

Vorhersage von Benutzeranforderungen im WWW*

Ernst-Georg Haffner, Uwe Roth, Thomas Engel, Christoph Meinel

Institut für Telematik

Bahnhofstr. 30-32

54292 Trier

{haffner,roth,engel,meinel}@ti.fhg.de

Zusammenfassung

Der Daten- und Informationsfluss im Internet wächst rasant an und wird zunehmend komplexer. Die gestiegenen Anforderungen an den effizienten und unkomplizierten Gewinn von Informationen aus dem Netz der Netze setzen komplexe und ausgeklügelte Methoden auf Server- und Clientseite voraus, um aus einer Vielzahl heterogener und sich ständig ändernder Datenquellen möglichst schnell jene zu ermitteln und aufzubereiten, die der Benutzer angefordert hat.

Neben Anstrengungen zur Verbesserung von Flexibilität und Skalierbarkeit von Datenservern im Internet gewinnen zunehmend Verfahren zur Vorhersage von Benutzeranforderungen an Bedeutung. Während sich die Vorhersage von HTML-Seiten durch Markov-Modelle abbilden lassen, erfordern reine Datenrequests allgemeinere Modellfelder zur adäquaten Abbildung künftiger Anforderungen. So kann eine aufwendige Datenermittlung aus diversen Datenbanken zusammen mit der zugehörigen Informationsaufbereitung bereits stattfinden, bevor ein Benutzer sie anfordert.

1. Einleitung

Die Verbreitung des World Wide Web (WWW) vollzieht sich mit rasanter Geschwindigkeit. Der hierbei stattfindende Informationsaustausch zwischen Millionen von Clients (Browsern) und den Webservern gestaltet sich zunehmend komplexer. Neben statischen HTML-Seiten [HTML] gewinnen auch dynamisch aufbereitete Daten stärker an Bedeutung. Das Standardverfahren zur Ermittlung dieser Informationen geschieht über das Common-Gateway-Interface (CGI), wobei auf der Serverseite Skripten ausgeführt werden, die in der Regel Datenbankabfragen vornehmen [BG98]. Die Ergebnisse solcher Requests werden von den Webservern in die für Browser verständliche Sprache HTML umgewandelt.

Inzwischen verbreiten sich auch solche Anwendungen im Internet, deren Kommunikation – vorzugsweise über Java-Applets [Java],[FAR98] – unmittelbar über einen eigens hierfür konzipierten Datenserver erfolgt. Hierbei ist die Umsetzung der Daten in HTML nicht mehr erforderlich, ja nicht einmal erwünscht! Insbesondere Anwendungen, die mit großen und komplexen Datenmengen operieren, wie etwa Börseninformationsdienste, machen von diesem Verfahren exzessiv gebrauch.

Neben einer strukturierten, modularen Architektur zeichnen sich solche Server durch eine Reihe ausgeklügelter Verfahren aus, um den Anforderungen an eine vielschichtige, sich ändernde Menge an vorhandenen Informationsquellen und Anforderungstypen der Clients gerecht zu werden.

Eine Idee, die im Bereich des Compilerbaus und ebenso im VLDB¹-Design bereits seit langem Früchte trägt, wird seit dem Aufblühen des WWW auch für den Einsatz im Internet heftig und

* In Proc. „Lernen, Wissensentdeckung und Adaptivität“, GI-Fachtagung, LWA99, Magdeburg, 1999, S. 283-294

¹ Very Large Data-Base, z.B. [KW98],[KW97]

kontrovers diskutiert: die *Vorhersage von Benutzeranforderungen*. Das primäre Ziel des Einsatzes derartiger Vorhersage- oder *Prediction*-Algorithmen ist die Verminderung der durch den Anwender wahrgenommenen Verzögerungszeit, also dem Antwortzeitverhalten der Anwendung. Damit findet wiederum ein in der Informatik hinlänglich angewendetes Verfahren statt: die Umwandlung von Raum (Speicherplatz auf lokalen oder entfernten Rechensystemen) in Zeit (Wartezeit des Benutzers). Aber hier sollte auch der Blick für die durchschnittliche Wartezeit **aller** Benutzer des Netzes nicht aus dem Auge verloren werden. Während der Einsatz von Prediction im Compilerbau fast keine negativen Nebeneffekte besitzt, gilt dies im Bereich des WWW nicht mehr uneingeschränkt. Wir werden im folgenden eine Reihe von Sicherheitsvorkehrungen kennen lernen, um die schädlichen Folgen von nicht zutreffender Vorhersage zu reduzieren.

