich ist es wichtig, die „Session-Killer“-Knoten so zu optimieren, dass diese aus Benutzersicht zu einem Point-Of-Success werden. Dienstanbieter sollten sich bei der Auswertung eine Site speziell auf diese Knoten konzentrieren, da diese Punkte einen Point-Of-No-Return bilden und damit die letzte Möglichkeit, Kundenfrustration zu verhindern.

3.10 Kookurrenz

Kookurrenzen betrachten das zusammenhängende Auftreten von Objekten. Es wird die Wahrscheinlichkeit abgefragt, mit der in einer beliebigen Sitzung zwei voneinander unabhängige Objekte aufgerufen werden.

Entsprechend der Richtung, in der die Kookurrenz betrachtet werden, kann man zwischen eingehenden, ausgehenden und ungerichteten Kookurrenzen unterscheiden. Eingehende Kookurrenzen betrachten die Wahrscheinlichkeit, mit der das betrachtete Objekt besucht wird, unter der Bedingung, dass zuvor ein konkretes anderes Objekt besucht wurde.

Ausgehende Kookurrenzen betrachten analog die Wahrscheinlichkeit, dass ein konkretes anderes Objekt aufgerufen wird, nachdem das aktuelle Objekt aufgerufen wurde.

Ungerichtete Kookurrenzen abstrahieren von der Reihenfolge, in der die Objekte aufgerufen wurden und betrachten ausschließlich die Wahrscheinlichkeit, mit der in einer Sitzung, die den Aufruf eines gegebenen Objekts enthält, ein konkretes anderes Objekt aufgerufen wird.

Berechnung

$$
\begin{aligned}
preoccur(\mathcal{A}, \mathcal{O}_1, \mathcal{O}_2) &= |\{(a_1, a_2) \in \mathcal{A} \times \mathcal{A} | a_1 \overset{+}{\prec} a_2 \wedge a_1[object] \in \mathcal{O}_1 \\
&\quad \wedge a_2[object] \in \mathcal{O}_2\}| \\
postoccur(\mathcal{A}, \mathcal{O}_1, \mathcal{O}_2) &= |\{(a_1, a_2) \in \mathcal{A} \times \mathcal{A} | a_1 \overset{+}{\succ} a_2 \wedge a_1[object] \in \mathcal{O}_1 \\
&\quad \wedge a_2[object] \in \mathcal{O}_2\}| \\
cooccur(\mathcal{A}, \mathcal{O}_1, \mathcal{O}_2) &= preoccur(\mathcal{A}, \mathcal{O}_1, \mathcal{O}_2) + postoccur(\mathcal{A}, \mathcal{O}_1, \mathcal{O}_2)
\end{aligned}
\tag{3.11}
$$

3.10.1 Eigenschaften

Kernaussage der Kookurrenz ist die Wahrscheinlich, dass ein Objekt besucht wird, unter der Bedingung, dass ein anderes Objekt davor, danach bzw. im gleichen Kontext aufgerufen wurde.

Die Schwierigkeit der Kookurrenzauswertungen liegt oft darin begründet, dass nicht von zwei konkreten Knoten ausgegangen wird und deren Kookur-

renzwert betrachtet wird, sondern vielmehr Knotenmengen gefunden werden sollen, deren Kookurrenz einen gewissen Schwellenwert überschreiten. Zur Berechnung dieser Mengen muss theoretisch für je zwei Knoten ermittelt werden, wie häufig diese gemeinsam auftraten. Man bewegt sich somit leicht in quadratischen Aufwandsklassen, die bei großen Präsenzen mit mehreren tausend Dokumenten leicht zum Problem werden.

3.10.2 Bedeutung

Durch Analyse von Kookurrenzen ist es möglich, wahrgenommene inhaltliche Beziehungen von Knoten zu beurteilen. Dabei fokussiert die Kookurrenz-Auswertung das cliquenhafte Auftreten von Objekten. Man entscheidet zwischen eingehenden, ausgehenden und ungerichteten Kookurrenzen.

Hohe Kookurrenzwerte zeigen Stellen auf, an denen via Precomputing oder Prefetching eine höhere Interaktivität bzw. kürzere Systemreaktionszeiten erreicht werden können.

Kookurrenzen zeigen Verbindungen in den Navigationsstrukturen auf und ermöglichen somit einen Einblick in die Verbindungen zwischen Objekten aus Sicht der Benutzer. Objekte, die häufig zusammenhängend aufgerufen werden, werden somit von Benutzern als zusammengehörig verstanden.

Um die Benutzerkonfidenz von Besuchern eines Objekts zu beurteilen, kann es auch hilfreich sein, die Kookurrenzen zu diesem Objekt zu betrachten. Je mehr tatsächliche inhaltliche Verwandschaft die kookurrenten Objekte zueinander haben, um so wahrscheinlicher ist die Übereinstimmung von inhaltlicher Erwartung des Benutzers und gebotenem Inhalt.

3.11 Klickpfade

Klickpfade werden in der Literatur häufig als zeitlich geordnete Listen von Seitenaufrufen ($\langle o_1, o_2 \dots o_n \rangle$ mit $o_i \in \mathbb{O}$) beschrieben, die Benutzer auf ihrem Weg zu einem konkreten Objekt zurücklegen [32, 3]. In dieser Arbeit möchte ich eine Verallgemeinerung dieses Konstrukts nutzen, um dem dieser Arbeit zu Grunde liegenden „Zoom"-Prinzip gerecht zu werden. Dabei werden nicht mehr einzelne Objekte als Sequenz-Elemente genutzt. Vielmehr treten an deren Stellen Objektmengen, die ganze Inhaltsbereiche umfassen können. Ein Klickpfad ist also in dieser Arbeit als eine Sequenz $\langle \mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n \rangle$ mit $\mathcal{O}_i \subseteq \mathbb{O}$ zu verstehen.

Bei der Analyse von Klickpfaden geht es darum, die genauen Wege der Benutzer zu studieren, um Informationen über ihre Intensionen zu erlangen. Um jedoch wesentliche Klickpfade von unwesentlichen zu unterscheiden, ist ein Messwert von Nöten, der die Häufigkeit eines konkreten Klickpfades in einem Zugriffsprotokoll wiederspiegelt.

Berechnung Zur Berechnung des eigentlichen Messwerts sei zunächst eine Hilfsfunktion eingeführt, die zu einem gegebenen
Klickpfad ($\langle \mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n \rangle$) die entsprechenden Instanzstartpunkte aus einem Zugriffsprotokoll $\mathcal{A}$ ermittelt.

$$\begin{aligned} instances(\mathcal{A}, \langle \mathcal{O}_1 \rangle) = & \quad \{ \quad a_1 \in \mathcal{A} | a_1[object] \in \mathcal{O}_1 \quad \} \\ instances(\mathcal{A}, \langle \mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n \rangle) = & \quad \{ \quad a_1 \in instances(\mathcal{A}, \langle \mathcal{O}_1 \rangle) | \\ & \qquad \exists a_2 \in instances(\mathcal{A}, \langle \mathcal{O}_2, \dots \mathcal{O}_n \rangle) : \\ & \qquad a_1 \prec a_2 \\ & \quad \} \end{aligned} \tag{3.12}$$

Der Messwert selbst ergibt sich schließlich durch zählen der Elemente aus dieser Instanzmenge:

$$clickPaths(\mathcal{A}, \langle \mathcal{O}_1, \dots, \mathcal{O}_n \rangle) = |instances(\mathcal{A}, \langle \mathcal{O}_1, \dots, \mathcal{O}_n \rangle)| \tag{3.13}$$

3.11.1 Eigenschaften

Für die zweckmäßige Analyse ist es notwendig, dass ein System in der Lage ist, effizient die häufigsten Klickpfade mit bestimmten Eigenschaften ermitteln zu können. Dabei sind Klickpfade von besonderer Bedeutung, die ihren Anfang oder ihr Ende in einem gegebenen Objektbereich nehmen.

Die Analyse von Klickpfaden weist sehr viele Parallelitäten zu Reisedatenbanken für Flugzeugpassagiere, die unterschiedliche Städte anfliegen, auf. In unserem Falle sind die Passagiere zwar Web-Seitenbenutzer und die Flughäfen Seiten einer Web-Präsenz, dennoch ist das Problem das Gleiche, wir müssen Reisen modellieren. Andersen und andere [3] zeigen, wie Reisetheorien auf Klickpfade angewendet werden können, wobei sie analysieren wie die Benutzer in dem Netzwerk von Web-Seiten reisen. Danach ist das Modellieren dieses Netzwerks ein Schlüsselelement zum Modellieren der Nutzungsinformation von Klickströmen. In ihrem „Subsession"-Modell betrachten sie die möglichen Teilpfade aller Klickpfade, sodass schnelle Abfragen auf ganzen Sub-Sessions ausführbar werden [3]. Aufbauend auf dieser Technik werde ich in Abschnitt 7.8 eine optimierte Berechnung vorstellen.

3.11.2 Bedeutung

Um zu erfahren, warum Benutzer etwas machen, gilt es zu hinterfragen, woher sie kommen. Klickpfade liefern die dafür notwendigen Informationen. Dabei dokumentieren sie das genaue Navigationsverhalten eines Besuchers in einer Site. In einer Analyse kann beispielsweise geprüft werden, ob Zyklen auftreten, also Dokumente mehrfach aufgerufen wurden oder welche Untersequenzen häufig genutzt wurden. Außerdem gibt eine Auswertung detailliert darüber Auskunft, ob Besucher die Site in einer vom Anbieter vorgesehenen Art und Weise durchqueren.

Die aus einer Klickpfad-Analyse hervorgehenden Information legen zudem offen, ob Besucher hin- und herirren oder ob sie sich in bisher unvorhergesehener Weise gezielt auf konkrete Objekte hinbewegen.

Betrachtet man die Bewegungen von Besuchern in einer Site, welche Knoten sie vor dem Aufruf eines konkreten Knotens besucht haben und welche danach, so lassen sich daraus Rückschlüsse ziehen, ob der aktuelle Knoten mit den inhaltlichen Anforderungen des Besuchers korrespondiert oder eher zufällig aufgerufen wurde. So kann beispielsweise davon ausgegangen werden, dass ein Besucher sich nicht für „Kochrezepte" interessiert, wenn er sich hauptsächlich in einem Inhaltsbereich „Unternehmensberatung" bewegt und plötzlich ein Dokument aus erstgenanntem Bereich aufruft.

Anhand der Klickpfad-Analyse kann man erkennen, ob Benutzer sich in zusammengehörenden Themenbereichen bewegen oder eher eine konfuse Wegwahl einschlagen. Eine Auswertung des Weges, den sie dabei zurücklegen, legt diese Zielstrebigkeit offen.

Speziell in Verbindung mit Web-orientierten Marketingaktionen, zum Beispiel dem Schalten eines Werbebanners, geben Klickpfadanalysen auch die Möglichkeit zur Effektivitätsbeurteilung. So kann anhand eines Klickpfades jeder Knoten auf dem Weg zu einem Bereich analysiert werden und die Nutzung entsprechender Wegweiser direkt eingesehen werden.

3.12 Suchbegriffe

In den meisten Situationen, in denen Besucher eine Seite im Internet aufsuchen, passiert dies durch entsprechende Suchanfragen in Suchmaschinen. Zur Auswertung der Auffindbarkeit von Dokumenten ist es daher notwendig zu beurteilen, in welchen Situationen ein Dokument überhaupt von einer Suchmaschine gefunden wird. Dazu kann ausgewertet werden, unter welchen Suchbegriffen das Dokument in einem Suchergebnis auftrat, sodass Benutzer es anschließend aufrufen konnten.

Berechnung Im Sinne dieser Arbeit handelt es sich bei der Analyse von Suchbegriffen eigentlich nicht um einen speziellen Messwert, sondern vielmehr um eine Erweiterung der bislang betrachteten Objektdomäne $\mathbb{O}$. So entspricht jede Suchmaschinenergebnisseite einem speziellen Objekt $o\mathbb{O}$ und jeder Suchbegriff (in Konjunktionen) einem Objektbereich $\mathcal{O} \subseteq \mathbb{O}$.

An dieser Stelle sei angemerkt, dass jede bool'sche Anfrage in eine disjunktive Normalform [37] überführt werden kann, wobei die Disjunktionen aus Benutzersicht als das parallele Aufrufen unterschiedlicher Suchmaschinenergebnisseiten verstanden werden können und somit dem gleichzeitigen Aufruf mehrerer Objekte $o_1, \ldots, o_n \in \mathbb{O}$ entsprechen.

3.12.1 Eigenschaften

Für die Auswertung von Suchbegriffen gibt es kein Standardverfahren. Wenngleich der beste Weg zum Analysieren von Suchbegriffen in der direkten Koppelung einer Suchmaschine mit dem Mining-System liegt, ist dies bei extern liegenden Suchmaschinen in den seltensten Fällen möglich. Es bleibt jedoch die Option, „intelligent“ ggfs. verfügbare „Referer“-Informationen (vgl. Abschnitt 5.1.2) zu interpretieren.

Zur „Referer“-Interpretation muss das Mining-System in der Lage sein, die URL-Syntax der jeweiligen Suchmaschine zu verstehen. Viele Suchmaschinen nutzen zwar für die Formulierung von Suchanfragen HTTP-GET-Requests [16] basierend auf URL-Encoding [10], sodass die notwendigen Informationen in den „Referer“-Daten existieren, allerdings zerlegen unterschiedliche Suchmaschinen die ihnen gestellten Anfragen auf heterogene Art und Weise. So nutzt „GOOGLE“ beispielsweise einen Parameter „q“ für die Übertragung des gestellten Anfragestrings, während „fireball.de“ dafür den Pa-

rameter „query“ verwendet. Darüber hinaus besitzen unterschiedliche Suchmaschinen in der Regel unterschiedliche Features mit jeweils proprietärer Syntax, sodass eine Normierung von Suchanfragen hohe Ansprüche an ein Mining-System stellt. Die Analyse gängiger Suchmaschinensprachen würde den Rahmen dieser Arbeit sprengen. Es liegt allerdings auf der Hand, dass es lediglich eine Fleißarbeit ist, Anfragen gängiger Suchmaschinen zu normalisieren und in eine gerichtete azyklische Graphenstruktur zu bringen.

Neben der Vielzahl von Interpretationsmöglichkeiten beliebiger CGI-Parameter ist anzumerken, dass viele Browser oder auch „personal Firewalls“ aus Datenschutzgründen die „Referer“-Informationen nicht mehr an die Web-Server übertragen. In Zukunft wird es also zunehmend schwerer werden, derartige Informationen zu gewinnen.

3.12.2 Bedeutung

Betrachtet man Suchbegriffe als „Referer“ und ermittelt, wie viele unterschiedliche Schlüsselwörter mit dem aktuellen Bereich assoziiert sind, ergibt sich die Möglichkeit zu erkennen, wie die Inhalte von außen vernetzt sind und damit die Option externe Zuordnungen eigener Inhalte zu validieren.

Durch Kookurrenzenanalyse ist es möglich zu erkennen, wie gut eigene Inhalte sich im Netz gegenüber anderen Inhalten aus gleichen Themengebieten behaupten. Eine hohe Häufigkeit von einem Schlüsselwort in Verbindung mit einem Inhalt lässt sich in der Regel direkt auf eine gute Positionierung des jeweiligen Inhalts unter dem besagten Schlüsselwort in Suchmaschinen zurückführen. [31]

Suchanfragen werden direkt vom Benutzer gestellt und drücken somit sein konkretes Inhaltsbedürfnis in eigenen Worten aus. So gesehen können Suchanfragen als authentische Repräsentation des situativen Benutzerinteresses verstanden werden, die nicht durch beispielsweise Beschränkungen existierender Navigationsmöglichkeiten eingegrenzt wurde. Durch Abgleich dieser Anfrage mit den eigentlichen Inhalten lässt sich somit beurteilen, in welchem Maße ein geeignetes Klientel erreicht wird.

In gleichem Maße kann auch die Benutzerkonfidenz beim Erreichen eines Objekts beurteilt werden. Gibt ein Benutzer die Suchanfrage „Was ist ein Chamäleon?“ ein, so ist einer natürlichen Person sofort klar, dass der Benutzer Informationen über ein echsenartiges Tier sucht. Wird diese Suchanfrage allerdings in Verbindung mit der Startseite der „CHAMAELEON AG für in-

novative Netzlösungen“ festgestellt, so zeigt dies sofort eine Fehlzuordnung an.

Kapitel 4

Benutzeridentifikation

Im vorangehenden Kapitel wurden die im konzipierten System verfügbaren Messwerte aus logischer Perspektive vorgestellt. Dabei wurde gezeigt, dass die Messwerte anhand eines trivialen Zugriffsprotokolls, welches über eine einfache Relation (vgl. Figur 3.1) darstellbar ist, berechnet werden können. Ein wesentlicher Bestandteil dieser Relation ist die Komponente „*user*“, über die eine konkrete Entität der Relation ausdrückt, welcher Benutzer einen Zugriff vorgenommen hat.

Das vorliegende Kapitel beschäftigt sich mit der Frage, wie diese Komponente bei einer Entitätserzeugung belegt werden kann. Dabei steht das Problem, dass durchgeführte HTTP-Aufrufe zunächst anonym sind und Zugriffe nicht immer eindeutig einer konkreten Person zugeordnet werden können, im Mittelpunkt. Trotzdem ist es möglich, sich dieser tatsächlichen Personenzuordnung auf verschiedene Weise anzunähern. Es werden drei konzeptionell unterschiedliche Ansätze vorgestellt, die jeweils eigene Vor- und Nachteile besitzen, deren Zusammenwirken der Realität jedoch sehr nahekommt.

4.1 Host-Tracking

Host-Tracking ist die einfachste Form der Benutzerzuordnung. Ein Benutzer wird dabei einzig und allein anhand eines Host-Keys identifiziert, sodass jeder Rechner genau einer realen Person entspricht.

4.1.1 Eigenschaften

Wenngleich die beste Host-Key-Variante sicherlich eine CPU-ID oder eine MAC-Adresse bildet [24], sind derartige Werte in der von HTML, Javascript und HTTP dominierten WWW-Domäne i.d.R. nicht zugänglich. Als Host-Keys wird daher in der Praxis häufig auf IP-Adressen bzw. persistente Browser-Cookies zurückgegriffen.

Beim IP-basierten Host-Tracking wird die Adresse, die der Server als Kommunikationsziel nutzt, als Identifikationsschlüssel eingesetzt. Dabei wird ausgenutzt, dass das IP-Protokoll [36] als ein End-Zu-End-Protokoll entworfen wurde und somit, zumindest für den Fall, dass jeder Rechner genau eine IP-Adresse hat, einen Rechner eindeutig indentifiziert.

Alternativ zum IP-basierten Host-Tracking kann auch gemäß dem „HTTP State Management Mechanism“ [27] beim erstmaligen Betreten einer Web-Präsenz ein persistenter Cookie mit einem eindeutigen Wert erzeugt werden, der dann als Identifikationsmerkmal angewendet werden kann.

Für den Fall, dass jede Person genau mit einem Rechner arbeitet und umgekehrt jeder Rechner nur von genau einer Person genutzt wird, erzeugen die oben aufgeführten Verfahren zumindest in der Theorie geeignete Merkmale zur Personenidentifikation.

4.1.2 Bedeutung

Ein besonderer Vorteil des IP-adressbasierten Ansatzes ist, dass eine Adresse grundlegende Vorraussetzung für eine Kommunikation ist. Damit liegt der zentrale Vorteil bei der Nutzung dieses Identifikationsansatzes darin, dass die notwendigen Daten immer vorhanden sind. Bei jedem Objekt-Zugriff, kann die IP-Adresse des jeweiligen Partners ermittelt werden und somit eine Zuordnung getroffen werden, die ohne weitere Authentifikationsmechanismen auskommt.

Neben der prinzipiellen Verfügbarkeit ist zudem für jede IP-Adresse in sogenannten WHOIS-Datenbanken [12] erfasst, welche Organisation die Kontrolle über die jeweilige IP-Adresse besitzt. Dabei werden auch nähere Informationen wie zum Beispiel Name, Land, Region und Stadt über die Organisationen bereitgestellt, die bei der Auswertung von Profil-Informationen genutzt werden können. Häufig kann auf diese Weise zumindest das Land eingegrenzt werden, aus dem ein Benutzer auf die Inhalte zugegriffen hat. Aufgrund global agierender Internetanbieter ist in vielen Fällen jedoch eine feinere Zuordnung anhand dieser Daten kaum möglich.

Die korrekte Rechnerzuordnung anhand von IP-Adressen wird in der Praxis durch das hier zum Einsatz kommende IPv4 Protokoll [36] beinträchtigt. So zwingt der hier begrenzte Adressraums Internetdienstanbieter, wenige IP-Adressen durch Proxy- [1] oder NAT [39]-Technologie auf mehrere Nutzer aufzuteilen, sodass am aufgerufenen Kommunikationsende ein und dieselbe IP-Adresse auf mehrere Endbenuzter passen kann. Die korrekte Aufschlüsselung des Endnutzers kann dann auch nur am Proxy- oder NAT-Knoten im Netzwerk vorgenommen werden, nicht jedoch am Endpunkt, der den eigentlichen Aufruf erfassen soll. Internet-Provider setzen zudem häufig Verfahren zur dynamischen Adressenvergabe [15] ein, wodurch Rechner in der Regel bei jeder Internetsitzung eine andere IP-Adresse haben und das Erkennen wiederkehrender Benutzer mit hoher Fehleranfälligkeit behaftet wird.

Wie bereits erwähnt, ermöglicht der „HTTP State Management Mechanism“ [27] ein alternatives Verfahren. Dabei wird bei Betreten einer Web-Site ein persistenter Cookie mit einem eindeutigen Schlüssel gesetzt, also ein Browsergebundenes Identifikationsmerkmal, welches bei folgenden HTTP-Anfragen an einen Server mitgereicht wird. Dieser Schlüssel identifiziert zwar keinen Rechner mehr, sondern vielmehr eine spezielle Browser-Instanz, jedoch kann in der Praxis davon ausgegangen werden, dass Benutzer beim Browsen im Internet im Wesentlichen ein präferiertes Werkzeug nutzen. Dieser Schlüssel ist dann unabhängig von der Netzwerktopologie und somit auch in der Lage, das Wiederkehren von Benutzern bei dynamischer Netzanbindung zu erkennen.

Wenngleich keine allgemeine Technik in der Lage ist, die Authentität einer realen Person zu gewährleisten ohne vom Benutzer direkt personenbezogene Daten abzufragen, sei an dieser Stelle nochmals darauf hingewiesen, dass IP-Adressen Computer- oder vielmehr Netzwerk-Schnittstellen und persistente Cookies Browserinstanzen identifizieren und nicht die diese nutzenden Personen. So gibt es bei der Zuordnung von IP-Adressen zu Personen bereits Probleme, wenn Arbeitsplätze von mehreren Personen geteilt werden oder

eine Person von unterschiedlichen Arbeitsplätzen aus agiert.

4.2 Session-Tracking

Ziel des Session-Trackings ist es, Benutzersitzungen oder „Besuche“ zu überwachen. Dazu erhält ein Besucher i.d.R. beim Betreten einer Web-Site einen Sitzungsschlüssel zugewiesen, der bei Folgeanfragen an den Server übertragen wird und so für die Dauer einer gesamten Sitzung einen, für diese Sitzung eindeutigen, Schlüssel mitreicht.

4.2.1 Eigenschaften

Die bekannteste Form zum Session-Tracking ist wohl das Cookie-Tracking [27], wobei beim Betreten einer Web-Site ein temporärer Cookie im Browser hinterlegt wird, der in Folgeaufrufen vom Browser an den Server mitgereicht wird und so dem Server die Sitzungszusammengehörigkeit bekanntgibt.

Temporäre Cookies haben die Eigenschaft, dass sie beim Schließen eines Browsers gelöscht werden und damit nur so lange zur Verfügung stehen, bis ein Benutzer seinen Browser und damit seinen Gang ins Internet beendet. Dabei wird explizit berücksichtigt, dass Benutzer im Laufe der Zeit mit unterschiedlichen Kennungen arbeiten. Obgleich die im vorangegangenen Kapitel beschriebenen persistenten Cookies potentiell höhere Informationen bergen, sei angemerkt, dass viele Browser das Verwenden temporärer Cookies zulassen, persistente Cookies aber aus Sicherheitsgründen beim Beenden löschen, sodass diese sich in der Realität oft wie temporäre Cookies darstellen.

Der Hauptnachteil von Cookies ist, dass einige Benutzer die Verwendung von Cookies in ihrer Web-Software vollständig unterbinden. Damit ist aber auch gleichzeitig die Möglichkeit zum Cookie-orientierten Tracking genommen.

Eine zweite Form des Session-Trackings basiert in der Möglichkeit anwendungsspezifische Session-Keys zu überwachen. Viele Programmiersprachen, wie zum Beispiel JSP [29] oder PHP [2], bieten hierzu spezielle Mechanismen an, um in Verlinkungen von einer Seite zur nächsten einen CGI-Parameter zur Weitergabe eines Session-Keys zu erzeugen. Die so übertragenen Session-Keys können ebenfalls von einem Tracking-System ausgewertet werden, setzen aber voraus, dass die zu überwachende Web-Site oder Web-Anwendung sämtliche Seitenübergänge entsprechend aufbereitet. Prinzipiell ist bei der Vermischung von Aufruf-Adresse und Session-Information das bewusste, aber auch das unbewusste Session-Hijacking [11] zu berücksichtigen. So passiert es allzu leicht, dass ein Benutzer eine mit Sitzungsschlüssel versehene Web-

Adresse per Copy-Paste als Link in einer eigenen Seite einsetzt. Benutzer, die dann über diesen Link auf die Zielseite gelangen, besitzen die gleiche Sitzungskennung, wie der Benutzer, der den Link zur Verfügung gestellt hat. In diesen Fällen müssen zusätzliche Kriterien (z.B. die IP-Adresse des Nutzers) hinzugezogen werden, wodurch der Gültigkeitsraum einer Sitzungskennung eingegrenzt und ambiguente Nutzung verhindert wird.

Das Sub-Domain-Tracking löst zwar das Session-Hijacking-Problem [11] nicht, stellt aber dafür nur geringe Anforderungen für den Einsatz. Die Grundphilosophie ist es hierbei, einem Benutzer bei Betreten einer Web-Seite eine sitzungsspezifische Domaine zuzuweisen, mit der er auf der jeweiligen Seite navigiert. Betritt ein Benutzer zum Beispiel die Internet-Adresse „http://www.example.com/...“, wird für ihn eine spezielle Session-ID ω erzeugt und er wird auf die URL „http://ω.www.example.com/...“ weitergeleitet. Von dort an arbeitet die Web-Site nur noch mit relativen Pfaden, sodass der Sessionkontext erst verlassen wird, wenn der Benutzer erneut über „http://www.example.com/...“ die Web-Site betritt.

Ein Problem an dieser Technik ist, dass beispielsweise Internet-Spider ebenfalls beim Betreten der Site eine neue Domaine zugewiesen bekommen. Dies ist zwar auch in den vorherigen Verfahren der Fall, jedoch können generische Spider im Falle des Sub-Domain-Trackings nicht erkennen, dass die jeweilige Sitzungs-Subdomäne keinen Lokalisationsaspekt, sondern Parameteraspekt darstellt. Das hat zur Folge, dass Links von Suchmaschinen direkt auf eine mit Sitzungsschlüssel behaftete Adresse zeigen und Nutzer, die diese Adresse in einer Suchmaschine aufrufen, immer den Sitzungsschlüssel benutzen, den der Suchmaschinenspider beim Erfassen der Inhalte zugewiesen bekam. Man könnte zwar Server-seitig den Gültigkeitsraum eines Sitzungsschlüssels auf eine spezielle IP-Adresse oder einen bestimmten Zeitraum begrenzen und dadurch die Robustheit des Verfahrens stärken, darf dabei allerdings nicht die entstehenden Seiteneffekte für das eigentliche System unterschätzen. So kopieren viele Web-Seiten-Redakteure zum Verweisen auf Fremdinhalte einfach eine Internetadresse unverändert in ihre Seiten. Suchmaschinen, die beispielsweise einen Page-Rank anhand der Verlinkungsintensität von Seiten bestimmen, sind dann nicht mehr in der Lage korrekte Verlinkungshäufigkeiten zu bestimmen, was prinzipiell fatale Folgen für die Auffindbarkeit von Internet-Inhalten hat. Ein zusätzlicher Nachteil ergibt sich aus der Tatsache, dass für jeden Benutzer ein neuer Domain-Name erzeugt wird. DNS-Server können daher diese Informationen kaum „cachen“ was wiederum zu zahlreichen DNS-LookUps in der Netzwerkinfrastruktur führt.

Sofern ein Web-System über einen SSL-Kanal [13] zum Beispiel via

HTTPS [38] arbeitet, existiert noch eine weitere Möglichkeit. In diesen Fällen gibt es aufgrund des SSL-Handshakes bereits einen Sitzungsschlüssel, der für die Verbindung genutzt wird. Dieser Sitzungsschlüssel ist auch innerhalb der Sitzung von Seitenaufruf zu Seitenaufruf konstant und kann daher als Sitzungsidentifikationsmerkmal verwendet werden.

4.2.2 Bedeutung

Allen Techniken zur Identifikation von Sitzungen ist gemein, dass sie keine Authentifikation benötigen. Alle Verfahren setzen beim Betreten einer Web-Site einen geeigneten Schlüssel fest und nutzen geeignete Hilfskonstrukte, die der Client von Aufruf zu Aufruf beibehält. Sieht man einmal vom Messwert „Benutzer“ ab, so zeigt sich, dass die hier gewonnene Protokollinformation bereits hinreichend für die Berechnung der Messwerte ist, da für die Berechnung der Vorgänger- und der Nachfolger-Relation an die Stelle der *user*-Komponente einfach der hier ermittelte Sitzungsschlüssel tritt.

Neben der Tatsache, dass keine Benutzer identifiziert werden und damit beispielsweise auch keine Revisits erkannt werden, muss auch berücksichtigt werden, dass das aus Benutzersicht durchgängige Navigieren über externe Inhalte sich technisch durchaus in neuen Sitzungen niederschlagen und damit eine weitere Verzerrung der Messwerte entstehen kann.

4.3 Authentication-Tracking

Die dritte Möglichkeit Benutzerzuordnungen zu erzeugen besteht in der Möglichkeit, auf eine irgendwie geartete Benutzerauthentifikation zurückzugreifen und damit eine echte Benutzeridentifikation zu erhalten. Auf diese Weise kann sicherlich die höchste Präzision beim Zuordnen von Benutzern erlangt werden, da die Authentität der Benutzer durch den eingesetzten Mechanismus erzwungen wird.

Berechnung Es gibt verschiedene Verfahren zur Benutzerauthentifikation, die nicht alle an dieser Stelle aufgeführt werden können, da es nicht Gegenstand dieser Arbeit ist, authentisches Handeln sicherzustellen. Zum Verständnis seien daher an dieser Stelle nur drei grundlegende Arten vorgeführt, die die Prinzipien erläutern.

Basic and Digest Access Authentication Als erstes Verfahren sei auf eine Erweiterung des HTTP-Protokolls zur Authentifikation [17] hingewiesen. Die Erweiterung sieht vor, dass ein Benutzer sich, bezogen auf eine vorgegebene „Realm“ (hier zu Verstehen als der Name einer Anmeldungsdomäne), mit einem Login-Namen und einem Passwort anmeldet. Dazu wird vom Web-Server bei Client-Anfragen auf geschützte Bereiche eine spezielle HTTP-Response geschickt, die den Client zur Authentifikation an einer „Realm“ auffordert. Web-Browser zeigen dann normalerweise einen Popup-Dialog an, in dem der Benutzer zur Eingabe eines Login-Namens und eines Passworts aufgefordert wird. Der Server gibt zudem in seiner Response einen Gültigkeitsbereich (z.B. URI-Präfix) an, in dem der Client die gleichen Authentifaktionsdaten nutzen darf. Außerdem wird über die Server-Response festgelegt, wie die Passwort-Überprüfung passieren soll. Bei Nutzung von Basic Authentication schickt der Client ab sofort für jede Seite aus dem Gültigkeitsbereich in seinem Request Login und Passwort unverschlüsselt als speziellen HTTP-Header mit. Im Falle der Digest Access Authentication werden aus dem Passwort und weiteren Werten MD5-Summen ermittelt, die dem Server die Möglichkeit bieten sicherzustellen, dass der aktive Nutzer das Passwort kennt, ohne dass das Passwort selbst übertragen werden muss.

Die Kombination aus „Realm“ und Login-Name stellt schließlich ein eindeutiges Identifiaktionsmerkmal für einen Benutzer dar.

Für ein User-Tracking-System existieren mehrere Möglichkeiten, auf diese Information zuzugreifen. Neben der prinzipiellen Option, die Daten über eine

spezielle API von der Anwendung selbst mitgeteilt zu bekommen, kann das System sich in den HTTP-Kanal „einklinken“ und entsprechend die Information aus dem HTTP-Header herausfiltern. Außerdem protokollieren die meisten Web-Server die über HTTP-Authentifikation ermittelten Login-Daten in ihren Zugriffsprotokollen, welche ebenfalls von einem Mining-System ausgewertet werden können (vgl. hierzu Abschnitt 5.1.

Anwendungsbasierte Anmeldung Bei anwendungsbasierter Anmeldung findet für gewöhnlich eine Benutzeranmeldung auf Basis eines speziellen Eingabeformulars statt. In einem solchen Formular gibt der Benutzer die notwendigen Informationen an. Zwar werden hier in vielen Fällen ebenfalls Login-Namen und Passwörter abgefragt, dennoch ist dies keine notwendige Voraussetzung. So können, je nach Kontext, auch Email-Adressen, Kontonummern o.ä. zur Anmeldung genutzt werden.

Im Gegensatz zur HTTP-basierten Authentifikation wird die während der Anmeldung festgestellte Authentität in einer Server-Session gespeichert und steht dann der Anwendung bei Folgeaufrufen innerhalb der gleichen Sitzung [21] zur Verfügung. Für das User-Tracking muss zur Nutzung allerdings eine anwendungsproprietäre Schnittstelle geschaffen werden, die dem Tracking-System den Zugang zur Personenidentität ermöglicht.

Zertifikate Eine weitere Variante zur Identifiaktion von Benutzern ist die Verwendung von Client-Zertifikaten [18]. Dabei werden vom Client während dem Verbindungsaufbau SSL-Zertifikatsinformationen an den Server übertragen, die den tatsächlichen Benutzer authentisch identifizieren. Beispielsweise werden diese Zertifikats-Informationen im J2EE-Context [4] anhand einer entsprechenden „Realm“ verifiziert und eine Benutzerzuordnung hergestellt. Die ermittelte Authentität wird in der SSL-Session vom Server gespeichert und steht hier auch bei Folgeaufrufen zur Abfrage bereit.

Durch „Einklinken“ in den über SSL laufenden HTTP-Kanal kann die Benutzer-Authentität abgefragt und so die Zuordnung der Benutzeridentität erkannt werden.

4.3.1 Eigenschaften

„Authentication“-Tracking ist sicherlich das Identifikationsverfahren mit der höchsten Datenpräzision. Letztendlich wird durch eine Authentifikation die

im Internet herrschende Anonymität durchbrochen, wodurch Benutzer zugeordnet werden können. Leider ist Authentifikation nur für geschützte Bereiche, wie zum Beispiel ein Intranet, sinnvoll und hätte beispielsweise für das Besuchen einer Homepage eher abschreckende Wirkung. So erwartet ein Internetnutzer nicht beim Abfragen öffentlich zugänglicher Informationen, dass er seine Identität preisgeben muss. „Authentication"-Tracking ist daher nur in Ausnahmesituationen und nicht im Allgemeinen anwendbar.

4.3.2 Bedeutung

Sofern ein System Zugriff auf Authentifikationsinformationen besitzt, ist prinzipiell eine hohe Datenpräzision gewährleistet. Dabei muss jedoch berücksichtigt werden, dass es für jeden Benutzer eine Menge von Authentifikationsidentifikatoren geben kann. Es kann bespielsweise sein, dass die Identität eines Benutzers in unterschiedlichen Anmeldedomänen über unterschiedliche Login-Namen festgelegt ist. Sofern daher ein Usage-Mining-System Daten aus unterschiedlichen Anmeldedomänen verwaltet, müssen Techniken vorgesehen werden, die die Identifikatoren zusammenführen.

Im Falle der anwendungsbasierten Authentifikation sei angemerkt, dass hier, je nach eingesetzter Technik, auch Probleme des Session-Managements entstehen können. Dabei ist vor allem an Session-Hijacking- oder Copy-Paste-Probleme, wie sie bereits im Abschnitt „Session-Tracking" (vgl. Abschnitt 4.2) beschrieben wurden zu denken. Wenngleich es in diesen Fällen zu Mehrdeutigkeiten bei der Benutzerzuordnung kommen kann, ist dieses Problem sicherlich vernachlässigbar. Sofern die zu Grunde liegende Anwendung derartige Mehrdeutigkeiten ignoriert, sind sie fürs User-Tracking sicherlich ebenfalls unbedeutend.

Kapitel 5

Datenerfassung

In diesem Kapitel wird die Entstehung der Daten für das Zielsystem genauer betrachtet. Zur Erfassung der Daten gibt es vielfältige Methoden. Da mir jedoch keine Methode bekannt ist, die alle anderen Methoden im Hinblick auf Vollständigkeit und Datengenauigkeit in den Schatten stellt, möchte ich an dieser Stelle die Möglichkeit bieten, das System auf verschiedenen Wegen mit Daten zu "füttern", um auf diese Weise eine Präzision und Vollständigkeit zu erlangen, die keine alleinstehende Variante erreicht.

Um einen generischen und insbesondere auch erweiterbaren Ansatz für die Erfassung der Daten zu verfolgen, möchte ich von der späteren Aufbereitung der Daten an dieser Stelle abstrahieren. Im Zentrum steht die Erzeugung bzw. Anpassung eines Zielprotokolls $\mathcal{A} \subseteq \mathbb{A}$. Dabei werden die Implikationen der Verarbeitung von Datenquellen für dieses Zielprotokoll erarbeitet.

5.1 Web-Server-Protokolle

Bei der Konzeption des World-Wide-Web oder vielmehr der Implementierung der zu Grunde liegenden Web-Server-Software, wurde speziell darauf geachtet, technische Netzwerkengpässe und Fehler erkennbar zu machen. Zu diesem Zwecke wurde ein Protokollstandard (das "Common Log Format" s.u.) entworfen, der die Nutzung der Web-Server festhält und dadurch im Nachhinein die Möglichkeit bietet, Engpässe und Missstände aufzudecken. Dieser Protokollstandard wird heute von den meisten Web-Servern unterstützt und soll daher im Folgenden näher betrachtet werden.

5.1.1 Common Log Format (CLF)

Der CLF-Protokoll-Standard speichert Zugriffsinformationen in einer zeilenweisen Textdatei, wobei jede Datenzeile den folgenden Aufbau hat:

Syntax:

```
REMOTEHOST  RFC931  AUTHUSER  [DATE] "REQUEST"  STATUS  BYTES
```

Beispiel:

```
217.91.71.203  - - [30/Apr/2006:14:45:37 +0200]  "GET /media-
    server/index.html HTTP/1.0" 200 15364

217.91.71.203  - - [30/Apr/2006:14:45:59  +0200]  "GET /data-
    agent/index.html HTTP/1.0" 200 15716

72.30.110.32  -  -  [30/Apr/2006:14:46:53  +0200]  "GET /cms-
    ionas/index.html HTTP/1.0" 304 -
```

REMOTEHOST Hier wird, je nach Verfügbarkeit, der DNS-Name oder die IP-Adresse des entfernten Rechners aufgeführt, der eine Abfrage auf dem Server durchgeführt hat. An dieser Stelle sei angemerkt, dass hier lediglich die Adresse des Kommunikationspartners aufgeführt wird, der direkt mit dem

Web-Server in Verbindung steht, wobei sich dieser im Zeitalter des IPv4-Adressmangels häufig vom Arbeitsplatzrechner des End-Nutzers unterscheidet.

RFC931 Nach RFC-931 [25] wird in der zweiten Spalte eines CLF-Eintrags der Login-Name des Besitzers des TCP-IP/Port-Paares der Verbindung auf dem Web-Server hinterlegt. In vielen Situationen entspricht dieser Besitzer dem Systemnutzer, der den Web-Service zur Verfügung stellt. Da dieser Wert bei vielen Servern zum Startzeitpunkt des Web-Services selbst festgelegt wird und von dort an konstant ist, führen viele Web-Server an dieser Stelle nur ein „-"-Zeichen auf.

AUTHUSER Bei einer Verbindung mit HTTP-basierter Authentifizierung speichert das CLF in diesem Feld den Benutzernamen des authentifizierten Nutzers ab. Falls HTTP-basierte Authentifizierung nicht genutzt wird und somit anonym auf die Inhalte zugegriffen wird, wird dieses Feld lediglich mit einem "-"-Zeichen belegt.

DATE Dieses Feld gibt den Zeitpunkt an, zu dem der entsprechende Zugriff stattgefunden hat. Neben lokalem Datum und lokaler Uhrzeit des Zugriffs wird hier auch die Zeitzone angegeben, auf die sich die Werte beziehen.

REQUEST Nach RFC-1945 [7] bzw. RFC-2068 [16] besteht die erste Zeile einer HTTP-Anfrage aus der sogenannten Request-Line. Diese beginnt mit dem "Method-Token", gefolgt vom "Request-URI" und der genutzten HTTP-Protokollversion. Im CLF wird diese Zeile als weiterer Wert in einer Protokolldatei aufgenommen, um den Gegenstand des Zugriffs zu beschreiben.

STATUS Der Status-Wert gibt den Antwort-Status des Servers auf die Anfrage an. Dieser Wert kodiert den Erfolg bzw. Misserfolg der Anfrage. Dabei wird im Wesentlichen ausgedrückt, ob die Anfrage erfolgreich verarbeitet werden konnte oder nicht, bzw. ob das aufgerufene Ziel verschoben wurde. Nähere Informationen zu HTTP-Status-Codes können den RFCs RFC-1945 [7] und RFC-2068 [16] entnommen werden.

BYTES Ein letzter Wert aus dem CLF gibt die Anzahl der übertragenen Daten-Bytes an, die zurück an den Client gesendet wurden.