Der weitere Aufbau dieses Artikels gliedert sich zunächst in einen Abschnitt zum Ausarbeiten der Grundlagen für die Anforderungsvorhersage. Anschließend wird in „3. *Kostenfunktionen und Schwellwerte*“ aufgezeigt, wann sich Prediction überhaupt „auszahlt“. Die grundsätzliche Methodik zur zahlenmäßigen Feststellung der erforderlichen Vorhersagewerte findet sich unter „4. *Ermittlung der Anforderungswahrscheinlichkeiten*“. An dieser Stelle werden ebenfalls einige wesentliche Unterschiede in der Vorhersage von HTML-Dokumenten im Vergleich zu allgemeinen Datenmengen erarbeitet. Den möglichen Stolperfallen im Umgang mit Prediction widmet sich der Abschnitt „5. *Kritische Aspekte der Anforderungsvorhersage*“, der nach den zuvor aufgezeigten Chancen die bedeutendsten Risiken aufdeckt. Eine Zusammenfassung mit einem Ausblick auf künftige Forschungsschwerpunkte schließen das Papier ab.

2. Grundlagen der Anforderungsvorhersage

Die Theorie der Prediction, der Anforderungsvorhersage, setzt einige – in der Literatur zumeist unausgesprochene - Axiome oder Thesen voraus, deren Gültigkeit gewährleistet sein muss und ohne die keine Performanceverbesserung mittels Vorhersagetechniken zu erwarten ist.

Die hier angeführte Argumentation ist zunächst nicht auf eine bestimmte Struktur der Datensätze oder Dokumente auf der Serverseite festgelegt. Wir werden im folgenden von der Anforderung von *Datenmengen* sprechen, um nicht durch die Verwendung eines der Begriffe *Datensatz* oder *Dokument* bereits einen Datenserver bzw. einen Webserver zu implizieren.

Die erste und wichtigste Voraussetzung fördert die Grundidee der Prediction-Theorie für das Internet zutage.

<p>Voraussetzung 1: Die Wahrscheinlichkeit dafür, dass eine Datenmenge (künftig) angefordert wird, lässt sich aus dem bisherigen Anforderungsverhalten berechnen.</p>
--

Diese Aussage ist keineswegs trivial und stellt dennoch die Grundvoraussetzung aller weiterer Bemühungen um die Anforderungsvorhersage dar. Es ist zu beachten, dass der These 1 nicht zu entnehmen ist, ob das Anforderungsverhalten eines bestimmten Benutzers nur aus seinem eigenen oder aber dem Anforderungsverhalten vieler anderer Benutzer abzuleiten ist. Außerdem wird Wert darauf gelegt, dass ein Zusammenhang zwischen vergangenem und zukünftigen Verhalten eines Anwenders nicht nur existiert, sondern auch berechenbar ist.

Insbesondere wird nicht festgelegt, ob eine solche Vorhersage auf der Clientseite oder aber auf der Serverseite stattfinden sollte. Wir werden im Abschnitt (5.3) hierauf näher eingehen.

Ebenso wichtig wie die erste Voraussetzung zur Vorhersage von Benutzeranforderungen ist die folgende.

Voraussetzung 2: Die Ergebnisdaten vorhergesehener Anforderungen lassen sich bereits vor der Anforderung berechnen.

Einige der Prediction-Algorithmen in der Literatur ignorieren die zweite Grundvoraussetzung. Andere setzen den Fall verfrüht berechneter und zum Zeitpunkt des Aufrufs nicht mehr gültiger Datenmengen mit dem Aktualitätsproblem von Proxy-Cache-Servern gleich. Eine Diskussion der Vorhersagetheorie im Zusammenhang mit solchen als Mittler zwischen Client und Server geschalteten Proxy-Systemen findet sich in [CDF98] und [KLM97].

Die dritte Voraussetzung stellt den Zusammenhang zwischen den beiden vorangegangenen her. Ihre Gültigkeit schränkt die möglichen Anwendungsgebiete von Prediction-Verfahren stark ein.

Voraussetzung 3: Durch die Vorhersage von Benutzeranforderungen und die vorzeitige Berechnung von angeforderten Daten steigt die subjektive Leistungsfähigkeit der individuellen Client-Server Kommunikation.