5.1.2 Extended Common Log Format (ECLF)

Bei der Analyse von Web-Protokollen erkannte man schnell, dass das CLF für die Erkennung von Fehlerursachen nicht hilfreich war. Zwar konnte man in den so gespeicherten Informationen feststellen, welche Inhalte aufgerufen wurden und wo es zu Fehlern (vgl. Spalte „STATUS“) kam, die Ursachen von Fehlern blieben jedoch meist unerkannt. So wurde zwar erkannt, dass ein aufgerufener URL nicht vom Web-Server angemessen ausgewertet werden konnte, nicht jedoch, wie dieser URL zu Stande kam. Ziel des Extended Common Log Formats (ECLF) war es daher, diesem Missstand Abhilfe zu schaffen. Dazu wurde das CLF um zwei weitere Aspekte “REFERER“ und “USERAGENT“ erweitert, die Hinweise auf den Ursprung von Fehlern geben sollten:

```
REMOTEHOST  RFC931 AUTHUSER [DATE] "REQUEST" STATUS BYTES "RE-
    FERER" "USERAGENT"
```

Beispiel:

```
217.91.71.203  - - [30/Apr/2006:14:45:37  +0200] "GET /media-
    server/index.html HTTP/1.0" 200 15364 "http://www.chamae-
    leon.de/sitemap_und_suche/index.html"  "Mozilla/5.0 (com-
    patible;  heritrix/1.6.0  +http://www.schluetersche.de)"

217.91.71.203  - - [30/Apr/2006:14:45:59  +0200]  "GET /data-
    agent/index.html  HTTP/1.0" 200 15716 "http://www.chamae-
    leon.de/sitemap_und_suche/index.html"  "Mozilla/5.0 (com-
    patible;  heritrix/1.6.0  +http://www.schluetersche.de)"

72.30.110.32  -  - [30/Apr/2006:14:46:53  +0200]  "GET  /cms-
    ionas/index.html  HTTP/1.0" 304 - "-"  "Mozilla/5.0 (com-
    patible;  Yahoo!  Slurp; http://help.yahoo.com/help/us/y-
    search/slurp)"
```

REFERER Um Verlinkungsfehler in Präsenzen ausfindig zu machen wurde das CLF um ein Feld erweitert, welches die URL der Seite aufführt, die den Aufruf der aktuellen Seite bewirkt hat.

USERAGENT Dies ist eine Bezeichnung, die die Client-Software für gewöhnlich mitsendet, um erkennbar zu machen, welche Software beim Betrachten der Seite verwendet wird.

5.1.3 Proprietäre Log-Formate

Während gängige Web-Server-Software (wie "Apache HTTP Server Project", "Microsoft Internet Information Server" u.a.m.) von Grund auf die Erzeugung von Zugriffsprotokollen im CLF oder im ECLF unterstützen [30], so sind weitere Protokolle denkbar. Da ich an dieser Stelle jedoch generische Web-Server-Software und keine anwendungsproprietären Systeme betrachte sei angemerkt, dass die Menge der protokollierbaren Informationen von vornherein auf die Informationen des HTTP-Protokolls [16, 7] sowie die Informationen der darunterliegenden Netzwerkschichten begrenzt ist. Da die zentralen Informationen dieser Schichten bereits vom ECLF erfasst werden, möchte ich an dieser Stelle für die Verwendung proprietärer Log-Formate voraussetzen, dass diese in CLF oder ECLF transformierbar sind.

5.1.4 Eigenschaften

Das zentrale Anliegen des hier vorgestellten Datenerfassungsverfahrens ist es, die notwendigen Informationen zum Usage-Mining einzig durch die Auswertung bereits existierender Informationen zu gewinnen.

Per Definition handelt es sich dabei um ein Vorgehen, welches vollständig abgekoppelt vom eigentlichen Anwendungssystem arbeitet und lediglich die ohnehin gesammelten Daten interpretiert.

Im Zuge eines High-Traffic-Anwendungssystems bedeutet dies, dass hier so gut wie keine zusätzliche Systemlast das Verhalten des High-Traffic-Systems beeinträchtigt, da die Auswertung der gesammelten Daten zum Beispiel auf einer losgelösten Umgebung passieren kann.

Da zudem keinerlei Anpassungen des Anwendungssystems notwendig sind, kommt es ebensowenig zu Seiteneffekten, was die Funktionalität des Systems

aus Nutzersicht betrifft. Es gibt demnach auch keinerlei zusätzliche Anforderungen für die vom Client genutzte Software, damit das Usage-Mining-System funktionieren kann.

Es sei jedoch angemerkt, dass das World Wide Web die ursprüngliche Aufgabe hatte, Informationen (Dokumente) weltweit und verteilt zugänglich zu machen. Als Online Retrieval System war das WWW demnach nie konzipiert. [31] Folglich haben die Logfiles die primäre Aufgabe, die Zugriffe auf diese weltweit verteilten Dokumente aufzuzeichnen, um Kommunikationsschwachstellen und technische Fehler aufzudecken (z.B. Verlinkungsfehler). Das Erzeugen von Benutzerprofilen war für den Entwurf der Protokolldaten zu keinem Zeitpunkt von Bedeutung. So sind beispielsweise Verweilzeiten nicht Bestandteil der Protokollinformation und können nur indirekt heuristisch ermittelt werden.

Außerdem sei angemerkt, dass Web-Server-Protokolle immer die Sicht des jeweiligen Web-Servers darstellen und nicht die Sicht der nutzenden Kunden bzw. des gesamten Anwendungssystems. Gerade im Zeitalter des IPv4-Adressraums [36] kommt es hier häufig zur Verfälschung der Informationen. So sind die Einträge, die sich in den Logfiles befinden, als Einträge zu verstehen, die von virtuellen Benutzern oder anders ausgedrückt von vernetzten Computern stammen. Die Folge ist, dass sich die Aktionen und damit die entstehenden Einträge dieser virtuellen Benutzer in der Regel nicht direkt auf eine bestimmte Person beziehen lassen. Das liegt daran, dass, wie bereits im Abschnitt 4.3 (siehe Seite 50) erwähnt, Authentifikationsdaten selten im Logfile zur Verfügung stehen und somit zur Identifikation von Benutzern lediglich die Internetadresse (IP-Adresse) des Computers übrigbleibt [31].

Es fehlen aber nicht nur spezielle Informationsfelder, vielmehr sind die Datensätze zum Teil auch unvollständig. So führt der Einsatz von Web-Proxy-Servern [14] zum Fehlen einzelner Datensätze im Zugriffsprotokoll, da entsprechende Anfragen den Server nicht erreichen [32].

Die Datensätze sind zudem mehrdeutig, da sie transaktionsorientiert den Zugriff auf eine einzelne Resource aufzeichnen und nicht ganze Benutzersitzungen zusammenhängend erfassen [32].

5.1.5 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$

Beim Lesen eines Eintrags aus dem CLF wissen wir, welches Objekt wann über welche Peer-Adresse aufgerufen wurde. Wir können allerdings nicht ge-

nau sagen, wann der Übergang zum aktuellen Objekt eingeleitet wurde, da der Server den Zeitpunkt zu dem er eine Antwort an den jeweiligen Client gesendet hat und nicht den Zeitpunkt, an dem der Benutzer den Aufruf der Seite eingeleitet hat, festhält. Prinzipiell bestimmt aber jeder Eintrag eines CLF-Protokolls $\mathcal{C}$ das Zielprotokoll $\mathcal{A} \subseteq \mathbb{A}$ wie folgt:

$$\begin{aligned} \forall c \in \mathcal{C} \Rightarrow \exists a \in \mathcal{A} \ : \quad & a[object] = objectOf(c[REQUEST - URL]) \\ \wedge \quad & a[address] = c[REMOTEHOST] \\ \wedge \quad & a[access] \geq c[DATE] - \tau \\ \wedge \quad & a[access] \leq c[DATE] \\ \wedge \quad & (c[AUTHUSER] \in \{a[user], \epsilon\}) \end{aligned} \tag{5.1}$$

Diese Eigenschaft möchte ich im Folgenden als CLF-Korrespondenzeigenschaft mit dem Symbol $\overset{clf}{\models}$ bezeichnen. Ist daher $\mathbb{C}$ die Menge aller Log-Einträge im Common Log Format, so gilt:

$$\begin{aligned} \forall c \in \mathbb{C}, \forall a \in \mathbb{A} \ : \quad & c \overset{clf}{\models} a \\ \Leftrightarrow \quad & a[object] = objectOf(c[REQUEST - URL]) \\ \wedge \quad & a[address] = c[REMOTEHOST] \\ \wedge \quad & a[access] \geq c[DATE] - \tau \\ \wedge \quad & a[access] \leq c[DATE] \\ \wedge \quad & (c[AUTHUSER] \in \{a[user], \epsilon\}) \end{aligned} \tag{5.2}$$

Eine weitere Eigenschaft von CLF-Protokolleinträgen ist, dass sie einen Teil aller tatsächlich durchgeführten Seitenaufrufe abbilden. Es gibt also zu jedem Eintrag im CLF-Protokoll mindestens einen Eintrag im Zielprotokoll, sodass es immer mindestens so viele Korrespondenz-Einträge geben muss, wie Einträge im CLF-Protokoll. Ist daher $\mathcal{C} \subseteq \mathbb{C}$ ein beliebiges CLF-Protokoll, so gilt für das resultierende Zielprotokoll $\mathcal{A} \subseteq \mathbb{A}$:

$$\forall \mathcal{C}' \in P(\mathcal{C}) : |\mathcal{C}'| \leq |\{a \in \mathcal{A} | \exists c \in \mathcal{C}' : c \overset{clf}{\models} a\}| \tag{5.3}$$

Aus Einträgen im ECLF können wir zum Einen eine nähere Einschränkung bei der Benutzerzuordnung vornehmen, aber auch Aussagen über die Objekte treffen, von denen aus auf das aktuelle Objekt zugegriffen wurde. Die Menge der Einträge eines ECLF-Protokolls $\mathcal{E} \subseteq \mathbb{E}$ bestimmen das Zielprotokoll $\mathcal{A} \subseteq$

$\mathbb{A}$ wie folgt:

$$\begin{aligned}\forall e \in \mathcal{E} \Rightarrow \exists a_c, a_r \in \mathcal{A} \ : \quad & a_c[object] = objectOf(e[REQUEST - URL]) \\ \wedge \quad & a_c[address] = e[REMOTEHOST] \\ \wedge \quad & a_c[agent] = e[USERAGENT] \\ \wedge \quad & a_c[access] \geq e[DATE] - \tau \\ \wedge \quad & a_c[access] \leq e[DATE] \\ \wedge \quad & (e[AUTHUSER] \in \{a_c[user], \epsilon\}) \\ \wedge \quad & (a_r \prec a_e \vee a[referer] = \epsilon)\end{aligned} \tag{5.4}$$

Diese Eigenschaft möchte ich im Folgenden als CLF-Korrespondenzeigenschaft bezeichnen. Dabei sei angemerkt, dass sich die ECLF-Korrespondenz immer auf ein konkretes Zielprotokoll $\mathcal{A} \subseteq \mathbb{A}$ bezieht. Ich werde die Beziehung daher im Folgenden durch das Symbol $\overset{eclf}{\models}_{\mathcal{A}}$ darstellen.

Ist $\mathbb{E}$ die Menge aller Log-Einträge im Extended Common Log Format, so gilt die ECLF-Korrespondenzeigenschaft in Bezug auf ein konkretes Zielprotokoll $\mathcal{A} \subseteq \mathbb{A}$ wie folgt:

$$\begin{aligned}\forall e \in \mathbb{E}, \forall a \in \mathbb{A} \ : \quad & e \overset{eclf}{\models}_{\mathcal{A}} a \\ \Leftrightarrow \quad & a[object] = objectOf(e[REQUEST - URL]) \\ \wedge \quad & a[address] = e[REMOTEHOST] \\ \wedge \quad & a[agent] = e[USERAGENT] \\ \wedge \quad & a[access] \geq e[DATE] - \tau \\ \wedge \quad & a[access] \leq e[DATE] \\ \wedge \quad & (e[AUTHUSER] \in \{a[user], \epsilon\}) \\ \wedge \quad & (a[referer] = \epsilon \vee (\exists a_r \in \mathcal{A} : a_r \prec a))\end{aligned} \tag{5.5}$$

Wie auch bei CLF-Protokollen werden bei ECLF-Protokollen Teile der tatsächlich durchgeführten Seitenaufrufe abgebildet. Es gibt also zu jedem Eintrag im ECLF-Protokoll mindestens einen Eintrag im Zielprotokoll, sodass es immer mindestens so viele Korrespondenzeinträge geben muss, wie Einträge im ECLF-Protokoll. Ist daher $\mathcal{E} \subseteq \mathbb{E}$ ein beliebiges ECLF-Protokoll so gilt für das resultierende Zielprotokoll $\mathcal{A} \subseteq \mathbb{A}$:

$$\forall \mathcal{E}' \in P/\mathcal{E}) : |\mathcal{E}'| \leq |\{a \in \mathcal{A} | \exists e \in \mathcal{E}' : e \overset{eclf}{\models}_{\mathcal{A}} a\}| \tag{5.6}$$

5.1.6 Bedeutung

Bei der Anwendung von Web-Server-Protokollen auf das hier zum Einsatz kommende Tracking-Modell, werden zahlreiche Schwächen deutlich.

Ein erstes Problem ist wohl darin zu sehen, dass Web-Server-Protokolle immer die Sicht des jeweiligen Web-Servers repräsentieren und nicht die Sicht der tatsächlichen Endnutzer. Daher kommt es leicht vor, dass von Benutzern durchgeführte Seitenanfragen nicht aufgeführt werden, da diese von im Netzwerk liegenden Caching-Systemen (z.B. Proxy-Servern) bereits vorab ausgewertet wurden [32].

In Verbindung mit Proxy-Servern kommt zudem eine zusätzliche Ungenauigkeit in den Daten zu Stande. Aus Sicht des Web-Servers kommen alle Anfragen, die unterschiedliche Benutzer über denselben Proxy-Server gestellt haben, von diesem einen Proxy-Server, sodass bei nichtauthentifizierter Nutzung kein Unterscheidungsmerkmal existiert, um zusammengehörige Aktionen zu identifizieren. Bei der Interpretation der Daten muss daher aufgepasst werden, dass keine unsinnigen Seitenwechsel oder Seitenaufrufe mit viel zu kurzen Verweilzeiten [32] entstehen.

Sofern eine Site mit Framesets arbeitet, sind erfasste Klickströme auch eher technischer Natur: Gibt es beispielsweise ein zweigeteiltes Frameset mit einem Navigationsmenü und einem Inhaltsbereich, so wird im ECLF meist das Navigationsmenü als *REFERER* aufgeführt und aus dem (aus Benutzersicht) gegangenen Navigationsweg wird ein sternartiges Gebilde, in dessen Zentrum das Navigationsmenü steht.

Im Sinne des hier angewendeten Logging-Modells liegt das Hauptproblem bei der Anwendung von Log-Dateien allerdings darin, dass *exit*-Werte nicht bestimmbar sind. Für den Sonderfall, dass eine Seite als Referer einer anderen Seite angegeben ist, kann zwar erkannt werden, dass die referenzierende Seite bis zu diesem Zeitpunkt noch aktiv war, allerdings kann keine obere Schranke für den *exit*-Wert eines Zugriffs angegeben werden. Im ungünstigsten Falle können, bedingt durch Caching, extrem lange Verweilzeiten entstehen, wenn ein Besucher eine Seite aufruft, zu einem wesentlich späteren Zeitpunkt diese Seite bei erneutem Aufruf aus einem Cache geladen wird und von dieser gepufferten Seite ausgehend weiternavigiert. In diesem Falle bekommt der Web-Server den via Cache aufgelösten Aufruf nicht mit und wertet einen entsprechenden *Referer*-Wert so, als ob der Benutzer die Ursprungsseite seit ihrem Erstaufruf aktiv hat.

Dennoch besitzen Web-Log-Protokolle einen unschlagbaren Vorteil, gegenüber allen anderen Techniken: "Sie existieren!" und zwar ohne dass weitere Anpassungen eines Systems notwendig sind und ohne dass weitere Anforderungen an Clients (wie z.B. das Aktivieren von Cookies oder JavaScript) gestellt werden müssen. Sie liefern damit Fall-Back-Informationen in Situationen, in denen andere Verfahren scheitern.

5.2 Redirector

Eines der prinzipiellen Probleme von Web-Protokollen ist, dass nicht erkannt werden kann, wann und wohin eine Site verlassen wurde. Die „Redirector"-Methode greift hier ein und bietet eine Möglichkeit, Übergänge zu externen Inhalten zu kontrollieren.

Die grundlegende Idee der „Redirector"-Technik ist es, Links auf externe Inhalte immer über eine dazwischengeschaltete Seite laufen zu lassen, die selbst eine automatische Weiterleitung auf den tatsächlichen externen Inhalt durchführt. Da diese „Stellvertreterseite" unter eigener Kontrolle liegt und vom Web-Server-Protokoll erfasst wird, kann so erkannt werden, dass ein Benutzer einen Link auf einen speziellen externen Inhalt aufgerufen hat.

5.2.1 Clientseitige Weiterleitung

Hat die „Chamaeleon AG" beispielsweise an einer bestimmten Stelle in ihrer Homepage einen Link auf „http://www.uni-koblenz.de" geschaltet, so kann sie stattdessen die folgende HTML-Seite auf ihrem Server ablegen und alle Verlinkungen, die zuvor auf die Universitätsseite gezeigt haben, entsprechend auf diese neue Seite umbiegen:

```
1  <html>
2      <head>
3          <title>
4              Weiterleitung zur Universität Koblenz
5          </title>
6          <meta http-equiv="Refresh"
7                content="1;url=http://www.uni-koblenz.de">
8      </head>
9      <body>
10         Hinweis: Sie haben einen Verweis zur Universität
11         Koblenz aufgerufen. Wenn Sie nicht automatisch
12         weitergeleitet werden klicken Sie bitte
13           <a href="http://www.uni-koblenz.de">hier</a>.
14     </body>
15 </html>
```

Da diese Seite auf dem lokalen Server der „Chamaeleon AG" liegt, kann anhand des Web-Server-Protokolls erkannt werden, wann diese Seite und damit ein Verweis zur „Universität Koblenz" aufgerufen wurde. Der „Refresh"-Meta-

Tag sorgt beim Aufrufen der Seite zudem dafür, dass der Browser zur Universitätsseite weiterspringt. Dazu muss die entsprechende Client-Software allerdings so eingestellt sein, dass diese speziellen Meta-Refresh-Tags ausgewertet werden. Um Probleme wie „Session-Hijacking“ [11] zu umgehen, deaktivieren jedoch viele Internetnutzer diese Funktion, sodass für diese Benutzer die Weiterleitung schließlich nicht mehr transparent ist.

5.2.2 HTTP-Weiterleitung

Eine weitere Variante ist die Verwendung von HTTP-Weiterleitungen [16, 7]. In gängiger Web-Server-Software können dazu direkt im Web-Server spezielle Weiterleitungen vergeben werden. Dabei werden einfach für bestimmte lokal aufgelöste URLs festgelegt, dass diese auf externe Seiten umgebogen werden sollen.

Sobald man allerdings eine größere Site verwalten muss wird es schnell sehr unhandlich für sämtliche externen Seiten zusätzliche lokale Weiterleitungen im Web-Server zu registrieren. In diesen Fällen bietet es sich an, einfach ein kleines Programm zu installieren, welches eine Weiterleitung auf eine beliebige Seite durchführt. Dazu kann beispielsweise der folgende in Perl geschriebene Ein-Zeiler auf dem Server hinterlegt werden (vgl. [20]), welcher diese Weiterleitung vornimmt:

```
1 print "Location: $ENV{QUERY_STRING}";
```

Der Web-Server transformiert eine derartige Ausgabe schließlich in eine entsprechende HTTP-Response, die direkt eine HTTP-basierte Weiterleitung durchführt. Der Vorteil dieser Variante ist, dass HTTP-Weiterleitungen eine prinzipielle Protokollfunktion darstellen und von allen gängigen Web-Clients verarbeitet werden können. Da die Sicherheitslücken der clientseitigen Weiterleitung prinzipiell auf „Code-Injection„ basieren, ist das Risiko bei HTTP-basierter Weiterleitung vergleichsweise minimal. Das Verfahren funktioniert daher auch in restriktiv eingestellten Clients.

Nehmen wir an, dieses Script wird auf dem Server so hinterlegt, dass es über „/redirect.cgi“ verlinkt werden kann. Dann reicht es aus, in der Site alle Links auf „http://www.uni-koblenz.de“ durch „/redirect.cgi?http://www.uni-koblenz.de“ zu ersetzen, um diese Übergänge zu erfassen.

5.2.3 Quell-Ziel-Analyse mit Redirectoren

Mit der „Redirector“-Technik können aber auch weitere Informationen über den Kontext eines Linkaufrufs geboten werden. Beispielsweise kann man bei jedem „Redirector“-Aufruf Informationen über den Ursprung eines Links angeben, sodass man bei Auswertungen nicht auf “Referer“-Informationen angewiesen ist. Statt nur einen Link auf das Weiterleitungsziel anzugeben, kann man gleichermaßen einen CGI-Parameter für den Ursprung angeben. Das im Folgenden dargestellte Perl-Script ermöglicht beispielsweise die Übergabe eines SOURCE- und eines TARGET-CGI-Parameters

```
1 use CGI;
2 $source := CGI->param("SOURCE");
3 $target := CGI->param("TARGET");
4 print "Location: $target\n\n";
```

Um nun von einer “Impressum“-Seite aus auf die Startseite der Universität Koblenz zu verweisen könnte man den folgenden Link nutzen:

```
1 <a href="/redirect.cgi?SOURCE=/impressum&amp;TARGET=http://
2 www.uni-koblenz.de">
3     [klicken Sie hier]
4 </a>
```

5.2.4 „Redirector“-Script für ECLF-Erzeugung

Die „Redirector“-Technik könnte theoretisch auch eingesetzt werden, wenn ein Web-Server keine Zugriffsprotokollierung durchführt oder die entsprechenden Protokolle nicht zur Verfügung stehen. In diesen Fällen muss lediglich das Redirector-Script so angepasst werden, dass es selbst eine Protokollierung durchführt. Da das CGI-Interface [20] bereits alle fürs ECLF notwendigen Informationen bereitstellt [33], können die Einträge wie folgt erzeugt werden:

```
1 #!/usr/bin/perl
2 use CGI;
3 use English;
4
5 my ($RFC931) = getpwnam($UID);
6
7 my $ECLF_ENTRY =
8           # Adresse des entfernten Rechners:
9           $ENV{REMOTE_ADDR}
```

```
10
11          # Besitzer des lokalen TCP-Ports
12          . "␣$RFC931"
13
14          # Login-Name des angemeldeten Benutzers
15          . "␣$ENV{REMOTE_USER}"
16
17          # Zeitpunkt des Zugriffs
18          . "␣[" . localtime() ."]"
19
20          # HTTP-Request
21          . "␣\"$ENV{REQUEST_METHOD}␣"
22             . CGI->param("TARGET")
23             . "␣$ENV{SERVER_PROTOCOL}\""
24
25          # HTTP-Status und Größe des Aufrufziels
26          # ist unbekannt, wird aber fürs Usage-Mining
27          # nicht benötigt. Von daher können hier
28          # Dummy-Werte eingesetzt werden
29          . "␣200␣?"
30
31          # Referer
32          "␣\"" . CGI-param("SOURCE") . "\""
33
34          # Name der Client-Software
35          "␣\"$ENV{USER_AGENT}\""
36      ;
```

In der Site muss dann jeder Link auf dieses „Redirector“-Script umgebogen werden, damit entsprechende Seitenaufrufe erfasst werden. Die alleinige Nutzung der „Redirector“-Technik würde jedoch dazu führen, dass die Eintrittspunkte in eine Präsenz nicht erkannt werden, da die Stellvertreterseiten für gewöhnlich nur innerhalb der Site genutzt werden und entsprechend erst beim Aufruf einer weiteren Seite aus der Site Protokollinformationen sammeln.

5.2.5 Eigenschaften

Die „Redirector“-Technik stellt lediglich eine Erweiterung zur Analyse von Web-Protokollen dar und ist daher in Bezug auf Web-Caches gleichermaßen fehleranfällig. Außerdem muss berücksichtigt werden, dass in High-Traffic-Umgebungen zusätzliche Stellvertreterseiten auch eine zusätzliche Belastung

für den jeweiligen Web-Server darstellen. Von den obigen CGI-Script-Beispielen ist daher in einer High-Traffic-Umgebung abzuraten, da jeder Link-Aufruf einen Server-Prozess, in diesem Falle mit einem Perl-Interpreter, starten würde, was leicht zu einer Überbelastung des Systems werden kann. Zum Glück gibt es zahlreiche Methoden, die es gestatten, derartige Server-Prozesse wieder zu nutzen und dadurch die Einsetzbarkeit in High-Traffic-Umgebungen ermöglichen. Als Beispiel sei an dieser Stelle auf FastCGI [35], ModPERL [5] oder JSP [4] verwiesen, über die es möglich ist, leichtgewichtige, ergebnisäquvalente Implementierungen zu liefern.

Es sei jedoch darauf hingewiesen, dass im Allgemeinen nur HTTP-Get-Re quests auf diese Weise abgehorcht werden können. Das liegt daran, dass in einem Redirect keine größeren Datenmengen mitgereicht werden können und somit zum Beispiel die Daten komplexer Post-Anfragen verloren gingen.

5.2.6 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$

Die „Redirector"-Technik versucht speziell das Problem in den Griff zu bekommen, Übergänge auf externe Inhalte bzw. Endpunkte einer Site zu erfassen. Dabei emuliert es diese Übergänge aus Sicht des Web-Servers durch spezielle Weiterleitungsseiten auf dem Web-Server, die ihrerseits vom Zugriffsprotokoll des Web-Servers erfasst werden. Da Auswertungen anschließend auf diesem Protokoll stattfinden, gelten für die Implikationen die gleichen Aussagen, die bereits im vorangegangen Abschnitt beschrieben wurden. Zusätzlich werden jedoch Übergänge zu externen und speziellen internen Seiten analysierbar.

5.2.7 Bedeutung

Die Technik kann nicht dazu beitragen, Probleme zu schmälern, die durch Web-Caches, Gateways oder auch Framesets entstehen. Lediglich die Erfassung von Ausgängen aus der Präsenz wird durch diese Technik ermöglicht.

Damit die Technik jedoch eingesetzt werden kann, müssen im überwachten System sämtliche Querverweise auf externe Seiten durch Verweise auf entsprechende Stellvertreterseiten umgebogen werden. Während dies für einfache Links i.d.R. kein Problem ist, ist dies bei komplexeren Formularen nur unter zu Hilfenahme zusätzlicher Technologie (z.B. Javascript) möglich. Dies kann jedoch die ursprüngliche Systemfunktionalität beeinträchtigen oder so-

gar gänzlich unbrauchbar machen (man denke an Anforderungen zur Barrierefreiheit [8]).

5.3 Ping á la WhatWG

Eine Alternative zur „Redirector“-Technik bildet das von der WhatWG (www.whatwg.org) spezifizierte ping-Attribut [22] für Hyperlinks. Dabei wird bei der Verlinkung von Inhalten direkt die Möglichkeit gegeben, zusätzliche URLs beim Aufruf des jeweiligen Links aufzurufen.

5.3.1 Benachrichtigung via Ping

Prinzipiell können durch die Ping-Technik in bestimmten Situationen Benachrichtigungssignale in Form von HTTP-Aufrufen erzeugt werden. Dabei ist die Option gegeben, Signale beim Anklicken von Verweisen zu versenden. Es ist daher möglich, die „Redirector“-Technik zu substituieren. Hat man beispielsweise zwei Scripte, von denen das erste („redirect.cgi“) den übergebenen Query-String protokolliert und den Benutzer entsprechend weiterleitet und von denen das zweite („register.cgi“) lediglich den Query-String protokolliert, so liefern die beiden folgenden HTML-Befehle das gleiche Ergebnis:

```
1  <a href="`/redirect.cgi?http://www.uni-koblenz.de"'>
2      Link zur Uni-Koblenz über Redirector-Script
3  </a>
4
5  <a ping="`/register.cgi?http://www.uni-koblenz.de"'
6      href="`http://www.uni-koblenz.de"'>
7      Link zur Uni-Koblenz mit ping-basierter
8      Registrierung
9  </a>
```

5.3.2 Eigenschaften

Gegenüber der „Redirect“-Technik besitzt die „Ping“-Technik den Vorteil, dass Benutzer nicht über irgendwelche zwischenliegenden Seiten geleitet werden und die Übertragung der Zusatzsignale im Hintergrund passieren kann. Es ist daher auch möglich, die „Ping“-Technik mit HTTP-Post-Anfragen zu kombinieren.

Außerdem können Benutzer direkt die finale Ziel-URL sehen und eine Client-Software kann den Benutzer über das Versenden entsprechender Zusatzinformationen informieren. „Paranoide“ Benutzer können diese Funktionalität

deaktivieren, ohne dass das eigentliche Anwendungssystem an Funktionalität verliert [22].

Der größte Nachteil an der „Ping"-Technik gegenüber der „Redirector"-Technik ist jedoch, dass sie noch in den Kinderschuhen steckt und dass der Mechanismus erst von wenigen Browsern unterstützt wird. Insbesondere ist zum jetzigen Zeitpunk noch unklar, welche Browserhersteller diesen Befehl überhaupt und vor allem wann implementieren [9].

5.3.3 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$

Wie auch die „Redirect"-Technik überwacht die „Ping"-Technik den Aufruf von Verweisen. Aus Usage-Mining-Sicht können auf diese Weise die gleichen Informationen gesammelt werden, die auch die „Redirect"-Technik liefert.

5.4 Cookie-Tracking

Eine zusätzliche kleine Erweiterung für Log-basiertes User-Tracking liegt in der Verwendung von Cookies. Dabei wird Benutzern beim Betreten einer Site ein Sitzungs-Cookie zugewiesen, den der Client an den Server überträgt. Auf diese Weise ist es im Folgenden besser möglich, zusammenhängende Aktionen zu erkennen.

5.4.1 Eigenschaften

Da Cookies i.d.R. über den HTTP-Header von Anfragen und Antworten übertragen werden, beeinträchtigen sie das ursprüngliche Systemverhalten nur geringfügig. Sie können daher leicht eingesetzt werden. Ihre Verwendung setzt allerdings voraus, dass Browser diese auch auswerten und entsprechend verarbeiten. Ein User-Tracking-System kann sich daher nicht darauf verlassen, dass diese Informationen auch wirklich zur Verfügung stehen. Vielmehr muss es in der Lage sein, sinnvolle Analysen zu ermöglichen, ohne auf Cookies angewiesen zu sein. Sofern sie verfügbar sind, können sie aber einfach fürs Usage-Mining als zusätzliche Spalte in einem Log aufgenommen werden.

5.4.2 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$

Im Normalfall kann davon ausgegangen werden, dass innerhalb einer Sitzung nicht mehrere Personen aktiv sind. Man kann daher beim Erzeugen eines Sitzungscookies einen beliebigen eindeutigen Wert als Sitzungs-Cookie setzen und diesen im Folgenden beispielsweise in einem erweiterten ECLF mit protokollieren. Für den Fall, dass dieser Wert entsprechend vom Client zurückgesendet wird, kann man die folgende zentrale Folgerungsbeziehung anwenden:

$$\begin{aligned} \forall a_1, a_2 \in \mathcal{A} \quad : \quad & \epsilon \neq a_1[cookie] \\ \wedge \quad & a_1[cookie] = a_2[cookie] \\ \Rightarrow \quad & a_1[user] = a_2[user] \end{aligned} \tag{5.7}$$

Die ECLF-Korrespondenz erweitert sich dann zu:

$$\begin{aligned}\forall e \in \mathcal{E} \Rightarrow \exists a_c, a_r \in \mathcal{A} \quad : \quad & a_c[object] = objectOf(e[REQUEST - URL]) \\ \wedge \quad & a_c[address] = e[REMOTEHOST] \\ \wedge \quad & a_c[agent] = e[USERAGENT] \\ \wedge \quad & a_c[cookie] = e[COOKIE] \\ \wedge \quad & a_c[access] \geq e[DATE] - \tau \\ \wedge \quad & a_c[access] \leq e[DATE] \\ \wedge \quad & (e[AUTHUSER] \in \{a_c[user], \epsilon\}) \\ \wedge \quad & (a_r \prec a_e \vee a[referer] = \epsilon)\end{aligned} \tag{5.8}$$

5.4.3 Bedeutung

Durch Cookies ist es also möglich nähere Informationen über die Zusammengehörigkeit von Aktionen zu gewinnen. Wie auch schon in Abschnitt 4.2 (siehe Seite 47) angesprochen ist es jedoch nicht möglich ein authentisches Benutzer-Tracking auf Basis von Cookies durchzuführen. Es passiert leider recht häufig, dass Benutzer-Cookies zurücksetzen oder auch ihre Verwendung komplett deaktivieren. In diesen Situationen müssen Zusammengehörigkeiten auf anderer Ebene gefunden werden.

5.5 Snap-Shot-Technik

Die bislang betrachteten Techniken zur Datenerfassung orientierten sich an dem Prinzip, auf dem Server ankommende Anfragen bzw. Seitenaufrufe zu überwachen. Einen gänzlich anderen Weg schlägt das nachfolgend vorgestellte Verfahren ein, welches ich als „Snap-Shot"-Technik bezeichnen möchte.

Bei diesem Verfahren wird versucht, browserseitig Zustandsberichte zu generieren, die an ein Mining-System übertragen werden und ereignisgesteuert oder auch in diskreten Zeitabständen den Zustand des jeweiligen Browsers beschreiben.

5.5.1

Um das Verfahren einzusetzen muss ein kleines Programm geschrieben werden, welches im Browser des Endnutzers ausgeführt werden kann. Zur Implementierung kann dabei auf Java-Applets oder Javascript [23, 19] zurückgegriffen werden. Eine Implementierung in Javascript kann beispielsweise wie folgt aussehen:

```
1  /*
2   * Die Snap-Shot-Technik verwendet ein Zeitintervall, in
3   * dem automatisch ein neuer Snap-Shot übertragen wird.
4   * Im Beispiel wird einfach im Minutentakt eine auto-
5   * matische Übertragung durchgeführt.
6   */
7  var AUTO_NOTIFY_INTERVAL = 60000;
8
9  /*
10  * Für die präzise Datenerfassung wird in jedem Fenster
11  * der Zeitpunkt festgehalten, an dem der gegenwärtige
12  * Inhalt geladen wurde.
13  */
14 var window.LOAD_TIMESTAMP = (new Date()).getTime();
15
16 /*
17  * Da jeder Snap-Shot einen kompletten Bericht über alle
18  * Frames eines Fensters darstellt, reicht es aus, wenn
19  * innerhalb eines Fensters (frameunabhängig) nur ein
20  * Snap-Shot-Manager läuft. Da die "top"-Variable immer
21  * das oberste Fenster-Objekt (oder vielmehr Frame-
```

```
22   * Objekt) referenziert, muss nur auf oberster Ebene ein
23   * Snap-Shot-Manager eingerichtet werden.
24   */
25  if (! top.SnapShotManager) {
26      top.SnapShotManager = new Object();
27
28      /*
29       * Die folgenden beiden Member-Attribute legen den
30       * Kontext fest, in dem der Snap-Shot-Manager arbei-
31       * tet. Die "window"-Variable gibt dabei das Fenster
32       * (bzw. Frame) an, an dem der Snap-Shot-Manager beim
33       * Erzeugen eines Snap-Shots beginnt.
34       */
35      top.SnapShotManager.window = top;
36
37      /*
38       * Für die spätere Verarbeitung ist es notwendig,
39       * dass jedes Fenster eine eindeutige ID zugewiesen
40       * bekommt. In einer realen Implementierung sollte
41       * hier ein Verfahren zur Generierung einer UUID
42       * zum Einsatz kommen. Im Internet finden sich zahl-
43       * reiche Strategien zum Erzeugen derartiger Werte
44       * in Javascript, daher möchte ich an dieser Stelle
45       * einfach einen derartigen UUID-Generator voraus-
46       * setzen:
47       */
48      top.SnapShotManager.windowId = UUID.nextId();
49
50      /*
51       * Die zentrale Methode der Snap-Shot-Technik ist
52       * die im Folgenden aufgeführte "create"-Methode.
53       * Sie erzeugt zu einem gegebenen Frame und dessen
54       * inneren Frames rekursiv eine dreispaltige Liste,
55       * wobei in der ersten Spalte eine hierarchische
56       * Frame-ID hinterlegt ist und in der zweiten die URL
57       * des im jeweiligen Frame dargestellten Inhalts. Die
58       * dritte Spalte gibt den Zeitpunkt an, an dem der
59       * Frame zuletzt geändert wurde.
60       */
61      top.SnapShotManager.create = function(framePath,
62                                                  frame){
63          var frameData = framePath + "\t"
64                        + frame.location.href + "\t"
```

```
65                          + frame.LOAD_TIMESTAMP + "\n";
66          for (var k = 0; k < frame.frames.length; k++){
67                  frameData += this.create (
68                  framePath + "/" + k, frame.frames[k]
69              );
70          }
71      };
72
73      /*
74       * Die "transfer"-Methode überträgt schließlich die
75       * angesammelten Daten an das Tracking-System. Dabei
76       * wird quasi ein Snap-Shot ausgehend vom Wurzel-
77       * Fenster erzeugt und zusammen mit weiteren Daten
78       * wie z.B. dem Wert eines Sitzungs-Cookies und dem
79       * Zeitpunkt, an dem die Daten erfasst wurden versen-
80       * det. Der eigentliche Versand ist an dieser Stelle
81       * nur per Kommentar angedeutet und müsste beispiels-
82       * weise unter Verwendung von AJAX oder dynamisch
83       * erzeugten Web-Bugs implementiert werden.
84       */
85      top.SnapShotManager.transfer = function(){
86          var timestamp = (new Date()).getTime();
87          var snapShotData = this.createSnapShot(
88              this.windowId, this.window
89          );
90          /*
91           * übertrage Daten via AJAX, dynamisch erzeugte
92           * Web-Bugs, versteckte IFrames o.a.m.
93            */
94      };
95
96      /*
97       * Die letzte und gleichzeitig zentrale Funktion für
98       * die Verarbeitung der Snap-Shots ist die im Fol-
99       * genden aufgeführte "notify"-Funktion. Sie dient
100      * gleichermaßen als Schnittstelle für die Benach-
101      * richtigung bei Seitenwechseln und gleichzeitig
102      * als periodischer Trigger für den Fall, dass der
103      * Besucher längere Zeit auf einer Seite verharrt.
104      */
105     top.SnapShotManager.notify = function(){
106         if (this.autoNotify){
107             this.autoNotify.cancelEvent();
```

```
108             }
109             this.transfer();
110             this.autoNotify = this.window.setTimeout(
111                 "SnapShotManager.notify()",
112                 AUTO_NOTIFY_INTERVAL
113             );
114         }
115 }
116
117 /*
118  * Nach dem Initialisieren des Snap-Shot-Managers bzw.
119  * nach dem Laden des vorliegenden Scripts in einer be-
120  * liebigen Seite braucht lediglich die "notify"-Methode
121  * auf oberster Fensterebene aufgerufen werden, um das
122  * Tracking im Client zu aktivieren bzw. Änderungen zu
123  * signalisieren.
124  */
125 top.SnapShotManager.notify();
```

Zu diesem browserseitigen Script gehört schließlich ein entsprechendes Gegenstück auf einem Web-Server, welches die gesammelten Daten entgegennimmt und im Mining-System erfasst [40]. Wie auch bei der „redirector"-Technik muss dabei sichergestellt sein, dass das Entgegennehmen der Daten nicht zu einer Überlastung des Systems führt. Dabei muss auch berücksichtigt werden, dass bedingt durch periodische Datenübertragung wesentlich mehr Daten übertragen werden und auch Benutzer, die eigentlich inaktiv sind und die gebotenen Inhalte lediglich im Hintergrund geöffnete haben, Last erzeugen. Außerdem kann davon ausgegangen werden, dass aufeinanderfolgende Snap-Shots nur geringfügige Differenzen aufweisen, sodass es sicherlich sinnvoll ist, Snap-Shots nicht einzeln, sondern in Bündeln zu verarbeiten.

5.5.2 Implikationen für $\mathcal{A} \subseteq \mathbb{A}$

Das Verfahren arbeitet mit client-seitiger Berechnung und setzt somit die Aktivierung einer entsprechenden Script-Sprache (bzw. bei Implementierung in Java die Aktivierung von Java) voraus. Clientseitig wird ein Zustandsbericht erzeugt, in dem alle Informationen des Clients zusammengetragen werden können. Dazu zählt insbesondere die Information, in welchem Frame seit wann welcher Inhalt angezeigt wird. Anhand der Frame-Pfade können zudem detailliert Übergänge in Dokumenten erkannt werden.

Für die Bedeutung von Snap-Shots im Sinne des hier angewendeten Datenmodells möchte ich davon ausgehen, dass in einer Vorbereitungsphase die Snap-Shots zerlegt wurden und in einzelne Zugriffstupel mit zusätzlichen Eigenschaften überführt wurden. Als zusätzliche Eigenschaften kommen dabei die Komponenten "window", "loadTimeStamp" und "frame", die ihrerseits den im obigen Script aufgeführten Variablenwerten entsprechen, hinzu.

Für Einträge, die über Snap-Shots erfasst wurden gilt, dass sie nur dann das gleiche Fenster betreffen können wenn sie der gleichen Sitzung und damit dem gleichen Benutzer zugeordnet werden können. Es gilt daher:

$$\forall a_1, a_2 \in \mathcal{A} : a_1[window] = a_2[window] \Rightarrow a_1[user] = a_2[user] \tag{5.9}$$

Da jedes Fenster für sich einen Navigationsraum bildet, besitzen alle Inhalte, die innerhalb eines Fensters dargestellt wurden eine Beziehung zueinander. Prinzipiell gilt, dass jeder Inhalt eines Fensters, der zeitlich vor einem beliebigen anderen Inhalt im gleichen Fenster angezeigt wurde, ein transitiver Vorgänger im Sinne des vorliegenden Datenmodells ist:

$$\forall a_1, a_2 \in \mathcal{A} : a_1[window] = a_2[window] \wedge a_1[loadTimeStamp] < a_2[loadTimeStamp] - \tau \Rightarrow a_1 \overset{+}{\prec} a \tag{5.10}$$

Innerhalb eines Frames gilt sogar eine direkte Nachfolgerbeziehung:

$$\forall a_1, a_2 \in \mathcal{A} : a_1[window] = a_2[window] \wedge a_1[frame] = a_2[frame] \wedge a_1[loadTimeStamp] < a_2[loadT \tag{5.11}$$

Ist $\triangleright$ eine binäre Relation, die für zwei gegebene Frames angibt, ob der erste den zweiten direkt enthält, so kann man eine Beziehung für die "exit"-Komponente von Zugriffseinträgen angeben. Prinzipiell kann kein innerer Frame ohne einen zugehörigen äußeren Frame angezeigt werden:

$$\forall a_1 \in \mathcal{A} : (\exists a_3 \in \mathcal{A} : a_3[frame] \triangleright a_1[frame]) \Rightarrow (\exists a_2 \in \mathcal{A} : a_2[frame] \triangleright a_1[frame] a_1[access] \leq a_2[e \tag{5.12}$$

Außerdem kann kein Frame zu irgendeiner Zeit mehr als einen Inhalt darstellen. Es gilt:

$$\forall a_1, a_2 \in \mathcal{A} : a_1[frame] = a_2[frame] \wedge a_1[access] < a_2[access] \Rightarrow a_1[exit] < a_2[access] \tag{5.13}$$

5.5.3 Bedeutung

Mit Hilfe der „Snap-Shot“-Technik wird es möglich, Aussagen über die Endzeitpunkte von Aufrufen zu treffen. Da Snap-Shots Zustandsberichte über die dargebotenen Inhalte liefern und auch periodisch Informationen generiert werden, liefert die Technik sogar Verweilzeitinformationen bei Knoten, die am Ende einer Sitzung liegen.