Hierbei wird durch die Verwendung des Adjektivs „subjektiv“ klargestellt, dass der Schwerpunkt der Prediction-Forschung darin besteht, die wahrgenommene Wartezeit aus Sicht des Benutzers zu verbessern. Dieser Fortschritt darf jedoch nicht etwa auf Kosten der „restlichen“ Internet-Gemeinde gehen, indem unzählige überflüssige und nicht angeforderte Datenmengen die objektive Leistungsfähigkeit des Gesamtsystems verringern. Einige Verfahren haben in der Vergangenheit jedoch die globalen Auswirkungen der Prediction in ihrem Ansatz nicht adäquat berücksichtigt, sondern lediglich isolierte, auf spezielle Anwendungen beschränkte Effizienzsteigerungen angestrebt. Eine derartiger Tradeoff kann über einen gewissen Zeitraum für bestimmte Benutzergruppen Zeitvorteile verschaffen, verbietet sich jedoch als allgemeines Paradigma im Internet: der Netzverkehr würde schließlich kollabieren.

Lediglich bestimmte Teilbereiche der Internet-Kommunikation erfüllen Voraussetzungen, wie sie oben gefordert werden. Und nur auf solche Bereiche, für die alle drei zutreffen, lassen sich vernünftigerweise Vorhersageparadigmen anwenden. Die möglichen Anwendungen der Prediction sind in diesen Bereichen jedoch enorm. Neben frühzeitiger Kalkulation aufwendiger Rechenoperationen lassen sich ebenso komplexe Datenretrievals aus unterschiedlichen Datenbanken bereits anstoßen, bevor der Benutzer diese anfordert und ergo darauf wartet. Darüber hinaus können die Ergebnisse solcher Berechnungen bereits über das Netz an den Client gesendet werden, so dass auf der Seite des Benutzers der Eindruck entstehen kann, die jeweiligen Datenmengen lägen lokal vor (*Prefetching*).

Gerade dort, wo der potentielle, individuelle Nutzen der Prediction am höchsten ist, nämlich im Bereich des *Prefetching*, werden auch die Risiken maximiert. Hier führen unzutreffende Vorhersagen zu einer negativen Gesamtbilanz und besonders hier ist eine sehr vorsichtige Herangehensweise geboten, die wir *defensive Strategie* nennen.

3. Kostenfunktionen und Schwellwerte

Um die Effekte der Prediction zahlenmäßig erfassen zu können, wird zumeist ein Kostenmodell vorgeschlagen. Wie hoch sind die zu erwartenden Kosteneinsparungen im Falle positiver Vorhersage? Welche negativen Kosten schlagen bei unzutreffenden Vorhersagen zu buche? Was kostet das Vorhersagemodul (ob auf Client- oder Serverseite) selbst? Wie sieht somit die gesamte Kostenbilanz der Prediction aus? In diesem Abschnitt werden zunächst nur individuelle

Einsparungen behandelt, die Folgen für die Gesamtbelastung werden dagegen in Kapitel 5 diskutiert.

Um Kosten kalkulieren zu können, müssen die Vorhersagen in einen zeitlichen Bezugsrahmen zur Anforderung gebracht werden. Dies lässt sich auch inhaltlich rechtfertigen, weil die meisten Informationen, die aus den übermittelten Daten gewonnen werden können, dazu tendieren, nach einer gewissen Zeitspanne ungültig oder wertlos zu werden.

Eine frühestmögliche Vorhersage auf Serverseite könnte etwa beginnen, sobald sich Daten, die in den jeweiligen Berechnungen eine Rolle spielen, verändern – lange bevor ein Client eine Verbindung aufgebaut hat. Diese Art der Prediction führt zur *Precalculation* auf der Serverseite. Weiterhin werden wir das *Prefetching* untersuchen, das eine vorzeitige Übertragung von Daten zur Rezipientenseite vorsieht - in der Regel von der Clientseite initiiert. Schließlich werden wir auch das *Huckepack-Verfahren* ansprechen, bei dem der Server Daten an den Client zusammen mit einer angeforderten Sendung übermitteln kann.

Für den zeitlichen Gewinn des Benutzers hat sich als Maß die *Sitzung* etabliert, die zwar technisch nicht belegt werden kann², jedoch dennoch die solideste und am häufigsten verwendete Zeiteinheit für die Prediction darstellt.

Eine Sitzung s sei eine Folge von Anforderungen innerhalb einer Zeit T . Schechter nennt hier beispielsweise eine Zeit von 30 Minuten [SKS98]. Wir werden im folgenden auch eine Reihe von Anforderungen auf der Serverseite *Sitzung* nennen, selbst wenn diese von verschiedenen Benutzern vorgetragen werden.