Insbesondere bei Präsenzen, die mit Framesets arbeiten führt diese Technik dazu, dass die logischen Navigationsströme losgelöst von der technischen Navigationsstruktur sichtbar werden. Für den Fall, dass eine Präsenz ein äußeres Umgebungsframeset besitzt, welches nur beim Betreten der Seite geladen wird und von dort an konstant bleibt, wird mit der „Snap-Shot“-Technik sogar ein Sitzungsverlauf erzeugt, der detailliert beschreibt, von wann bis wann der Benutzer welchen Inhalt in welchem Frame betrachtet hat.

Kapitel 6

Datennormalisierung

Das vorangegangene Kapitel hat Möglichkeiten aufgezeigt, wie Daten für eine spätere Analyse erzeugt werden können. Dazu wurden unterschiedliche Strategien aufgeführt, die jeweils verschiedene Informationsarten beinhalten. Damit die Daten im Sinne des konzipierten Systems verarbeitbar sind, müssen sie in eine Form überführt werden, die den Entitäten des in Abbildung 3.1 (Seite 13) gezeigten Basis-Protokolls entsprechen.

Das vorliegende Kapitel zeigt, wie diese Datennormalisierung vonstatten geht, wobei das Ziel die Erzeugung eines chronologischen Datenstroms über Entitäten dieses Basis-Protokolls ist.

6.1 Koppeln der Eingangsströme

Eine erste Phase bei der Datenaufbereitung ist die zeitliche Sortierung der Eingangsströme. Dabei muss sichergestellt werden, dass die eingehenden Daten in der Reihenfolge weitergereicht werden, in der die Informationen bei der Datenerfassung ankamen. Im Falle von Log-Dateien muss beispielsweise sichergestellt werden, dass die Datenzeilen nach den jeweiligen Zugriffszeitpunkten in zeitlicher Abfolge weitergereicht werden. Im Falle von Snap-Shots sollten die Daten in der Reihenfolge weitergerreicht werden, in der sie erzeugt wurden.

An dieser Stelle möchte ich voraussetzen, dass die Datenquellen ihre Daten als sortierten Strom an das aktuelle System reichen.

Die aktuelle Systemkomponente liest dann die Daten aus und erzeugt einen zusammenführenden, sortierten Datenstrom, in dem immer das älteste Element weitergereicht wird.

Die vorliegende Schicht besitzt die Aufgabe, aus den verschiedenen Datenquellen einen serialisierten Datenstrom zu erzeugen. Dazu geht sie im Wesentlichen nach dem Merge-Sort-Prinzip vor und liefert bei jedem Aufruf immer das älteste Element aller Datenquellen zurück.

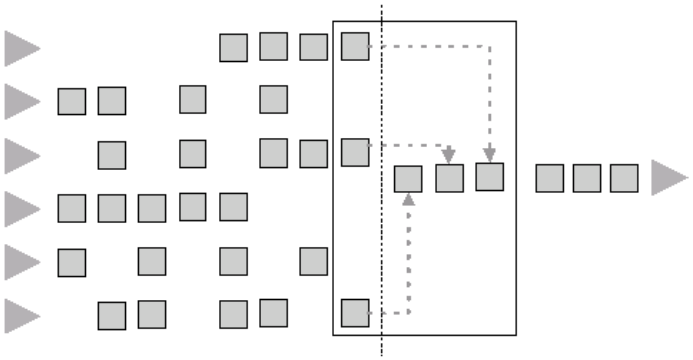

Abbildung 6.1: Merging der Datenquellen

6.2 Zusammenführen der Daten

In den vorherigen Kapiteln wurde bereits beschrieben, welche Daten für das Usage-Mining von zentraler Bedeutung sind. Es wurde dabei insbesondere gezeigt, dass die zentralen Informationen zur Analyse aus der in Abbildung 3.1 (Seite 13) dargestellten 4-Tupel-Relation hergeleitet werden können.

Leider liefert keine der vorgestellten Erfassungstechniken direkt die geforderten Daten. Vielmehr bietet jede Technik Abschätzungen über die tatsächlichen Zugriffsdaten, wobei aus verschiedenen Richtungen unterschiedliche Annäherungen geliefert werden.

Für eine effiziente spätere Aufbereitung ist es notwendig die eingehenden Daten in eine homogene Form zu bringen und als kontinuierlichen Strom weiterzuleiten. Das bedeutet, dass die einmal weitergereichten Daten nicht mehr im Nachhinein angepasst werden dürfen.

Glücklicherweise entstehen die Daten für das Usage-Mining in den unterschiedlichen Systemen zeitnah. So kann davon ausgegangen werden, dass Korrekturmaßnahmen nur innerhalb relativ kurzer Zeitspannen notwendig sind. Es bietet sich daher an, eine Pufferungsstrategie für das Zusammenführen der Daten zu verwenden, sodass beim Auswerten der Quelldaten Zugriffstupel in einem Korrekturpuffer abgelegt werden, die dort für eine bestimmte Zeit lang korrigiert und angepasst werden können und nach einer gewissen Reifephase in einem Datenstrom an die nächste Schicht geleitet werden können.

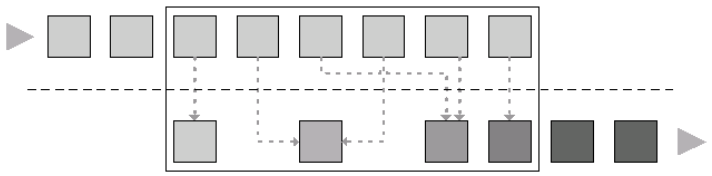

Abbildung 6.2: Reifungsprozess im Korrekturpuffer

Vorausgesetzt, dass jede Datenquelle ihre Daten direkt an den Korrekturpuffer weitergibt, kann davon ausgegangen werden, dass die Korrekturphasen im Wesentlichen dem Zeitraum entsprechen, den Benutzer maximal auf einer Seite verbringen.

Was bedeutet es aber aus Usage-Mining-Sicht, wenn Benutzer sehr lange auf einem Inhalt verweilen? Kann man sich wirklich Personen vorstellen, die stundenlang denselben Inhalt betrachten? Im Internet ist in solchen Situatio-

nen sicherlich eher davon auszugehen, dass das System kontinuierlich weiter misst, der Benutzer aber zwischenzeitig zu anderen Tätigkeiten übergegangen ist. Kommt der Benutzer also nach einem längeren Zeitraum zurück, so kann sicherlich davon ausgegangen werden, dass er sich nicht mehr im selben Kontext befindet, in dem er ursprünglich auf den Inhalt gelangt ist. Entsprechend ist also davon auszugehen, dass es sich um einen neuen Besuch handelt. Daher ist es sogar ein gewünschtes Verhalten, wenn der Korrekturpuffer Datensätze aufteilt, die zu lange Verweilzeiten erfassen. Die maximale Zeit, die Einträge manipulierbar im Korrekturpuffer verbringen, entspricht also genau der maximal angenommen Besuchszeit einer Seite. Für eine typische HTML-Seite kann man sicherlich mit gutem Gewissen sagen, dass diese maximal sinnvolle Verweilzeit kürzer als eine Stunde (der Wert liegt in der Realität sicherlich im Minutenbereich) ist, sodass die Daten also mit einer Verzögerung von ca. einer Stunde zur Analyse bereitstehen.

Den Korrekturpuffer können wir uns als eine Relation vorstellen, in der jedes Tupel alle Attribute der eingehenden Datenquellen besitzt. Dabei sind im Korrekturpuffer Mehrfachvorkommen von Einträgen zulässig. Bei der Weitergabe werden die Daten auf die in Abblidung 3.1 dargestellten Attribute projeziert, wobei bei der Projektion Mehrfachvorkommen eliminiert werden.

Da die Daten sehr heterogenen Quellen entstammen können, ist im Allgemeinen davon auszugehen, dass es kein triviales „Join“-Kriterium gibt, mit dem die Daten zusammengeführt und im Idealfall durch wenige Aggregatfunktionen in die aufgeführte Zielrelation überführt werden können. Da zudem im Usage-Mining kontinuierlich neue Daten anfallen, gestaltet sich der Datenerfassungsprozess als ebenfalls kontinuierlicher Korrekturprozess, bei dem stetig Einträge für die Relation hinzukommen, durch andere ersetzt werden oder sogar gänzlich wegfallen.

Bevor die Daten jedoch in die nächste Ebene als *AccessEntry*-Objekte gelangen, müssen sie erst mal im Korrekturpuffer ankommen und hier entsprechend mit anderen Daten verschmolzen werden.

Die folgenden Abschnitte zeigen daher, wie die einzelnen Datenquellen im Korrekturpuffer aufbereitet werden müssen.

6.2.1 Sammeln von CLF-Daten

Das erste vorgestellte Verfahren zur Erfassung von Daten besteht in der Verarbeitung von CLF-Protokoll-Informationen. Bei der Überführung von Ein-

trägen aus dem Common Log Format müssen die folgenden Eigenschaften berücksichtigt werden:

1. Jeder Eintrag aus dem CLF-Protokoll muss einen korrespondierenden Eintrag im Zielprotokoll besitzen.
2. Kein Eintrag aus dem Zielprotokoll vertritt mehr als einen Eintrag aus dem CLF-Protokoll.

Pendant-Existenz

Die erstgenannte Eigenschaft kann beim Verarbeiten der Einträge relativ leicht erreicht werden. Da der Korrekturpuffer immer alle Elemente vorhält, die innerhalb einer maximal sinnvollen Verweilzeit bis zum aktuellen Zeitpunkt erfasst wurden und die Messverzerrung von Zeitwerten serverseitiger Zugriffsmessung im Allgemeinen vergleichsweise klein sind, braucht lediglich für jeden Eintrag aus dem CLF-Protokoll geprüft werden, ob bereits ein korrespondierender Eintrag im Korrekturpuffer liegt. Bei dieser Korrespondenzprüfung werden Attributvergleiche, in denen ϵ-Werte auftauchen ignoriert, sodass auch Einträge gefunden werden, die von anderen Messtechniken partiell erfasst wurden. Wird bei der Suche ein passendes Pufferelement gefunden, so können gegebenenfalls fehlende Attributbelegungen ergänzt werden. Dabei können vor allem die Felder *user*, *object* und *address*, wie unter Pendant-Erzeugung beschrieben (s.u.), aufgefüllt werden.

Liegt kein korrespondierender Eintrag im Korrekturpuffer vor, so muss prinzipiell ein neuer Eintrag erzeugt werden (s.u.).

Injektivität der Pendant-Abbildung

Um sicherzustellen dass jedem Eintrag aus dem Zielprotokoll höchstens ein Eintrag aus dem CLF-Protokoll entspricht, möchte ich die Verwendung einer Markierungstechnik empfehlen. Im Korrekturpuffer wird dazu für jedes Element ein binäres Attribut hinterlegt, welches angibt, ob das jeweilige Element bereits einem anderen Element zugeordnet ist. Beim Prüfen der Pendant-Existenz für einen neuen Eintrag müssen dann alle Pufferelemente ignoriert werden, bei denen diese binäre Markierung bereits gesetzt ist. Sofern bei diesem erweiterten Pendant-Existenz-Test kein Element gefunden wird, bedeutet dies, dass entweder kein Pendant existiert oder alle korrespondierenden Objekte bereits von anderen CLF-Protokolleinträgen zugeordnet wurden.

Wird bei dem erweiterten Pendant-Existenz-Test genau ein Element gefunden, so muss dieses Element entsprechend markiert werden und der nächste Eintrag kann verarbeitet werden.

Liefert der erweiterte Pendant-Existenz-Test mehr als ein Element zurück, so muss das älteste Element als genutzt markiert werden. Da die einkommenden CLF-Einträge gemäß 6.1 zeitlich sortiert sind, wird es durch Markieren des ältesten Elements möglich, Folgeeinträge lückenfrei anderen Elementen zuzuordnen

Pendant-Erzeugung

Wird für einen CLF-Eintrag erkannt, dass kein freies Pendant-Element im Korrekturpuffer existiert, muss dem Korrekturpuffer ein neues Pendant-Element hinzugefügt werden. Dabei werden die Komponenten dieses neuen Elements a wie folgt belegt:

$a[user]$ Der tatsächliche Benutzer des Zugriffs kann, sofern verfügbar, dem CLF-Protokollwert $AUTHUSER$ entnommen werden. Falls der Wert nicht verfügbar ist, wird hier der ϵ-Wert verwendet.

$a[object]$ Der Wert des aufgerufenen Objekts muss anhand des CLF-Protokollwerts $REQUEST - URL$ hergeleitet werden, wobei im Wesentlichen der entsprechende Rückgabewert der Hilfsfunktion $objectOf$ genutzt werden muss.

$a[access]$ Der Zeitpunkt, an dem der Benutzer den Aufruf des Objekts eingeleitet hat, steht bei serverseitiger Erfassung nicht korrekt zur Verfügung. Vielmehr ist der tatsächliche Wert prinzipiell kleiner, als der erfasste Wert. Da jedoch keine anderen Daten zur Verfügung stehen, muss der verfügbare $DATE$-Wert aus dem CLF-Protokoll verwendet werden.

$a[exit]$ Das CLF-Protokoll erfasst keine Endzeitpunkte für die Betrachtung von Inhalten. Dem *exit*-Wert des Zielprotokoll-Eintrags muss daher ein ϵ zugewiesen werden.

$a[address]$ Damit andere Datenquellen mit dem neu erzeugten Pufferelement assoziierbar sind, wird im Element die IP-Adresse des TCP-Verbindungspartners als Assoziationsinformation hinterlegt. Der Wert entspricht dem
REMOTEHOST-Wert aus dem CLF-Protokoll-Eintrag.

Alle weiteren Attribute, die in Verbindung mit anderen Log-Einträgen notwendig sind, erhalten an dieser Stelle ϵ zugewiesen, da der CLF-Protokolleintrag keine zusätzlichen Informationen birgt. Beim Einfügen in den Korrekturpuffer, wird der Eintrag zudem markiert, sodass er für weitere CLF-Einträge nicht erneut genutzt wird.

6.2.2 Sammeln von ECLF-, Redirector und Ping-Daten

Die erste Erweiterung der CLF-Protokolldaten bildeten die ECLF-Protokoll-Informationen für die, neben der Erfassung durch Server-Protokolle, auch die Möglichkeiten der Erfassung über „Redirector"- oder „Ping"-Mechanismen erläutert wurden.

Der Aufbereitung von ECLF-Protokoll-Informationen liegt prinzipiell die Verarbeitung der CLF-Informationen zu Grunde. Im Falle, dass keine $referer$-Information zur Verfügung steht, sind die Verfahren sogar identisch. Es wird ebenfalls eine zusätzliche Markierung eingeführt, die für bereits zugeordnete Daten genutzt wird. Beim Erzeugen bzw. Ergänzen von Datensätzen wird zusätzlich die *agent*-Komponente mit dem entsprechenden $USERAGENT$-Wert aus dem ECLF-Protokolleintrag belegt.

Einträge mit $referer$-Informationen beeinträchtigen die Elemente im Korrekturpuffer zusätzlich. Sofern ein Eintrag einen $referer$-Wert besitzt, muss im Korrekturpuffer ein entsprechender $referer$-Eintrag existieren dessen *exit*-Wert mit dem *access*-Wert des aktuellen Eintrags korrespondiert. Sofern kein solcher $referer$-Eintrag existiert, kann dem Korrekturpuffer entweder ein neues Element hinzugefügt werden oder der *access*-Wert des aktuellen Eintrags leicht verändert werden, sodass die Bedingung erfüllt ist. Falls ein solches Objekt existiert, aber einen ϵ-Wert als *exit*-Wert verwendet, kann für dieses Objekt einfach der aktuelle *access*-Wert als *exit*-Wert eingesetzt werden.

Anpassung der Komponente *access*

Sofern bei der Verarbeitung eines ECLF-Protokolleintrages für den $referer$ ein geeignetes Objekt π im Korrekturpuffer existiert, dessen *exit*-Wert ($\pi[exit]$) nur minimal vom aktuellen *access*-Wert ($e[access]$) abweicht, so kann der aktuelle *access*-Wert entsprechend auf diesen *exit*-Wert gesetzt werden. Dabei bedeutet minimal: $e[access] - \tau \leq \pi[exit] < e[access]$ mit τ als maximalen Verzögerungswert, mit dem Zugriffe serverseitig erfasst werden.

Erzeugung eines Referer-Pendant

Für den Fall, dass zu einem Zugriffsprotokolleintrag e kein passender $referer$ im Korrekturpuffer liegt, gibt es drei Gründe:

1. Es gab im Vorfeld keinen zum $referer$ passenden ECLF-Eintrag, da beispielsweise ein Web-Cache Inhalte frühzeitig zurückgegeben hat oder der $referer$ ein extern liegendes Objekt betrifft.

2. Der $referer$ wurde zwar im Vorfeld erfasst, ist mittlerweile aber bereits aus dem Korrekturpuffer weitergereicht worden, da er älter als die maximal zulässige Verweilzeit war.

3. Der Referer selbst ist ein spezielles Navigationselement, welches dem Benutzer beispielsweise in einem separaten Frame dargeboten wird und damit Ausgangspunkt einer Vielzahl von Übergängen ist.

In den ersten beiden Fällen existiert im Korrekturpuffer überhaupt kein Eintrag, der einen Zugriff auf das besagte $referer$-Objekt durch den aktuellen Benutzer darstellt. Es ist daher in diesen Fällen nur natürlich, ein neues Pendant-Objekt im Korrekturpuffer abzulegen, das allerdings als *exit*-Wert den aktuellen *access*-Wert nutzt und als *object*-Wert das entsprechend referenzierende Objekt (ermittelbar als $objectOf(e[REFERER])$ verwendet. Als *access*-Wert bietet sich der um die maximal vom Korrekturpuffer verwendete Seitenverweilzeit verringerte *exit*-Wert des neuen Eintrages an.

Im dritten Fall hat der Benutzer einen Wechsel von einem dritten Inhaltsobjekt auf dieses referenzierende Objekt durchgeführt, was daher einem Neuaufruf dieses referenzierenden Objekts gleichkommt. Im Korrekturpuffer kann die Situation dadurch erkannt werden, dass für den gleichen Benutzer bereits

Zugriffe auf das $referer$-Objekt mit länger zurückliegender *exit*-Zeit existieren. In diesem Falle muss im Korrekturpuffer Π ein neues Element wie folgt erzeugt werden:

1 $\pi_{ref}[exit] = e[access]$
2 $\pi_{ref}[object] = objectOf(e[referer])$
3 $\pi_{ref}[user] = e[user]$
4 $\pi_{ref}[access] = max(\{t \in \mathbb{T} | \exists \pi \in \Pi : \pi[user] = e[user] \wedge \pi[exit] = t\}$

6.2.3 Berücksichtigung der Cookie-Erweiterung

Cookies können beim Aufbereiten trivial berücksichtigt werden. Letztendlich handelt es sich bei diesen Werten lediglich um genauere Sitzungsidentifikationsmerkmale, die beim Anpassen und Aktualisieren der Daten als zusätzlicher *cookie*-Wert gesetzt werden. In den obigen Korrespondenzprüfungen muss dann lediglich zusätzlich geprüft werden, ob auch dieser Wert in Pendants passt. Ist dies nicht der Fall, so bedeutet dies, dass kein entsprechendes Pendant existiert.

6.2.4 Sammeln von Snap-Shots

Für die Verarbeitung von Snap-Shot-Daten gilt ebenfalls ein ähnliches Vorgehen wie beim Verarbeiten von Protokolldaten. Damit die Daten verbunden werden können, möchte ich an dieser Stelle voraussetzen, dass beim Erfassen der Snap-Shots die IP-Adresse des Kommunikationspartners, ggfs. verfügbare Cookie-Werte und der User-Agent ebenfalls gespeichert werden.

Bei der Verarbeitung von Snap-Shot-Einträgen müssen 4 Fälle unterschieden werden:

1. Im Korrekturpuffer existieren bereits Einträge, die den gleichen Frame betreffen wie der zu verarbeitende Snap-Shot. Falls der letzte dieser Einträge das gleiche Objekt wie der aktuelle Eintrag betrifft, muss lediglich die *exit*-Komponente des jeweiligen Eintrags angepasst werden, da der neue Snap-Shot möglicherweise eine Verlängerung der Verweilzeit darstellt. Falls ein anderes Objekt aufgeführt wird, muss der *exit*-Wert dieses alten Eintrags auf den *access*-Wert des neuen Snap-Shots gesetzt werden und der Snap-Shot als neues Pufferelement aufgenommen werden, da zum *access*-Zeitpunkt des Snap-Shots der Inhalt des Frames ausgetauscht wurde.

2. Es gibt zwar keinen den Frame betreffenden Inhalt, aber für das zugehörige Fenster wurden bereits Daten erfasst. Da ein Frame nie ohne ein zugehöriges Frameset existieren kann, handelt es sich entweder um eine framesetfreie Seite oder das zugehörige Frameset bzw. ein in diesem Frameset enthaltener Frame wurde geladen, wobei dieser durch die gleiche Benutzeraktion aufgerufen wurde, die auch das Laden des vom Snap-Shot umfassten Eintrags bewirkt hat. Falls keine Framesets zum Einsatz kommen, ist dies zu werten als ob das Fenster aus genau einem Frame besteht. Dabei ist wie in Fall 1 vorzugehen. Andernfalls liegt für den aktuellen Snap-Shot ein verzerrter *access*-Wert vor, sodass der *access*-Wert des zuletzt im Fenster geladenen Elements ebenfalls für den gegenwärtigen Snap-Shot gilt.

3. Es gibt zwar noch keine Snap-Shot-Daten für den betreffenden Inhalt, wohl aber Protokolldaten, die aus anderen Messverfahren herrühren. In diesem Fall müssen lediglich die Snap-Shot spezifischen Informationen übernommen werden.

4. Zum Snap-Shot existieren keine verwandten Einträge im Korrekturpuffer. In diesem Falle muss der Snap-Shot unverändert aufgenommen werden.