3.1 Precalculation

Die Funktion γ_s zur Kostenermittlung von Berechnungen innerhalb einer Session s auf Serverseite lässt sich allgemein ansetzen mit:

$$\gamma(s) = \sum_{i=1}^A (\gamma_R(d_i) + \gamma_Z(d_i)) \cdot p_i \quad (1)$$

wobei A Berechnungen der Datenmengen d_1, d_2, \dots, d_A mit der jeweiligen Wahrscheinlichkeit p_i innerhalb s erfolgen.

Die Kosten auf der Serverseite lassen sich in zeitlich *optimierbare* Kosten γ_Z und in zwingend erforderliche Ressourcen-Kosten γ_R unterteilen. Wenn beispielsweise Berechnungen von Zeitintervallen mit hoher Systemauslastung auf solche mit niedrigerer verlegt werden, können die Resultate schneller erwartet werden, die zugehörigen Algorithmen verbrauchen dennoch dieselben Systemressourcen wie beispielsweise Speicherplatz. Eine Precalculation kann deshalb auch nur γ_Z reduzieren, während sich γ_R nicht verbessern lässt. Es stellt sich jedoch die Frage, wie sich die Werte für die einzelnen Typen von Kosten ermitteln lassen. Im Sinne einer defensiven Strategie sollten die Abschätzungen eher pessimistisch erfolgen.³

Die Kostenfunktion für Berechnungen mit Einsatz von Anforderungsvorhersagen ergeben sich dagegen zu:

$$\gamma(s) = \sum_{i=1}^B \gamma_R(d_i) + \sum_{i=1}^A \gamma_P(d_i) + \sum_{i=B+1}^A (\gamma_R(d_i) + \gamma_Z(d_i)) \cdot p_i \quad (2)$$

wobei $A \geq B$ und ohne Beschränkung der Allgemeinheit die ersten B Datenmengen d_1, d_2, \dots, d_B vorhergesehen werden konnten. Der Anteil an Ressourcen-Kosten γ_R ist in (2) gegenüber (1) gestiegen, weil – unabhängig von der Trefferquote – im Voraus berechnete Anforderungen stets

² Zumindest die Serverseite kann mittels des WWW-Standard-Protokolls [HTTP] nicht erkennen, ob ein Benutzer zwischen zwei Requests ununterbrochen „online“ gewesen ist oder nicht.

³ Aktuelle Überlegungen gehen ebenfalls dahin, durch eine Verfeinerung des Ansatzes die Kostenwerte dynamisch ermitteln zu lassen.

Systemressourcen beanspruchen. Der Anteil an optimierbaren Kosten für die Elemente d_{B+1}, \dots, d_A wird dagegen als vernachlässigbar angenommen.⁴ Dagegen findet sich nunmehr ein neuer Kostenanteil γ_P , der für die Berechnung der Vorhersage selbst aufgewendet werden muss („Buchhaltungskosten“).

Da die Precalculation nur Sinn macht, wenn (2) für jede Datenmenge d_i mit $i \in \{1, \dots, B\}$ kleiner als (1) wird, ergibt sich als *Schwellwert* für die Vorhersagewahrscheinlichkeit einer einzelnen Datenmenge d_i :

$$p_i > \frac{\gamma_R(d_i) + \gamma_P(d_i)}{\gamma_R(d_i) + \gamma_Z(d_i)} = \frac{1 + \frac{\gamma_P(d_i)}{\gamma_R(d_i)}}{1 + \frac{\gamma_Z(d_i)}{\gamma_R(d_i)}} = \frac{1 + S_o(d_i)}{1 + S_f(d_i)} \quad (3)$$

Wobei S_o als Quotient des *Prediction-Overheads* im Verhältnis zur Komplexität und dem Ressourcenaufwand der zu ermittelnden Daten den Sinn der Precalculation auf komplexe Aufgabenstellungen beschränkt. S_f als ein Maß für die *Flexibilität* des Systems legt eine Precalculation nur dann nahe, wenn hier das Verhältnis von optimierbaren Kosten zu festen Ressourcen-Kosten recht hoch ist. Insbesondere muss stets gelten: $S_o < S_f$.

Der Gleichung (2) ist zu entnehmen, dass auch für nicht vorhergesagte Berechnungen Prediction-Kosten γ_P entstehen. Deswegen macht gemäß Voraussetzung 3 der Einsatz von Precalculation nur dann Sinn, wenn für die Summe λ der durch Prediction reduzierten Kosten weiter gilt:

$$\lambda > \sum_{i=B+1}^A \gamma_P(d_i) \quad (3a)$$

Diese Zusatzbedingung (3a) wurde bisher nicht behandelt. Der Grund hierfür liegt in der mangelnden Messbarkeit. λ lässt sich nur sinnvoll *rückblickend* angeben, wenn die Kostenreduktionen bereits vorliegen. Die Autoren schlagen hier vor, die gewählten Mechanismen um eine Kontrolle zur Erfüllung von (3a) zu erweitern und gegebenenfalls die Prediction abzuschalten⁵.