Der folgende Algorithmus stellt die prinzipielle Vorgehensweise zum Aufnehmen eines Snap-Shot-Eintrages s in einen Korrekturpuffer Π dar.

```
1  Π' ← {π ∈ Π|π[frame] = s[frame] ≠ ε}
2  Π'' ← {π ∈ Π|π[window] = s[window] ≠ ε}
                          eclf
3  Π''' ← {π ∈ Π|π ⊨ s}
4  if Π' ≠ ∅ then
5       π ← last(Π')
6       if π[object] = s[object] then
7            π[exit] ← max(π[exit], s[date])
8       else
9            π[exit] ← s[access]
10           Π ← Π ∪ {s}
11      end if
12 else if Π'' ≠ ∅ then
13      π ← last(Π'')
14      if s[frame] = ε then
15           if π[object] = s[object] then
16                π[exit] ← max(π[exit], s[date])
17           else
```

```
18                  π[exit] ← s[access]
19                  Π ← Π ∪ {s}
20              end if
21          else
22              s[access] ← π[access]
23              Π ← Π ∪ {s}
24          end if
25  else if Π''' ≠ ∅ then
26          π ← last(Π'')
27          π[access] ← min(π[access], s[access])
28          π[exit] ← max(π[exit], s[date])
29          π[window] ← s[window])
30          π[frame] ← s[frame])
31  else
32          Π ← Π ∪ {s}
33  end if
```

Kapitel 7

Vorberechnung und Abfrage

In den vorangegangenen Kapiteln wurde beschrieben, wie über unterschiedliche Datenquellen ein kontinuierlicher Datenstrom über elementare Zugriffsprotokolleinträge (vgl. Abbildung 3.1 auf Seite 13) erzeugt wird.

In Kapitel 3 wurden zudem Messwerte vorgestellt, die allein anhand dieser Relation berechenbar sind. Die Berechnung der Messwerte ausschließlich anhand dieser Basisrelation liegt mindestens in einer linearen Komplexitätsklasse, wobei sich der lineare Koeffizient auf die Anzahl der Einträge der Basisrelation bezieht. So muss beispielsweise für die Berechnung des Messwerts „Hits“ (s. Seite 15) gezählt werden, wie viele Elemente in dieser Relation enthalten sind.

Um vor Augen zu führen, was dieser lineare Aufwand für eine gewöhnliche Web-Site mit hohem Besuchsaufkommen bedeutet, hier ein kurzes Rechenexempel: Gegeben sei eine zu analysierende Web-Site mit einem durchschnittlichen Besuchsaufkommen von täglich 200000 Seitenzugriffen, von denen etwa jede Seite 5 Dateien lädt. In einem Jahr entstehen dabei über 182 Millionen Datensätze. Nehmen wir weiter an, jeder der obigen Werte nimmt einen 128-Bit Wert (pro Komponente aus Abbildung 3.1 ein 32-Bit Integer-Wert) in der Datenbank ein, so würden die reinen Protokolldaten nach einem Jahr bereits 5 Gigabyte umfassen.

Natürlich möchte ein Analyst diese Daten aus verschiedenen Blickwinkeln betrachten. Insbesondere möchten Analysten verschiedene Zeiträume oder Inhaltsbereiche miteinander vergleichen. Dabei stellen sie eine Vielzahl von Anfragen wobei man sich ausmalen kann, dass selbst die Best-Case-Situation, in der lediglich genau diese 5 Gigabyte pro Abfrage einmal ausgelesen werden,

kein akzeptierbares Systemverhalten bieten.

In dieser Arbeit möchte ich daher eine alternative Vorgehensweise vorschlagen. Dazu werden Statistiken über das Basisprotokoll vorberechnet, die eine effiziente Berechnung benötigter Messwerte ermöglichen.

Aufbereitungsdimensionen

Die Standardvorgehensweise beim Web-Usage-Mining ist es, auf Clustering-Strategien aus dem Data-Mining [6] zurückzugreifen. Dabei werden im Vorfeld die Daten nach vorgegebenen Kriterien analysiert und ein Bericht zurückgegeben. Es wird in Kauf genommen, dass diese Analyse längere Zeit in Anspruch nimmt, da man im Vorfeld versucht, die Analysen in der Form durchzuführen, dass alle für den Analysten zentralen Informationen in diesem Bericht enthalten sind. Diese Vorgehensweise ist insbesondere dann sinnvoll, wenn bestehende Theorien oder Erwartungswerte geprüft werden sollen. Bei der Analyse von unbekannten Domänen ist diese Vorgehensweise jedoch, wie einleitend bereits erwähnt, nur bedingt nützlich und erfordert vom Analysten eine hohes Maß an Erfahrung.

Meine berufliche Erfahrung hat mir gezeigt, dass diese Analysten selten langwierige Erfahrung im Auswerten von Web-Präsenzen besitzen. Das ist auch wenig verwunderlich, wenn man berücksichtigt, seit wann der Nutzen derartiger Analysen erkannt wurde. Die Analysten versuchen daher mit Erfahrungen aus anderen Bereichen (z.B. Marketing-Aktionen in Druckwerken oder Fernsehen) an die Materie heranzugehen. Das zentrale Vorgehensmodell dabei ist, einen **konkreten Zielbereich** zu fokussieren und für diesen die **zeitliche Entwicklung** zu beobachten.

Übertragen auf den hier vorliegenden Anwendungsfall existieren also die folgenden zwei zentralen Kriterien zur Beschreibung einer Analyse-Perspektive:

Betrachteter Zielbereich

Wie bereits am Anfang dieser Arbeit erwähnt, ist ein inhaltlicher Bereich einer Web-Site oder allgemeiner einer Web-Anwendung zu verstehen als eine Menge von Inhaltsobjekten. Theoretisch ist denkbar, dass Analysten beliebige Bereiche definieren und diese später auswerten. Für eine Vorberechnung wäre es daher prinzipiell notwendig, für alle denkbaren Bereiche angemessene Aggregatinformationen bereitzustellen. Glücklicherweise sind Inhalte in

der Regel nicht völlig unorganisiert, sondern über Kategorien, Kapitel oder über ein Dateisystem strukturiert. Diese Organisationseinheiten stellen dabei auch eine inhaltliche Hierarchie dar. In einer Hierarchie kann dann jeder Teilbaum mit einem Bereich des Analysesystems gleichgesetzt werden, sodass bereits initial eine komplexe Bereichemenge gebildet werden kann, die dem einleitend vorgestellten „Zooming"-Prinzip gerecht wird.

Im Folgenden möchte ich davon ausgehen, dass es eine Menge $\Omega \subseteq P(\mathbb{O})$ von Bereichen (Mengen von Objekten) gibt, die für Analysten zumindest initial hinreichend sind. Sofern Anaylsten im Späteren erkennen, dass feinere Bereiche notwendig sind, wird in Kauf genommen, dass sämtliche Statistiken neu berechnet werden müssen. Aufgrund der Tatsache, dass bereits im Initialzustand eine feingranulare Bereichsaufgliederung existiert, kann jedoch davon ausgegangen werden, dass das Definieren neuer Bereiche hinreichend selten vorkommt, sodass in diesen Situationen längere Laufzeiten für eine vollständige Neuberechnung akzeptierbar sind.

Betrachtungszeitraum

Die zeitliche Dimension besitzt eine besondere Bedeutung bei Analysen. Sie gibt die Möglichkeit, eine Zuordnung zu internen und externen Ereignissen vorzunehmen.

Ein Betrachtungszeitraum in einer Analyse ist als ein Intervall über Zeitpunkte zu verstehen, für die der Analyst die Auswirkungen von Ereignissen betrachten möchte. Abgesehen von punktuellen Intervallen besitzen solche Intervalle im Allgmeinen unendlich viele Zeitpunkte. Es kann davon ausgegangen werden, dass es Analysten kaum interessiert, ob ein Benutzer eine Millisekunde früher oder später auf ein Objekt zugegriffen hat. Da natürliche Personen die Erzeugung von Zugriffsdaten bewirken, ist sicherlich auch Sekunden- oder Minutenpräzision nicht unbedingt notwendig. Bei der Betrachtung großer Zeiträume (wie z.B. Jahre) werden zunehmend geringere Anforderungen an die Präzision der Zeitzuordnung gestellt.

In diesem Kapitel wurde zudem einleitend erwähnt, dass leicht ein großes Datenaufkommen entstehen kann. Dabei ist ein wesentlicher Aspekt, dass die Größe dieses Datenaufkommens von der Größe des Betrachtungszeitraums abhängt. In Verbindung mit einer geeigneten Datenbankindizierung ist es daher durchaus vertretbar, Analysen für kleine Betrachtungszeiträume direkt auf der Basisrelation $\mathcal{A}$ auszuführen.

Effiziente Verarbeitung spielt daher im Wesentlichen eine Rolle, wenn es darum geht, größere Zeiträume zu analysieren. Für große Zeiträume ist es aufgrund der obigen Aussagen möglich, Betrachtungszeiträume über diskrete Abtastpunkte festzulegen. Als geeignete Abtastperiode möchte ich im Folgenden von 24 Stunden ausgehen, da Personen bei der Betrachtung großer Zeiträume für gewöhnlich datumsorientiert arbeiten und von konkreten Uhrzeiten abstrahieren.

Vorgehen

Ziel der im folgenden aufgeführten Berechnungen ist es, zusätzliche Statistiken über die Protokolleinträge aus $\mathcal{A}$ zu hinterlegen, die eine effiziente Abfrage des jeweiligen Messwerts für größere Zeiträume ermöglichen. Grundlegende Idee hierbei ist es, den jeweiligen Messwert für jeden Bereich und jeden Tag vorzuberechnen, sodass Tageswerte durch einfaches Nachschlagen abfragbar werden. Für die Abfrage größerer Zeiträume müssen dann Korrekturrelationen erzeugt werden, die es ermöglichen durch einfache Aggregation der im Betrachtungszeitraum liegenden Tageswerte zum Gesamtwert für den jeweiligen Betrachtungszeitraum zu gelangen. Auf diese Weise hängt der Berechnungsaufwand für Analyseanfragen im Wesentlichen von der Anzahl der Tage im Betrachtungszeitraum ab. Für eine Jahresstatistik ist somit im besten Falle die Aggregation von 365 Werten notwendig.

7.1 Hits

Für die Vorberechnung des Messwerts „Hits“ ist lediglich eine Relation erforderlich, die zu jedem Tag und jedem Bereich die Anzahl der Zugriffe auf diesen Bereich zählt. Es handelt sich demnach um eine einfache 3-Tupel-Relation:

	HitStatistic	$s \in \mathcal{S}$
$\mathbb{T}:$	DATE	$s[date]$
$\Omega:$	AREA	$s[area]$
$\mathbb{N}:$	HitCount	$s[hits]$

Abbildung 7.1: Tagesstatistik für Messwert Hits

7.1.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$

Beim Verarbeiten eines neuen Protokolleintrages a wird einfach der Zähler für die entsprechenden Tagesbereichseinträge inkrementiert, bzw. ein neuer Eintrag erzeugt, wenn noch kein entsprechendes Tupel existiert:

```
1  for  O ∈ Ω|a[object] ∈ O
2       S' ← {s ∈ S|O = s[area] ∧ s[date] = date(a[access])}
3       if  S' = ∅  then
4            S ← S ∪ {(date(a[access]), O, 1)}
5       else
6            for  s ∈ S'
7                 s[hits] ← s[hits] + 1
8            end  for
9       end  if
10 end  for
```

7.1.2 Abfrage des Werts der Tage $d_1 \ldots d_n$ für Bereich $\mathcal{O} \in \Omega$

Um die Anzahl der Zugriffe für einen gegebenen Zeitraum zu bestimmen, muss die Summe der entsprechenden Tageswerte gebildet werden:

```
1  S' ← {s ∈ S|s[area] = O ∧ s[date] ≥ d_1 ∧ s[date] ≤ d_n}
2  return  Σ_{s∈S'} s[hits]
```

7.2 Verweilzeit

Wie auch bei den „Hits“-Werten wird lediglich eine Relation benötigt, die zu jedem Tag und jedem Bereich die entsprechende Gesamtverweilzeit zählt:

	DwellStatistic	$s \in \mathcal{S}$
$\mathbb{T}$:	DATE	$s[date]$
Ω :	AREA	$s[area]$
$\mathbb{N}$:	DwellSum	$s[dwell]$

Abbildung 7.2: Tagesstatistik für Messwert „Verweilzeit“

7.2.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$

Analog zur Berechnung der Statistikrelation für den Messwert „Hits“ muss beim Hinzukommen eines neuen Zugriffsprotokolleintrags a einfach die Verweilzeit dieses Eintrags addiert werden:

```
1  for  O ∈ Ω|a[object] ∈ O
2       S' ← {s ∈ S|s[area] = O ∧ s[date] = date(a[access])}
3       if  S' = ∅  then
4            S ← S ∪ {(date(a[access]), O, a[exit] − a[access])}
5       else
6            for  s ∈ S'
7                 s[dwell] ← s[dwell] + a[exit] − a[access]
8            end  for
9       end  if
10 end  for
```

7.2.2 Abfrage des Werts der Tage $d_1 \ldots d_n$ für Bereich $\mathcal{O} \in \Omega$

Um die Gesamtverweilzeit für einen gegebenen Zeitraum zu bestimmen, muss die Summe der entsprechenden Tageswerte gebildet werden:

```
1  S' ← {s ∈ S|s[area] = O ∧ s[date] ≥ d_1 ∧ s[date] ≤ d_n}
2  return  Σ_{s∈S'} s[dwell]
```

7.3 Besucher

Bei der Vorberechnung der Besucherzahl muss berücksichtigt werden, dass Besucher in größeren Zeiträumen zurückkehren können und somit die Anzahl der Besucher, die innerhalb zweier Tage einen Bereich besucht haben, in der Regel kleiner ist als die Summe der jeweiligen Tageswerte. Um dennoch die Anzahl der Besucher für ein konkretes Zeitintervall zu bestimmen, wird eine Korrekturrelation benötigt, über die sich effizient der in der Summenbildung enthaltene Fehler abfragen lässt.

Die Basisstatistik für die Bestimmung der Besucherzahl besitzt den typischen Aufbau der vorangegangenen Verfahren:

	VisitorStatistic	$s \in \mathcal{S}$
$\mathbb{T}$:	DATE	$s[date]$
Ω :	AREA	$s[area]$
$\mathbb{N}$:	VisitorCount	$s[visitors]$

Abbildung 7.3: Tagesstatistik für Messwert „Verweilzeit“

Dazu kommt eine Korrekturrelation, die für jeden Bereich und je zwei Tage festhält, wie viele unterschiedliche Besucher am ersten dieser beiden Tage den Bereich betraten und am zweiten dieser beiden Tage zurückkehrten. Das Schema dieser Relation sieht wie folgt aus:

	ReturnerStatistic	$k \in \mathcal{K}$
$\mathbb{T}$:	PREVIOUSDATE	$k[date_1]$
$\mathbb{T}$:	NEXTDATE	$k[date_2]$
Ω :	AREA	$k[area]$
$\mathbb{N}$:	ReturnerCount	$k[returners]$

Abbildung 7.4: Korrekturtabelle für Messwert „Besucher“

7.3.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$

Der *visitors*-Zähler in der Relation $\mathcal{S}$ muss immer genau dann inkrementiert werden, wenn ein neuer Zugriffseintrag einen Benutzer referenziert, der am Tag des Zugriffseintrags noch nicht auf diesen Bereich zugegriffen hat. Zusätzlich muss der *returners*-Zähler für zwei Tage genau dann inkrementiert werden, wenn ein Benutzer erstmalig an einem Tag einen konkreten Bereich

betritt und den selben Bereich zuletzt an einem früheren zweiten Tag betreten hat.

Bei der Verarbeitung eines Zugriffseintrags muss demnach in Erfahrung gebracht werden, wann ein Benutzer zuletzt die diesen Zugriffseintrag betreffenden Bereiche betreten hat. Nun kann diese Anfrage zwar prinzipiell direkt anhand der Basisrelation $\mathcal{A}$ ausgeführt werden, jedoch muss dazu ein natürlicher Verbund der Basisrelation mit einer Relation aufgebaut werden, die zu jedem Objekt alle dieses enthaltende Bereiche aufführt. Aus Performanzsicht wäre es an dieser Stelle am besten, wenn dieser Verbund bereits in einer zusätzlichen Relation existieren würde oder zumindest eine Relation, die zu jedem Benutzer und zu jedem Bereich den Zeitpunkt des letzten Zugriffs angibt. Da im System sicherlich zahlreiche Bereiche definiert sind, wächst diese Verbundrelation leicht auf das zehn oder hundertfache der Basisrelation an.

In einem eigenen Projekt konnte ich feststellen, dass über 30% der Besucher, die zu einem Bereich überhaupt zurückkehrten, dies direkt am nächsten Tag taten. In über 95% der Fälle kamen sie, wenn überhaupt, innerhalb von 14 Tagen wieder. Natürlich sind diese Daten exemplarisch und erheben auch keinen Anspruch auf allgemeine Gültigkeit. Trotzdem zeigt sich hierdurch eine allgemeingültige Optimierungsoption:

Prinzipiell ist es durchaus plausibel, das Benutzer die eine Site häufig besuchen, dies auch in relativ kurzen Zeitabständen (beispielsweise innerhalb von 14 Tagen) machen. Außerdem ist es Plausibel, dass diese Besucher bestimmte Seiten, ihre Ausgangspunkte, sehr häufig besuchen und den Großteil einer Site eher einmalig oder überhaupt nicht. Auf der anderen Seite gibt es für Besucher, die eine Site nur selten aufsuchen, auch nur wenige Zugriffsprotokolleinträge für die getestet werden müsste, welche zugehörigen Bereiche der Besucher bereits besucht hat.

Es bietet sich daher an, für die Ermittlung des letzten Zugriffs auf einen Bereich ein Drei-Wege-Verfahren zu verwenden. Dazu gibt es einen Cache Γ_δ, der für einen vorgegebenen Zeitraum δ (beispielsweise die letzten 14 Tage) festhält, welcher Benutzer zuletzt auf welchen Bereich zugegriffen hat und damit einen Ausschnitt der letzten 14 Tage aus der zuvor erwähnten Verbundrelation vorhält. Für den Fall, dass ein Besucher innerhalb von 14 Tagen erneut auf einen Inhalt zugreift, kann somit durch einfaches Nachschlagen in der folgenden Cache-Relation der Zeitpunkt des Zugriffs ermittelt werden:

Die Frage, ob ein Benutzer erstmalig die gesamte Site besucht, kann durch einfaches Nachschlagen in der Basisrelation beantwortet werden, wobei ein Index für den Gleichheitsvergleich auf der Komponente *user* ausreicht.

	RecentVisitCache	$\gamma \in \Gamma_\delta$
Ω :	AREA	$\gamma[area]$
$\mathbb{U}$:	USER	$\gamma[user]$
$\mathbb{T}$:	RecentDate	$\gamma[date]$

Abbildung 7.5: Cache für Besucher-Berechnung

Für den Fall, dass ein Benutzer selten auf der Seite aktiv ist, ist das Erzeugen des gesamten Verbundausschnitts für diesen Benutzer ebenfalls relativ unkritisch, da es für den Benutzer nur wenige Einträge in der Basisrelation geben kann.

Der einzige Fall, der nun noch ineffizient verarbeitet wird, liegt vor, wenn ein Benutzer häufig auf einer Site agiert, aber dann einen Bereich betritt, den er schon lange nicht mehr oder noch überhaupt nicht besucht hat. Da aber davon ausgegangen werden kann, dass Besucher sich in bestimmten Bereichen häufig aufhalten und vergleichsweise selten in andere Bereiche wechseln, ist dieser Fall entsprechend unkritisch. Es ist daher aktzeptabel, dass die Verarbeitung dieses Falls ein wenig länger dauert. (An dieser Stelle sei angemerkt, dass noch die im Folgenden nicht weiter berücksichtigte Option besteht, für jeden Bereich benutzerunabhängig zu hinterlegen, wann dieser Bereich zuerst und wann zuletzt betreten wurde. Für den Spezialfall, dass eine Site sehr viele aktualitätsbezogene Inhalte bietet, kann dadurch die Anzahl der zu verarbeitenden Einträge aus dem Zugriffsprotokoll möglicherweise stark reduziert werden. Da diese Strategie allerdings nur in Spezialfällen nützlich ist, möchte ich sie hier nicht weiter vertiefen.)