Im Abschnitt (4.2) findet sich ein Vorschlag zur Ermittlung der Anforderungswahrscheinlichkeiten für einzelne Datenmengen.

3.2 Prefetching

Auf der Clientseite gewinnen die Kosten zur Herstellung und Aufrechterhaltung einer Verbindung zum Server stark an Bedeutung, während die Berechnungskosten auf der Clientseite – je nach Anwendung – zumeist an Gewicht verlieren. Allgemein ergibt sich die Kostenfunktion für eine Session s zu:

$$\gamma(s) = \sum_{i=1}^A (\gamma_C(d_i) + (\gamma_V(d_i) + \gamma_O(d_i)) + (\gamma_R(d_i) + \gamma_Z(d_i))) \cdot p_i \quad (4)$$

wobei A Anforderungen⁶ von Datenmengen d_1, d_2, \dots, d_A mit der jeweiligen Wahrscheinlichkeit p_i innerhalb s erfolgen. Die Gesamtkosten setzen sich hierbei zum einen aus den bereits unter (2.1) angegebenen Serverkosten γ_R und γ_Z zusammen. Weiterhin entstehen Systemkosten auf der Clientseite, die pro Anforderung mit γ_C angegeben sind. Schließlich lassen sich die Übermittlungskosten in einen von der Datenmenge abhängigen Teil γ_V und einen *Routing-Overhead*-Anteil γ_O einteilen⁷. In der Literatur werden die Systemkosten zumeist nicht weiter

⁴ Dies läßt sich durch die Precalculation selbst begründen: wenn aufgrund der Systemlastverteilung keine Ersparnis zu erwarten ist, findet auch keine Precalculation statt!

⁵ Diese Kontrolle darf jedoch nicht so oft oder zu so ungünstigen Zeitpunkten erfolgen, daß sie selbst wieder γ_P erhöht.

⁶ Selbstverständlich sind lokal „ge-cache-te“ Datenmengen ausgenommen.

⁷ Als Protokoll wird hier das Internet-Protocol (IP) vorausgesetzt.

differenziert. In diesem Sinne ist der vorliegende Ansatz allgemeiner. Zum grundlegenden Vorgehen sei auch auf [JK98] verwiesen.

Falls eine Anforderungsvorhersage mit Prefetching auf der Clientseite erfolgt, werden in Zeitintervallen, in denen genügend freie Systemressourcen und Übertragungsbandbreiten bereitstehen, ohne explizite Anweisung des Benutzers Anforderungen an den Server initiiert, um in naher Zukunft wahrscheinlich benötigte Datenmengen sofort über das Netz zu laden.

Die Kostenfunktion ergibt sich dann zu:

$$\gamma(s) = \sum_{i=1}^B (\gamma_R(d_i) + \gamma_Z(d_i)) + \sum_{i=B+1}^A (\gamma_C(d_i) + \gamma_U(d_i) + \gamma_O(d_i)) \cdot p_i \quad (5)$$

wobei wiederum die Datenmengen d_1, d_2, \dots, d_B vorausgesagt werden konnten. Für korrekt vorhergesagte Dokumente ergibt sich somit ein Verschwinden von Verbindungs- und Client-Berechnungskosten.

Der Wahrscheinlichkeits-Schwellwert für eine positive Kostenbilanz ermittelt sich wiederum leicht aus (4) und (5) und ergibt:

$$p_i > \frac{\gamma_R(d_i) + \gamma_Z(d_i)}{\gamma_R(d_i) + \gamma_Z(d_i) + \gamma_U(d_i) + \gamma_O(d_i)} = \frac{1}{1 + \frac{\gamma_U(d_i) + \gamma_O(d_i)}{\gamma_R(d_i) + \gamma_Z(d_i)}} \quad (6)$$

Die Anforderungswahrscheinlichkeit muss also um so höher liegen, je teurer Systemressourcen auf der Serverseite im Vergleich zu Kosten im Übertragungsweg sind.

3.3 Huckepack-Übertragung

Zwischen den in (3.1) und (3.2) aufgezeigten Möglichkeiten zur Anforderungsvorhersage auf Server- bzw. Clientseite existiert noch eine dritte. Im Moment der Datenbereitstellung kann der Server *zusätzlich* zu den angeforderten Daten eine in naher Zukunft zu erwartende Datenmenge „Huckepack“ an den Client senden. Allerdings zeigt dieses Verfahren neben schlechteren Performancesteigerungen noch einige gravierende Netzprobleme, die in Abschnitt (5.2) weiter besprochen werden. Vorschläge, dieses Verfahren in einer abgeschwächten Form zu verwenden, wurden bereits auch gemacht [CKR98].

Die Kostenberechnungen ergeben sich schnell aus (4), (5) und (6), wobei jedoch neben den Serverkosten γ_R und γ_Z auch ein Teil der Übertragungskosten, nämlich γ_U , für jede Datenmenge anzurechnen ist.

3.4 Multiples Preprocessing

Im Idealfall sollten sich die Vorteile von (3.1) und (3.2) in einer *kombinierten Lösung* zusammenbringen lassen. Dies ist tatsächlich auf Serverseite möglich, jedoch ist hierzu eine persistente Verbindung zwischen Client und Server vonnöten. Dann kann die Serverseite in der „freien Zeit“ des Clients selbständig und unaufgefordert Datenmengen senden, die zur weiteren Verarbeitung auf der Clientseite im Zeitpunkt der Anforderung unmittelbar zur Verfügung stehen.

Allerdings hat diese Lösung den Einsatz von Proxy-Cache-Servern als Mittler zwischen Client und Server zu berücksichtigen. Ansätze hierzu finden sich in [CKR98] und [CI98]. Insbesondere sind in der Literatur persistente Verbindungen zwischen Server und Proxy besprochen worden [CDF98]. Vor- und Nachteile des Einsatzes von Prediction auf Server- bzw. Clientseite werden in (5.3) diskutiert.

4. Ermittlung der Anforderungswahrscheinlichkeiten

4.1 Das mathematische Modell

Zur Ermittlung von Anforderungswahrscheinlichkeiten für HTML-Seiten finden sich in der Literatur inzwischen einige Ansätze, darunter *Path-Profiling* [SKS98]. Die baumartige Verbindung zwischen derartigen Dokumenten über Hyperlinks gestattet eine Modellierung der Vorhersage als stochastische Prozesse in Form von *Markov-Ketten*. Die Wahrscheinlichkeit zur Anforderung eines Dokumentes, das nicht als Link im aktuellen vorhanden ist, wird zumeist mit 0 angesetzt. [BA96]. Neben der Modellierung des stochastischen Prozesses als diskrete Markov-Kette, wird auch ein kontinuierlicher Ansatz angeführt [KW98]. Für den allgemeinen Fall der Datenanforderung eignet sich dieses mathematische Modell jedoch nicht, falls keine Verbindungen zwischen den Datenmengen bestehen, wie dies für HTML-Dokumente mit entsprechenden Hyperlinks der Fall ist. Wir werden deshalb einen Ansatz mit bedingten Wahrscheinlichkeiten wählen, der grundsätzlich auch auf das Vorhersagen von HTML-Anforderungen anwendbar ist.

4.2 Der Grundalgorithmus

Wir haben bereits die möglichen Datenmengen, die ein Server im Internet potentiell für eine Session s bereitstellt, mit $D = d_1, d_2, \dots, d_A$ bezeichnet. Es ist zu beachten, dass für den allgemeinen Fall der Wert A sehr groß werden kann. Insbesondere stellt D möglicherweise den gesamten auslieferbaren Datenbestand des Servers da. Für komplexe Aufbereitungen und Berechnungen auf der Serverseite lassen sich die wesentlichsten Funktionen ebenfalls mittels D abbilden.

So könnte ein System, das Informationen über Börsenbestände bereitstellt, durchaus auch Indikatoren und Trends für die wichtigsten Papiere unter den d_i aufzählen.

Die Anforderungen einer Benutzersitzung s lassen sich dann - ohne Berücksichtigung der Reihenfolge oder wiederholter Aufrufe - als binärer \mathbf{A} -Vektor von Datenmengenindizes darstellen:

$$s = (s_i) \text{ mit } s_i = \begin{cases} 1: \text{Datenmenge } \mathbf{d}_i \text{ wurde vom Client angefordert} \\ 0: \text{Datenmenge } \mathbf{d}_i \text{ wurde nicht angefordert} \end{cases}$$

Das „Gedächtnis“ für die Vorhersage besteht in einer symmetrischen⁸, quadratischen Matrix φ , deren Elemente die Häufigkeiten der Aufrufe der Funktionen beinhalten. Die Matrix φ bezeichnen wir auch als *Erinnerungsmatrix*.