Das Hinzufügen eines neuen Protokolleintrags a ist im folgenden Pseudo-Code nochmals dargestellt:

```
1  Ω' ← ∅
2  for O ∈ Ω|a[object] ∈ O
3      Γ' ← {γ ∈ Γ_δ|a[object] ∈ γ[area] ∧ γ[user] = a[user]}
4      if Γ' ≠ ∅ then
5          for γ ∈ Γ'
6              incrementVisitor(O, a[user], date(a[access]), γ[date])
7          end for
8      else if ∄a' ∈ A : a'[user] = a[user] then
9          incrementVisitor(O, a[user], date(a[access]), ε)
10     else
11         Ω' ← Ω' ∪ {O}
12     end if
13 end for
```

```
14
15  if  Ω' ≠ ∅  then
16      Γ'' ← {(O,t)|∃a' ∈ A : a'[user] = a[user] ∧ a'[object] ∈ O ∈ Ω
17                                                  ∧t = date(a'[access])}
18
19      Γ''' ← {(O,t) ∈ Γ''| ∄t' : t' > t ∧ (O,t') ∈ Γ''}
20      for  O ∈ Ω'
21          if  ∃(O,t) ∈ Γ'''  then
22              incrementVisitor(O, a[user], date(a[access]), t)
23          else
24              incrementVisitor(O, a[user], date(a[access]), ϵ)
25          end  if
26      end  for
27  end  if
```

Dabei ist die u.a. Methode *incrementVisitor* als Prozedur zu verstehen, die die eigentliche Datenanpassung der Statistik-Relationen vornimmt:

```
1   procedure  incrementVisitor(O, user, currentDate, previousDate)
2       if  lastDate ≠ date(a[access])  then
3           Γ_δ ← Γ_δ \ {(area, user, previousDate)}
4                   ∪{(area, user, currentDate)}
5
6           S' ← {s ∈ S|O = s[area] ∧ s[date] = currentDate}
7           if  S' = ∅  then
8               S ← S ∪ {(currentDate, O, 1)}
9           else
10              for  s ∈ S'
11                  s[visitors] ← s[visitors] + 1
12              end  for
13          end  if
14
15          K' ← {k ∈ K|O = k[area] ∧ k[date_1] = currentDate
16                                          ∧k[date_2] = previousDate}
17          if  previousDate = ϵ  then
18              //  skip
19          else  if  K' = ∅  then
20              K ← K ∪ {(currentDate, previousDate, O, 1)}
21          else
22              for  k ∈ K'
23                  k[returners] ← k[returners] + 1
24              end  for
25          end  if
26      end  if
```

```
27 end procedure
```

7.3.2 Abfrage des Werts der Tage $d_1 \dots d_n$ für Bereich $\mathcal{O} \in \Omega$

Um die Anzahl der Besucher eines Bereichs für einen gegebenen Zeitraum zu bestimmen, muss von der Summe der entsprechenden Tageswerte die Anzahl der Wiederkehrer abgezogen werden:

1 $\mathcal{S}' \leftarrow \{s \in \mathcal{S} | s[area] = \mathcal{O} \wedge s[date] \geq d_1 \wedge s[date] \leq d_n\}$
2 $\mathcal{K}' \leftarrow \{k \in \mathcal{K} | k[area] = \mathcal{O} \wedge k[date_1] \geq d_1 \wedge k[date_2] \leq d_n\}$
3 return $\sum_{s \in \mathcal{S}'}^{} s[visitors] - \sum_{k \in \mathcal{K}'}^{} k[returners]$

7.4 Besuche

Der Messwert „Besuche“ ist für die Vorberechnung relativ unkritisch. Im Wesentlichen verfolgt sie die Vorgehensweise für die Berechnung des Messwerts „Hits“.

Zentrale Relation für die Vorberechnung ist eine Relation, die zu jedem Tag und jedem Bereich festhält, wie oft dieser besucht wurde.

	DwellStatistic	$s \in \mathcal{S}$
$\mathbb{T}$:	DATE	$s[date]$
Ω :	AREA	$s[area]$
$\mathbb{N}$:	VisitCount	$s[visits]$

Abbildung 7.6: Tagesstatistik für Messwert Besuche

7.4.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$

Beim Hinzufügen eines Protokolleintrags muss geprüft werden, ob der zugehörige Besuch bereits für den Bereich gezählt wurde oder nicht. Ist dies nicht der Fall, so muss der entsprechende *visits*-Zähler einfach inkrementiert werden.

Zur optimierten Statistikberechnung können verschiedene Aspekte ausgenutzt werden. Als erstes kann ein Cache eingeführt werden, der zu jedem Benutzer festhält, welchen Bereich er wann zuletzt besucht hat. Einträge können dabei aus diesem Cache gelöscht werden, wenn die zugehörige Sitzung abgelaufen ist. Außerdem können nur solche Sitzungen noch aktiv sein, für die innerhalb der letzten maximalen Verweilzeit irgendwelche anderen Zugriffsdaten erfasst wurden. Zur Verarbeitung möchte ich daher eine Cache-Struktur einführen, die zu jedem Benutzer den Anfangszeitpunkt der zuletzt genutzten Sitzung sowie den Zeitpunkt des letzten zu dieser Sitzung erfassten Zugriffs festhält. Dieser Cache braucht dabei nur solche Einträge vorhalten, deren letzter Zugriff jünger als die maximale Seitenverweilzeit ist.

Ist δ die maximale Seitenverweilzeit, so gestaltet sich das Hinzufügen von Datensätzen wie folgt:

1 $\Gamma_\delta \leftarrow \{\gamma \in \Gamma_\delta | \gamma[access] > a[access] - \delta\}$
2 for $\mathcal{O} \in \Omega | a[object] \in \mathcal{O}$
3 if $\nexists \gamma \in \Gamma_\delta : \gamma[user] = a[user]$ then

	RecentVisitCache	$\gamma \in \Gamma_\delta$
Ω :	AREA	$\gamma[area]$
$\mathbb{U}$:	USER	$\gamma[user]$
$\mathbb{T}$:	SessionStart	$\gamma[start]$
$\mathbb{T}$:	LastAccess	$\gamma[access]$

Abbildung 7.7: Cache für letzten Bereichszugriff pro Benutzer

```
4            S' ← {s ∈ S | s[area] = O ∧ s[date] = date(a[access])}
5            if S' = ∅ then
6                S ← S ∪ {(date(a[access]), O, 1)}
7            else
8                for s ∈ S'
9                    s[visits] ← s[visits] + 1
10               end for
11           end if
12           Γδ ← Γδ ∪ {(O, a[user], a[access], a[access])}
13       else if ∄γ ∈ Γδ : γ[user] = a[user] ∧ γ[area] = O then
14           T ← {t ∈ 𝕋 | ∃γ ∈ Γδ : date(γ[start]) = t ∧ γ[user] = a[user]}
15           assertion : |T| = 1
16           for t ∈ T
17               S' ← {s ∈ S | s[area] = O ∧ s[date] = t}
18               if S' = ∅ then
19                   S ← S ∪ {(t, O, 1)}
20               else
21                   for s ∈ S
22                       s[visits] ← s[visits] + 1
23                   end for
24               end if
25               Γδ ← Γδ ∪ {(O, a[user], t, a[access])}
26           end for
27       end if
28   end for
29   for γ ∈ Γδ | γ[user] = a[user]
30       γ[access] ← a[access]
31   end for
```

7.4.2 Abfrage des Werts der Tage $d_1 \dots d_n$ für Bereich $\mathcal{O} \in \Omega$

Die Besuchszahl für einen gegebenen Zeitraum bestimmt sich aus der Summe der entsprechenden Tageswerte:

1 $\mathcal{S}' \leftarrow \{s \in \mathcal{S} | s[area] = \mathcal{O} \wedge s[date] \geq d_1 \wedge s[date] \leq d_n\}$
2 return $\sum_{s \in \mathcal{S}'} s[visits]$

7.5 Eingangs- und Zielpfade

In Abschnitt 7.7 werde ich zeigen, wie Kookurrenzen effizient vorberechnet werden können. Dabei wird eine Statistik-Relation eingeführt, die zu jedem Tag und zu je zwei Bereichen angibt, wie häufig am jeweiligen Tag ein direkter oder indirekter Übergang vom ersten Bereich zum zweiten stattgefunden hat. Aus dieser Relation können die hier geforderten Messwerte hinreichend effizient ermittelt werden, sodass an dieser Stelle keine Vorberechnung notwendig ist.

7.5.1 Abfrage des Werts der Tage $d_1 \dots d_n$ für Bereich $\mathcal{O} \in \Omega$

Zur Berechnung der Anzahl der Eingangspfade eines konkreten Bereichs muss in der statistischen Transitionenrelation $\mathcal{S}$ aus Abschnitt 7.7 gezählt werden, wie viele Objekte in den eingehenden Bereichen aufgeführt werden. Das Vorgehen gestaltet sich wie folgt:

1 return $|\{o \in \mathbb{O} | \exists s \in \mathcal{S} : o \in s[area_1] \wedge s[area_2] = \mathcal{O} \wedge s[date] \geq d_1 \wedge s[date] \leq d_2\}|$

Die Berechnung der Zielpfade passiert analog wie folgt:

1 return $|\{o \in \mathbb{O} | \exists s \in \mathcal{S} : o \in s[area_2] \wedge s[area_1] = \mathcal{O} \wedge s[date] \geq d_1 \wedge s[date] \leq d_2\}|$

7.6 Pfadlänge, Session-Starter und Session-Killer

Die Vorberechnung der Messwerte „Pfadlänge“, „Session-Starter“ und „Session-Killer“ passiert anhand der folgenden Relation:

	DwellStatistic	$s \in \mathcal{S}$
$\mathbb{T}$:	DATE	$s[date]$
Ω :	AREA	$s[area]$
$\mathbb{N}$:	OffsetSum	$s[offset]$
$\mathbb{N}$:	StartCount	$s[starts]$
$\mathbb{N}$:	KillCount	$s[kills]$

Abbildung 7.8: Tagesstatistik für Pfadpositionen

Dabei werden zu jedem Tag und zu jedem Bereich drei Zählwerte hinterlegt, die angeben, an welcher Position in allen Sitzungen der Bereich stand, wie häufig er direkt am Anfang stand und wie häufig er am Ende stand.

7.6.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$

Ähnlich dem Vorgehen zur Erfassung des Messwerts Besuche wird eine Caching-Struktur aufgebaut, die zu allen aktiven Benutzersitzungen das letzte aufgerufene Objekt und die aktuelle Sitzungsposition festhält:

	RecentVisitCache	$\gamma \in \Gamma_\delta$
$\mathbb{U}$:	USER	$\gamma[user]$
$\mathbb{T}$:	SessionStart	$\gamma[start]$
$\mathbb{T}$:	LastAccess	$\gamma[access]$
$\mathbb{O}$:	LastObject	$\gamma[object]$
$\mathbb{N}$:	Offset	$\gamma[offset]$

Abbildung 7.9: Cache für Pfadberechnungen

Beim Hinzufügen eines neuen Zugriffsprotokolleintrags muss für die Pfadsumme $offset$ einfach der Positionszähler des Benutzers hinzuaddiert werden und im Falle, dass die Pfadsumme den Wert 1 hat, der Zähler *start* inkrementiert werden. Während dem Hinzufügen eines jeden Eintrags wird immer davon ausgegangen, dass dieser der vorerst letzte Eintrag in einer Sitzung ist. Entsprechend wird der *kills*-Zähler für die Bereiche eines Zugriffsprotokolleintrags inkrementiert. Wird allerdings festgestellt, dass die Pfadsumme

einen Wert größer als 1 besitzt, so muss der *kills*-Zähler für alle zuvor als Endpunkt geltenden Bereiche wieder dekrementiert werden. Wie bereits erwähnt wird dazu im Cache Γ_δ festgehalten, welches das zuletzt angenommene Endobjekt war:

```
1  Γ_δ ← {γ ∈ Γ_δ | γ[start] > a[access] − δ}
2
3  o ← ε
4  t ← date(a[access])
5  n ← 1
6  for γ ∈ Γ_δ | γ[user] = a[user]
7      o ← γ[object]
8      n ← γ[offset]
9      t ← γ[start]
10 end if
11
12 for O ∈ Ω | a[object] ∈ O
13     S' ← {s ∈ S | s[area] = O ∧ s[date] = t}
14     if S' = ∅ then
15         S ← S ∪ {(t, O, 1, 1, 1)}
16     else
17         for s ∈ S'
18             s[offset] ← s[offset] + n
19             if n = 1 then
20                 s[starts] ← s[starts] + 1
21             end if
22             s[kills] ← s[kills] + 1
23         end for
24     end if
25 end for
26
27 if n = 1 then
28     Γ_δ ← Γ_δ ∪ {(a[user], t, a[access], a[object], 1)}
29 else
30     for s ∈ S | o ∈ s[area] ∧ s[date] = t
31         s[kills] ← s[kills] − 1
32     end for
33 end if
34
35 for γ ∈ Γ_δ | γ[user] = a[user]
36     γ[access] ← a[access]
37     γ[object] ← a[object]
38     γ[offset] ← γ[offset] + 1
39 end for
```

7.6.2 Abfrage des Werts der Tage $d_1 \ldots d_n$ für Bereich $\mathcal{O} \in \Omega$

Die Gesamtpfadlänge für einen gegebenen Zeitraum bestimmt sich aus der Summe der entsprechenden Tageswerte:

1 $\mathcal{S}' \leftarrow \{s \in \mathcal{S} | s[area] = \mathcal{O} \wedge s[date] \geq d_1 \wedge s[date] \leq d_n\}$
2 return $\sum_{s \in \mathcal{S}'} s[offset]$

Die Session-Starter-Häufigkeit bestimmt sich als Summe der entsprechenden *starts*-Zähler:

1 $\mathcal{S}' \leftarrow \{s \in \mathcal{S} | s[area] = \mathcal{O} \wedge s[date] \geq d_1 \wedge s[date] \leq d_n\}$
2 return $\sum_{s \in \mathcal{S}'} s[starts]$

Die Session-Killer-Häufigkeit bestimmt sich als Summe der entsprechenden *kills*-Zähler:

1 $\mathcal{S}' \leftarrow \{s \in \mathcal{S} | s[area] = \mathcal{O} \wedge s[date] \geq d_1 \wedge s[date] \leq d_n\}$
2 return $\sum_{s \in \mathcal{S}'} s[kills]$

7.7 Kookurrenzen

Eine erste Lösungsmöglichkeit zur Berechnung von indirekten Übergängen ist es, die Basisrelation nach Benutzern und Aufrufzeitpunkten zu sortieren und anschließend in einer linearen Iteration über die sortierte Liste eine Matrize der abgefragten Übergänge zu ermitteln. Der Berechnungsaufwand für diese Vorgehensweise liegt dann in der Aufwandsklasse $\mathcal{O}(n)$ oder schlechtestenfalls in der Aufwandsklasse $\mathcal{O}(ln(n) \cdot n)$, falls nicht bereits auf einen vorsortierten Datenbestand zugegriffen werden kann. Wie auch bei den zuvor betrachteten Messwerten bezieht sich n allerdings auf die Anzahl der Log-Einträge aus dem Betrachtungszeitraum und bezieht sich damit auf zu viele Datensätze.

Ich möchte zur Optimierung ein Verfahren vorschlagen, welches zwar theoretisch quadratischen Speicher- und Berechnungsaufwand besitzt, sich jedoch in der Praxis eher sublinear verhält und vor allem nicht direkt von der Anzahl der Elemente aus der Basisrelation abhängig ist. So bezieht sich die Komplexität hier auf die Anzahl der Objekte einer Site und nicht mehr auf die Anzahl aller Zugriffe.

Die grundlegende Idee der Vorberechnung von Übergängen ist es, zu je zwei Bereichen die tägliche Übergangshäufigkeit zu erfassen. Bei der Auswertung kann für zwei Bereiche direkt abgefragt werden, wie häufig vom ersten in den zweiten gewechselt wurde.

Natürlich muss man bei dieser Vorgehensweise berücksichtigen, dass eine High-Traffic-Site sehr viele Inhalte besitzen kann. Hat man beispielsweise eine Site, die aus 5000 unterschiedlichen Objekten besteht, wobei extern verlinkte Seiten ebenfalls mitgezählt werden müssen, so sind täglich bis zu 25 Millionen Übergangseinträge in dieser Statistik denkbar, wobei noch angesetzt ist, dass jedes Dokument in nur genau einem Bereich vorkommt.

Glücklicherweise ist keine größere Web-Site so aufgebaut, dass jede Seite jede andere verlinkt. Außerdem ist es auch sehr unwahrscheinlich, dass jeden Tag jeder Link jeder Seite aufgerufen wird, sodass sich diese Anzahl in der Praxis weiter reduziert. Tatsächlich ist es sogar unwahrscheinlich, dass in einer großen Site jede Seite täglich aufgerufen wird. In einem eigenen Projekt konnte ich feststellen, dass die entstehende Komplexität sogar sublinear zur Anzahl aller Objekte war. Tatsächlich kamen hier auf insgesamt 7786 Objekte, die 10121 Bereichen zugewiesen waren, nach eineinhalb Jahren täglich nur durchschnittlich 4413,24 Bereichsübergänge.

Der zentrale hierbei entstehende Vorteil ist jedoch die Komplexitätsunab-

hängigkeit von der Anzahl der durchgeführten Zugriffe. Für die Berechnung ist es aus Komplexitätssicht bei der Abfrage egal, ob an einem Tag 1000 Zugriffe oder 10.000.000 Zugriffe stattfanden. Tatsächlich ist davon auszugehen, dass bei kleinen Zugriffszahlen noch vergleichsweise viele Übergangseinträge erfasst werden, diese Anzahl bei großen Zugriffszahlen jedoch schnell eine gewisse Sättigung erreicht.

Für die Vorberechnung der Daten wird eine Relation benötigt, die zu jedem Tag und je zwei Bereichen angibt, wie häufig ein Übergang vom ersten Bereich in den zweiten stattgefunden hat:

	TransitionStatistic	$s \in \mathcal{S}$
$\mathbb{T}$:	DATE	$s[date]$
Ω :	SOURCEAREA	$s[area_1]$
Ω :	TARGETAREA	$s[area_2]$
$\mathbb{N}$:	UsageCount	$s[usage]$

Abbildung 7.10: Tagesstatistik für Messwert „Verweilzeit“

7.7.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$

Beim Hinzufügen eines neuen Zugriffseintrags wird ebenfalls eine dem vorangehenden Abschnitt vergleichbare Caching-Strategie eingesetzt. Dabei wird für jeden Benutzer und jede aktive Sitzung zwischengespeichert, welche Objekte der Benutzer bis zu diesem Zeitpunkt aufgerufen hat:

	RecentVisitCache	$\gamma \in \Gamma_\delta$
$\mathbb{U}$:	USER	$\gamma[user]$
Ω :	AREA	$\gamma[area]$
$\mathbb{T}$:	SessionStart	$\gamma[start]$
$\mathbb{T}$:	LastAccess	$\gamma[access]$

Abbildung 7.11: Cache für Sitzungspfadepro Benutzer

Für den neuen Zugriffseintrag wird dann dieser Benutzersitzungsinhalt abgefragt und zu jedem Eintrag dieser Sitzung der entsprechende Bereichsübergangszähler zum aktuellen Eintragsobjekt inkrementiert:

1 $\Gamma_\delta \leftarrow \{\gamma \in \Gamma_\delta | \gamma[access] > a[access] - \delta\}$
2
3 `t = date(a[start])`

```
4  for  O ∈ Ω|a[object] ∈ O
5       for  γ ∈ Γ_δ|γ[user] = a[user]
6            t = γ[start]
7            S' ← {s ∈ S|s[area_1] = γ[area] ∧ s[area_2] = O ∧ s[date] = t}
8            if  S' = ∅  then
9                 S ← S ∪ {(t, γ[area], O, 1)}
10           else
11                for  s ∈ S'
12                     s[count] ← s[count] + 1
13                end  for
14           end  if
15      end  for
16      Γ_δ ← Γ_δ ∪ {a[user], O, t, a[access]}
17 end  for
18
19 for  γ ∈ Γ_δ|γ[user] = a[user]
20      γ[access] ← a[access]
21 end  for
```

7.7.2 Abfrage des Werts der Tage $d_1 \dots d_n$ für Bereich $\mathcal{O}_1, \mathcal{O}_2 \in \Omega$

Um den eingehenden Kookurrenzwert für einen konkreten Zeitraum zu bestimmen, muss die Summe der zu den Bereichen gehörenden Übergänge gebildet werden:

```
1  S' ← {s ∈ S|s[area_1] = O_1 ∧ s[area_2] = O_2 ∧ s[date] ≥ d_1 ∧ s[date] ≤ d_n}
2
3  return  Σ_{s∈S'} s[usage]
```

Zur Bestimmung der ausgehenden Kookurrenzen müssen lediglich die beiden Bereiche vertauscht werden:

```
1  S' ← {s ∈ S|s[area_1] = O_2 ∧ s[area_2] = O_1 ∧ s[date] ≥ d_1 ∧ s[date] ≤ d_n}
2
3  return  Σ_{s∈S'} s[usage]
```

7.8 Klickpfad

Für eine Komplexitätsanalyse bei der Vorberechnung von Klick-Pfaden gelten im Wesentlichen die vorangegangenen Komplexitätsaspekte zur Kookurrenzenberechnung. Allerdings kommt hinzu, dass für die Vorberechnung von Klickpfaden theoretisch für jede mögliche Permutation von Bereichen ein Tagesstatistikwert entsteht. Die dabei entstehende Datenmenge wäre exorbitant.

Für die Vorberechnung von Klickpfaden möchte ich daher eine kleine Ungenauigkeit einbringen, die allerdings während einer praktischen Analyse nicht weiter von Belang ist. Dabei nehme ich Bezug auf die Arbeit von Andersen u.a. [3], die Klick-Pfade zur Analyse in Teilpfade zu zerlegen.

Diese Zerteilung möchte ich ausnutzen und gleichzeitig eine maximale Länge n für präzise analysierbare Klickpfade einführen. Die Statistikrelation für die Vorberechnung sieht dann wie folgt aus:

	ClickPathStatistic	$s \in \mathcal{S}$
$\mathbb{T}$:	DATE	$s[date]$
Ω :	1. AREA	$s[area_1]$
Ω :	2. AREA	$s[area_2]$
	$\vdots$	$\vdots$
Ω :	N. AREA	$s[area_n]$
$\mathbb{N}_0$:	AppearanceCount	$s[count]$

Abbildung 7.12: Tagesstatistik für Messwert Klick-Pfad

7.8.1 Zustandsübergang $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$

Beim Hinzufügen eines neuen Eintrags a in das Basisprotokoll müssen zunächst die letzten $n-1$ Einträge des Basisprotokolls ermittelt werden, die mit dem neuen Eintrag in einer Vorgängerbeziehung ($\prec$) stehen. Zusammen mit dem neuen Eintrag ergibt sich hieraus eine Objektaufrufssequenz. Mittels rekursiver Iteration über die Bereiche der Elemente dieser Sequenz werden dann die Einträge der Statistikrelation ermittelt, deren Häufigkeitszähler inkrementiert werden muss. Um bei der späteren Abfrage keine Anomalien zu erhalten, sollte sichergestellt werden, dass innerhalb eines Klickpfades kein Datumswechsel passiert. Dabei bietet es sich an, Klickpfade immer dem Tag zuzuordnen, an dem der Anfang des gesamten Klickpfades stand.