$$\varphi = (\varphi_{ij}) \text{ mit } \mathbf{1} \leq i, j \leq \mathbf{A}. \quad \varphi \text{ wird als 0-Matrix initialisiert: } \varphi_0 = (\varphi_{ij}) = 0 \quad \forall 1 \leq i, j \leq \mathbf{A} \quad (7)$$

Innerhalb jeder Sitzung s wird φ zu φ' angepasst, so dass die einzelnen Elemente die Häufigkeit des Abrufs einer Datenmenge in Abhängigkeit des Auftretens der jeweilig anderen widerspiegeln.

$$\varphi' = (\varphi'_{ij}) = \begin{cases} \varphi_{ij} + 1: \text{falls } s_i = 1 \wedge s_j = 1 \\ \varphi_{ij}: \text{sonst} \end{cases} \quad (8)$$

Die bedingte Wahrscheinlichkeit p_i dafür, dass \mathbf{d}_i angefordert wird, nachdem \mathbf{d}_j in derselben Sitzung bereits angefordert wurde, lässt sich dann folgendermaßen angeben:

$$p_i(\mathbf{d}_i | \mathbf{d}_j) = \frac{\varphi_{ij}}{\varphi_{jj}} \quad (9)$$

⁸ Die Matrix bleibt symmetrisch, solange man die Reihenfolge der Anforderungen außer Acht lässt. Wird der Reihenfolge eine stärkere Bedeutung geschenkt, verliert die Matrix ihre Symmetrieeigenschaft.

Diese Formel drückt aus, dass sich die Wahrscheinlichkeit dafür, dass eine Datenmenge am Server angefordert wurde, aus den bisherigen Häufigkeiten schließen lässt (Voraussetzung 1). Wenn eine Funktion beispielsweise stets zusammen mit einer anderen aufgerufen wurde, wird dieser Quotient zu 1.

Zur Verdeutlichung dieses Prinzips sei ein einfaches Beispiel angeführt:

Angenommen $\mathbf{D} = \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4$ mit $A = 4$. Dann führen die beiden Sitzungen $\mathbf{s}_1 = (\mathbf{1} \ \mathbf{0} \ \mathbf{1} \ \mathbf{0})$ und $\mathbf{s}_2 = (\mathbf{1} \ \mathbf{1} \ \mathbf{0} \ \mathbf{1})$ zu folgender Erinnerungsmatrix:

$$\varphi_2 = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

So ergibt sich beispielsweise die bedingte Wahrscheinlichkeit dafür, dass \mathbf{d}_2 aufgerufen werden wird, nachdem \mathbf{d}_1 bereits aufgerufen worden ist, nunmehr zu:

$$P(\mathbf{d}_2 | \mathbf{d}_1) = \frac{\varphi_{12}}{\varphi_{11}} = \frac{1}{2}$$

Das „Lernen“ der Matrix φ beginnt in der ersten Sitzung und hört niemals auf. Jede Anforderung führt zum Anwachsen des entsprechenden Eintrags in der Matrix. Werden jedoch Datenmengen vorausberechnet, so führen diese Anforderungen nicht zu einem Eintrag in φ , so dass die Erinnerungsmatrix dazu tendiert, Zusammenhänge zwischen den Datenmengen „zu vergessen“. Selbstverständlich muss dieser Grundalgorithmus für den praktischen Einsatz modifiziert werden, um beispielsweise den Speicherbedarf zu reduzieren.⁹

Um Voraussetzung 1 gebührend Rechnung zu tragen, schlagen wir vor, ähnlich der Zusatzforderung (3a) aus Abschnitt (3.1), folgende Gültigkeit (10) zu prüfen:

$$\sum_{i=j} \varphi_{ij} \ll \sum_{i<j} \varphi_{ij} \wedge \sum_{i=j} \varphi_{ij} \cdot \frac{A}{A+1} \approx \sum_{i=1}^A \max_{j \neq i}(\varphi_{ij}) \quad (10)$$

Die Summe der Hauptachsenwerte, welche die Häufigkeit aller Anforderungen angeben, sollte erheblich kleiner sein als die Summe hiermit zusammenhängender Datenanforderungen. Außerdem sollte die *Kontinuität* des Benutzerverhaltens durch eine enge Beziehung zwischen miteinander zusammenhängenden Datenmengenrequests belegbar sein.¹⁰ Die Anwendung der Forderung (10) kann als Methode zur Klassifikation des Benutzerverhaltens verwendet werden.