Zum Anpassen der Statistik wird zunächst eine Hilfsfunktion benötigt, die durch Rekursion die Sequenzen der notwendigen Bereichsübergänge ermittelt:

```
1  function ϵSequence(n)
2      if n = 0 then
3          return <>
4      else
5          return < ϵSequence(n − 1), ϵ >
6      end if
7  end function
8
9  function getAreaTransistions(a_n, n)
10     if n = 1 then
11         return {< O_n > |a_n[object] ∈ O_n ∈ Ω}
12     else
13         α ← ∅
14         for a_{n−1} ∈ A|a_{n−1} ≺ a_n
15             α ← α ∪ getAreaTransistions(a_{n−1}, n − 1)
16         end for
17         if α = ∅ then
18             α ← {ϵSequence(n − 1)}
19         end if
20         return {< O_1, ..., O_{n−1}, O_n > | < O_1, ..., O_{n−1} >∈ α
21                                          ∧a_n[object] ∈ O_n ∈ Ω}
22
23     end if
24 end function
```

Das Hinzufügen eines neuen Elements a in das Basisprotokoll gestaltet sich aus Sicht einer Klickpfadstatistik mit Maximallänge n wie folgt:

```
1  for < O_1, O_2, ..., O_n >∈ getAreaTransitions(a, n)
2      S' ← {s ∈ S| ∧ O_1 = s[area_1] ∧ O_2 = s[area_2] ... ∧ O_n = s[area_n]
3                                          ∧s[date] = date(a[access])}
4      if S' = ∅ then
5          S ← S ∪ {(date(a[access]), O_1, O_2, ..., O_n, 1)}
6      else
7          for s ∈ S'
8              s[count] ← s[count] + 1
9          end for
10     end if
11 end for
```

7.8.2 Abfrage des Werts der Tage $d_1 \dots d_n$

Die Nutzungshäufigkeit von Klickpfaden über die Bereiche $< \mathcal{O}_1, \dots, \mathcal{O}_k >$ kann effizient und präzise anhand der aufgeführten Statistikrelation berechnet werden, falls k kleiner oder gleich der maximal von der Statistik-Relation abgedeckten Pfadlänge ist. In diesem Fall wird der Messwert wie folgt ermittelt:

$$
\begin{aligned}
&1 \quad \mathcal{S}' \leftarrow \{s \in \mathcal{S} | s[area_1] = \mathcal{O}_1 \wedge s[area_2] = \mathcal{O}_2 \dots \wedge s[area_k] = \mathcal{O}_k \\
&2 \qquad\qquad\qquad\qquad\qquad\qquad \wedge s[date] \geq d_1 \wedge s[date] \leq d_n\} \\
&3 \quad \text{return} \sum_{s \in \mathcal{S}'}^{} s[count]
\end{aligned}
$$

Falls Klickpfade über Bereichssequenzen der Länge $n + k$ abgefragt werden, also länger als die maximal verzeichnete Klickpfadlänge sind, kann effizient eine obere Schranke für den entsprechenden Wert berechnet werden. Dabei gilt, dass eine längere Pfadsequenz höchstens so häufig aufgerufen werden kann wie sämtliche Teilpfadsequenzen. Die Berechnung eines Näherungswerts für den Statistikwert über die Bereichssequenz $< \mathcal{O}_1, \dots, \mathcal{O}_n, \dots, \mathcal{O}_{n+k} >$ sieht dann wie folgt aus:

$$
\begin{aligned}
&1 \quad \gamma \leftarrow \epsilon \\
&2 \quad \text{for} \;\; i \in \mathbb{N}_0 | i < k \\
&3 \qquad \mathcal{S}' \leftarrow \{s \in \mathcal{S} | s[area_i] = \mathcal{O}_i \wedge s[area_{i+1}] = \mathcal{O}_{i+1} \dots \wedge s[area_{n+k}] = \mathcal{O}_{n+k} \\
&4 \qquad\qquad\qquad\qquad\qquad\qquad \wedge s[date] \geq d_1 \wedge s[date] \leq d_n\} \\
&5 \qquad \gamma' \leftarrow \sum_{s \in \mathcal{S}'}^{} s[count] \\
&6 \qquad \text{if} \;\; \gamma = \epsilon \vee \gamma > \gamma' \;\; \text{then} \\
&7 \qquad\qquad \gamma \leftarrow \gamma' \\
&8 \qquad \text{end if} \\
&9 \quad \text{end for} \\
&10 \quad \text{return} \;\; \gamma
\end{aligned}
$$

Natürlich gilt dabei, dass mit größerem n auch die Präzision dieser Näherung zunimmt. Es sei jedoch angemerkt, dass n auch gleichzeitig den Polynomialkoeffizienten für die Speicher- und Berechnungskomplexität der Statistikrelation festlegt. Ist daher jedes Objekt in durchschnittlich 4 Bereichen, so müssen zur Unterstützung der maximalen Pfadlänge 4 beim Einfügen von Einträgen ins Basisprotokoll bis zu 256 Datenzeilen aktualisiert werden. Erfahrungsgemäß können aber bereits mit kurzen Pfadlängen fundamentale Analyseaspekte erkannt werden, sodass in den häufigsten Situationen kurze Pfadlängen bereits hinreichend sind.

Kapitel 8

Abschließende Betrachtungen

Die vorliegende Arbeit liefert ein Konzept für die Implementierung eines Web-Usage-Mining-Systems im High-Traffic-Bereich. Die Arbeit orientiert sich dabei an dem Prinzip, dass Analysten sich schrittweise relevanten Daten annähern und sich nach einem „Zoom"-Prinzip an die für sie situativ relevanten Aspekte herantasten. Analysten werden dabei in die Lage versetzt, jederzeit die Nutzung einer Web-Anwendung oder eines Web-Systems autark auszuwerten.

Neben der Vorstellung geeigneter Messwerte für eine Analyse wurde gezeigt, wie diese Daten anhand einer einfachen Basisrelation berechnet werden können, wie diese Daten erzeugt, gesammelt und die Messwerte durch angemessene Vorberechnung effizient abfragbar werden.

8.1 Aspekte zur Benutzerführung

Die vorliegende Arbeit betrachtet lediglich die Konzeption eines Usage-Mining-Systems. Zentral ist dabei, dass jene Aspekte durchleuchtet werden, die im High-Traffic-Bereich zu besonderen Problemen führen. Wesentliche Aufgabe für die Implementierung ist daher der Entwurf einer angemessenen Benutzeroberfläche für Analysten. Da ergonomische Betrachtungen zur Oberflächengestaltung den Rahmen dieser Arbeit sprengen würden, seien an dieser Stelle nur einige Ideen zur Benutzerführung dargestellt:

Vergleichende Sichten Zur Aufbereitung der Informationen sollte der Analyst prinzipiell nicht mit einzelnen Messwerten hantieren. Vielmehr sollten ihm vergleichende Übersichten über zusammenhängende Messwerte geboten werden. Beispielsweise sollte es möglich sein, den zeitlichen Verlauf eines konkreten Messwerts für einen konkreten Bereich in einem Graphen zu betrachten. Umgekehrt sollte es aber auch möglich sein, einen konkreten zeitlich fixierten Messwert für unterschiedliche Bereiche graphisch oder tabellarisch gegenüberzustellen.

Analysten-spezifische „Scopes“ Jeder Analyst sollte die Möglichkeit haben, die für ihn zentralen Bereiche in speziellen Scopes zusammenzuführen und diese zu speichern. Auf diese Weise wird den Analysten die Möglichkeit gegeben, komplexere Gegenüberstellungen wieder zu nutzen.

Simplexe Filterung Beim Heransuchen an Informationen bietet es sich an, immer eine Filterdimension zu verändern und alle anderen unverändert zu lassen, sodass ein Analyst sich schrittweise und vor allem systematisch an relevante Informationen annähern kann.

Organisation von Bereichen Zur Navigation über Bereiche sollte prinzipiell an eine hierarchische Herangehensweise gedacht werden, da diese dem „Zoom“-Prinzip sehr nahe kommt. So ist es denkbar, die Bereiche in einer mehrfach verwurzelten Baumstruktur anzuordnen, derart, dass auf Ebene der Blätter die einzelnen Objekte stehen und die Wurzelknoten weitestgehend unabhänigigen Themenbereichen entsprechen.

Lupenartige Zeitraumauswahl Für die Auswahl von Zeiträumen sollte auch überlegt werden, inwieweit ein Mechanismus wie beispielsweise die Lupenfunktion eines Bildverarbeitungsprogramms, zum Einsatz kommen kann. So kann man sich vorstellen, dass ein Analyst die Sicht in einem Verlaufsgraphen durch Bereichsmarkierung auf einen kleineren Zeitraum beschränkt oder ein in einem Graphen dargestellter Aggregat-Wert durch anklicken in seine Bereichszusammensetzung auf nächster Ebene zerlegt wird.

8.2 Authorisierter Zugriff

Eine weitere Aufgabe für die Implementierung ist die Betrachtung von Techniken für einen geschützten Informationszugang. So sind Analyse-Daten potentiell wertvolle und schützenswerte Unternehmensinformationen, die im Allgemeinen nicht frei zugänglich sein dürfen. Bei der Implementierung eines Analysesystems muss somit eine geeignete Authentifikationstechnik und auch ein angemessenes Authorisationskonzept eingeführt werden. Für ein Authorisationskonzept sollte dabei auch berücksichtigt werden, dass in großen Unternehmen durchaus oder sogar konkurrierende Fachbereiche existieren können. Es sollte daher möglich sein, für Analysten festlegen zu können, welche Bereiche diese analysieren dürfen und welche nicht.

8.3 Stetiges Anwachsen des Datenvolumens

Im Laufe der Zeit werden die anfallenden Daten eines Analysesystems zunehmend umfangreicher. Die wesentliche Datenmasse fällt dabei in der Basisrelation für Detailanfragen an. Die Bedeutung von Detail-Informationen sinkt jedoch andererseits mit ihrem Alter. So ist es sehr unwahrscheinlich, dass ein Analyst in Erfahrung bringen möchte, welche Seiten Benutzer X-Y vor Jahren einmal aufgerufen hat.

Außerdem wurden in Kapitel 7 Verfahren zur Datenverdichtung vorgestellt, deren Datenmasse wesentlich kleiner ist als das entsprechende vollständige Basisprotokoll.

In einer Implementierung sollte daher vorgesehen werden, dass sehr alte Daten der Basisrelation gelöscht werden, sodass dann für lange zurückliegende Zeiträume lediglich verdichtete Informationen abfragbar bleiben.

Falls die Daten aus den in Kapitel 7 vorgestellten Datenverdichtungen zu umfangreich werden, können auch die verdichteten Daten weiter verdichtet werden. Dabei muss lediglich eine zusätzliche Statistik eingeführt werden, die größere Zeitintervalle zusammenfasst.

8.4 Fazit

Die Arbeit selbst konzipiert ein Usage-Mining-System für High-Traffic-Anwendungen. Dabei werden geeignete Strategien zur Informationsaufbereitung vorgestellt. Letztendlich bleibt es Aufgabe einer Implementierung diese Strategien in angemessene Dialoge zu „gießen“ und einem Analysten den ergonomischen Zugang zu den verfügbaren Informationen zu bieten.

Ich denke, dass mit dieser Arbeit eine robuste Basis für die Erstellung eines Analysesystems geschaffen wurde. Den zentralen Aspekt der Berücksichtigung von Anforderungen im High-Traffic-Bereich wird eine Problemklassenverlagerung entgegengestellt, die das System in seinem Laufzeitverhalten bei Analysen unabhängig von der Anzahl der erfassten Seitenzugriffe macht. Außerdem bietet die in Kapitel 6 behandelte Datennormalisierung die Möglichkeit leicht weitere Datenquellen anzubinden, wodurch die Beschränkung auf Analysen von Web-Systemen umgangen werden kann. So ist es denkbar, dass das System durch alleinige Anpassung der Datenquellen auch für das Studium andersartiger zugriffsorientierter Verhaltensmuster einsetzbar ist.

Inwieweit die vorgeführten Messwerte für eine pertinente Auswertung hinreichend sind und damit das System selbst den Anforderungen des Einsatzgebiets gerecht wird, ist natürlich anwendungsfallspezifisch. Dennoch zeigt diese Arbeit den Umfang des gebotenen Informationsgehalts und gibt damit Grund zur Annahme, dass das konzipierte System in vielen Situationen ein geeignetes Hilfsmittel für Analysen bildet.

Literaturverzeichnis

[1] Aboba, Bernard und John Vollbrecht: *Proxy Chaining and Policy Implementation in Roaming*. Nummer 2607 in *Request for Comments*. Network Working Group, Juni 1999. www.ietf.org/rfc/rfc2965.txt.

[2] Achour, Mehdi, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Philip Olson, Georg Richter, Damien Seguy, Jakub Vrana et al.: *PHP Handbuch*. The PHP Group, 2005.

[3] Andersen, Johnny, Anders Giversen, Allan H. Jensen, Rune S. Larsen, Torben Bach Pedersen und Janne Skyt: *Analyzing Clickstreams Using Subsessions*. Technischer Bericht 00-5001, Dept. of Computer Science, Aalborg University, Aalborg, Oktober 2000.

[4] Armstrong, Eric, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase und Eric Jendrock: *The J2EE 1.4 Tutorial*. J2EE 1.4 SDK. Sun Microsystems, Dezember 2005. java.sun.com/j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf.

[5] Bekman, Stas und Eric Cholet: *Practical mod_perl*. O'Reilly and Associates, Mai 2003. modperlbook.org.

[6] Berkhin, Pavel: *Survey Of Clustering Data Mining Techniques*. Technischer Bericht, Accrue Software, San Jose, CA, 2002. citeseer.ist.psu.edu/berkhin02survey.html.

[7] Berners-Lee, Tim, Roy T. Fielding und Henrik Frystyk Nielsen: *Hypertext Transfer Protocol – HTTP/1.0*. Nummer 1945 in *Request for Comments*. Network Working Group, Mai 1996. www.ietf.org/rfc/rfc1945.txt.

[8] BMI (Herausgeber): *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz*. Bundesministe-

rium des Innern und Bundesministerium für Arbeit und Sozialordnung, Juli 2002. www.webakus.de/textlinks/bitv.pdf.

[9] Braun, Herbert: *Neue Firefox-Version protokolliert Klicks.* Heise News, Januar 2006.

[10] Connolly, Dan und Tim Berners-Lee: *Naming and Addressing: URIs, URLs, ...* W3C Empfehlung, 1997. www.w3.org/Addressing/URL/4_Recommentations.html.

[11] Czeponik, Thomas et al.: *Session Hijacking.* In: *Wikipedia - Die freie Enzyklopädie.* Wikimedia Foundation, Juli 2006. Version: 2006-07-25T14:19.

[12] Daigle, Leslie: *WHOIS Protocol Specification.* Nummer 3912 in *Request for Comments.* Network Working Group, San Diego, September 2004. www.ietf.org/rfc/rfc3912.txt.

[13] Dierks, Tim und Christopher Allen: *The TLS Protocol Version 1.0.* Nummer 2246 in *Request for Comments.* Network Working Group, Januar 1999. www.ietf.org/rfc/rfc2246.txt.

[14] Dithardt, Dirk: *Squid Administrationshandbuch zum Proxyserver*, Kapitel Grundlagen und Prinzip eines Proxys, Seiten 3 – 22. dpunkt.verlag GmbH, Heidelberg, 1 Auflage, Oktober 2003. ISBN 3-89864-246-1.

[15] Droms, Ralph: *Dynamic Host Configuration Protocol.* Nummer 2131 in *Request for Comments.* Network Working Group, Lewisburg, März 1997. www.ietf.org/rfc/rfc2131.txt.

[16] Fielding, Roy T., Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul Leach und Tim Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1.* Nummer 2616 in *Request for Comments.* Network Working Group, 1999. www.ietf.org/rfc/rfc2616.txt.

[17] Franks, John, Phillip Hallam-Baker, Jeff Hostetler, Scott Lawrence, Paul Leach, Ari Luotonen und Lawrence C. Stewart: *HTTP Authentication: Basic and Digest Access Authentication.* Nummer 2617 in *Request for Comments.* Network Working Group, Juni 1999. www.ietf.org/rfc/rfc2617.txt.

[18] Freier, Alan O., Philip Karlton und Paul C. Kocher: *The SSL Protocol Version 3.0*. Netscape – Transport Layer Security Working Group, November 1996. wp.netscape.com/eng/ssl3/draft302.txt.

[19] Guisset, Fabian (Herausgeber): *Gecko DOM Reference*. Mozilla Organisation, Dezember 2002. www.mozilla.org/docs/dom/dom.pdf.

[20] Gundavaram, Shishir: *CGI Programmierung im World Wide Web*. O'Reilly International Thomson Verlag, 1 Auflage, 1996. ISBN: 3-930673-43-6.

[21] Hall, Marty: *Tutorial on Servlets and JSP*, Kapitel Session Tracking. Prentice Hall PTR, 1999. ISBN 0-13-089340-4.

[22] Hickson, Ian (Herausgeber): *Web Applications 1.0*, Working Draft The a element. What WG, Juli 2006. www.whatwg.org/specs/web-apps/current-work.

[23] Horwat, Waldemar (Herausgeber): *Standard ECMA-262 – ECMAScript Language Specification*. ECMA International, Geneva, 3 Auflage, Dezember 1999.

[24] IEEE, Computer Society (Herausgeber): *Logical Link Control*. Nummer 802.2 in *IEEE Standard*. IEEE Press, New York, R2003 Auflage, May 1998. ISBN 1-55937-959-6.

[25] Johns, Mike St.: *Authentication Server*. Nummer 931 in *Request for Comments*. Network Working Group, January 1985. www.ietf.org/rfc/rfc931.txt.

[26] Kleinberg, Jon M.: *Authoritative sources in a hyperlinked environment*. Journal of the ACM, 46(5):604–632, 1999.

[27] Kristol, Dave und Lou Montulli: *HTTP State Management Mechanism*. Nummer 2965 in *Request for Comments*. Network Working Group, October 2000. www.ietf.org/rfc/rfc2965.txt.

[28] Levene, Mark, Jose Borges und George Loizou: *Zipf's law for web surfers*. Knowledge and Information Systems, 3:120 – 129, 2001.

[29] Luehe, Jan (Herausgeber): *JSR-000245 JavaServer Pages TM 2.1*. Nummer 245 in *Java Specification Requests*. Sun Microsystems, Inc., Mai 2006.

[30] Luotonen, Ari und Henrik Frystyk Nielsen: *Logging Control In W3C httpd.* W3 Empfehlung, 1995. http://www.w3.org/Daemon/User/Config/Logging.html.

[31] Mayr, Philipp: *Entwicklung und Test einer Logfilebasierten Metrik zur Analyse von Website Entries am Beispiel einer akademischen Universitäts-Website.* Berliner Handreichungen zur Bibliothekswissenschaft 129, Institut für Bibliothekswissenschaft, Berlin, 2004. ISSN 14 38-76 62.

[32] Mortazavi-Asl, Behzad: *Discovering And Mining User Web-Page Traversal Patterns.* Master's Thesis, Simon Fraser University, Burnaby, April 2001.

[33] NCSA, CGI Group (Herausgeber): *Documentation for the Common Gateway Interface (CGI)*, Kapitel CGI Environment Variables. University of Illinois, Urbana - Champaign, März 1996. hoohoo.ncsa.uiuc.edu/cgi.

[34] Oldenburg, Heike: *Analysen von Webserver-Logfiles zur Kategorisierung des Navigationsverhaltens von Nutzern.* Magisterarbeit, Institut für Bibliothekswissenschaft, Berlin, 2003.

[35] Open Market, Inc. (Herausgeber): *FastCGI: A High-Performance Web Server Interface.* Open Market, Inc., April 1996. www.fastcgi.com/devkit/doc/fastcgi-whitepaper/fastcgi.htm.

[36] Postel, Tom (Herausgeber): *Internet Protocol.* Nummer 791 in *Request for Comments.* Information Sciences Institute, California, September 1981. www.ietf.org/rfc/rfc791.txt.

[37] Rechenberg, Peter und Gustav Pomberger (Herausgeber): *Informatik Handbuch*, Kapitel Logik. Carl Hanser Verlag, München, 2 Auflage, 1999. ISBN 3-446-19601-3.

[38] Rescorla, Eric: *HTTP Over TLS.* Nummer 2818 in *Request for Comments.* Network Working Group, Mai 2000. www.ietf.org/rfc/rfc2818.txt.

[39] Senie, Daniel: *Network Address Translator (NAT)-Friendly Application Design Guidelines.* Nummer 3235 in *Request for Comments.* Network Working Group, Bolton, Januar 2002. www.ietf.org/rfc/rfc3235.txt.

[40] Shahabi, Cyrus und Farnoush Banaei-Kashani: *Efficient and Anonymous Web-Usage Mining for Web Personalization*. In: *Special Issue on Data Mining*, Band 15 der Reihe *INFORMS Journal on Computing*, Seiten 123–147, Los Angeles, Frühjahr 2003. University of Southern California.

[41] Vossen, Gottfried: *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*, Seiten 316–320. Oldenbourg Wissenschaftsverlag GmbH, München, 4 Auflage, 2000. ISBN 3-486-25339-5.

Versicherung

„Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst, alle aus anderen Werken wörtlich oder sinngemäß entnommenen Stellen unter Angabe der Quelle als Entlehnung kenntlich gemacht und andere als die angegebenen Hilfsmittel nicht benutzt habe.“

Bad Ems, 10. September 2006