5. Kritische Aspekte der Anforderungsvorhersage

5.1 Wiedererkennung ähnlicher Anforderungen

Die Grundidee jeder Anforderungsvorhersage beruht darauf, dass künftiges Benutzerverhalten aus den vorhandenen Zugriffen abgeleitet werden kann (Voraussetzung 1). Sinn macht dieses Verfahren

⁹ In der Praxis wird man die Erinnerungsmatrix so speichern, daß aufgrund der Symmetrie gleiche Werte nur einfach gehalten werden. Wenn die Anzahl der Datenelemente jedoch zu groß wird, zum Beispiel im Online-Buchhandel mit mehreren Millionen Datensätzen, müssen Datenstrukturen wie gerichtete Graphen die Matrix ersetzen.

¹⁰ Die Werte entstammen unseren Erfahrungen und können als Mindestvoraussetzungen betrachtet werden.

jedoch nur, wenn de facto ähnliche Aufrufe auch als solche erkannt werden. Dabei können beide Extreme auftreten: zum einen kann der Abruf derselben Datenmenge nicht erkannt werden, weil die Aufrufparameter unterschiedlich sind. So können nutzer- oder sitzungsidifizierende Cookies etwa die Zwischenspeicherfunktion von Proxy-Cache-Servern sehr erschweren [SKS98]. Dieses Argument gilt analog auch für funktionale, parametrisierte Aufrufe. Man kann dieses Problem umgehen, indem die Prediction-Algorithmen von allen veränderlichen, aber inhaltlich irrelevanten Parametern absehen.¹¹ Zum anderen können scheinbar gleiche Aufrufe zu völlig unterschiedlichen Ergebnissen führen. Die Datenmenge mit Informationen über die aktuelle Wetterlage, über Börsenkurse oder Abfahrtszeiten kann trotz gleicher Aufrufparameter jeden Tag, ja jede Minute eine andere sein. Auch statische HTML-Dokumente werden hin und wieder verändert. Ein Prediction-Algorithmus muss die Aktualität der im voraus ermittelten Daten stets gewährleisten können. Hier können die Methoden der Proxy-Server analog eingesetzt werden [CDF98].

5.2 Netzwerkparadoxien

Obgleich scheinbar offensichtlich ist, dass die Sendung von wenigen größeren Paketen aufgrund geringeren Routing- und TCP-Overheads effizienter ist als jene von vielen kleineren Paketen mit derselben Nutzlast in der Summe, trifft diese Aussage dennoch nicht zu [RS94]. De facto können größere Pakete zu Schwierigkeiten mit Router-Puffern führen, die sich noch stärker bemerkbar machen als der bessere Anteil an Nutzlast [CB98]. Für die Prefetching-Algorithmen bedeutet dies, dass eine unangeforderte Datensendung möglichst vom aktuellen Zeitpunkt bis zum vermuteten Moment des Gebrauchs „gleichmäßig verteilt“ über das Netz gehen sollte [BA95]. Für uns ist dieses Vorgehen Teil einer Strategie, die wir mit *defensiv* bezeichnen.

Zur Voraussage der zu erwartenden Zeit bis zur Anforderung kann der Grundalgorithmus aus (4.2) dienen. Die entstehende Erinnerungsmatrix bleibt dann aber nicht mehr symmetrisch. In diesem Papier beschränken wir uns jedoch auf eine symmetrische Matrix. Es sollte hier noch Erwähnung finden, dass das Vorhandensein von *freien Zeiten*, wie sie üblicherweise zwischen zwei Benutzerrequests angenommen werden, nicht mehr gelten, wenn Datensendungen gleichmäßig auf diese Intervalle verteilt werden.

5.3 Serverseite versus Clientseite

Wir haben in den Kapiteln 2 und 3 stets offen gelassen, welche Form der Vorhersage vorzuziehen ist, jene, die auf der Serverseite stattfindet, oder jene auf der Clientseite. Vielmehr wurde eine Kombination aus beiden propagiert (3.4). An dieser Stelle sollen einige weitere Argumentationshilfen geliefert werden.

Der Server kennt nicht nur das Anforderungsverhalten eines speziellen Nutzers, sondern kann, insbesondere im Anfangsstadium des Lernprozesses, auf das durchschnittliche Verhalten der gesamten Nutzergemeinde zurückgreifen. Aus wirtschaftlichen Gründen ist es ebenso angemessen, hocheffiziente und teure Datenbanken, die das „Wissen“ des Systems repräsentieren, auf der Serverseite zu fokussieren, weil hier der Nutzen für eine größere Benutzergemeinde gegeben ist. Andererseits lässt die Lastverteilung von Mehrbenutzersystemen weniger Spielraum für Precalculation, als dies bei Einzelbenutzersystemen der Fall ist. Die durchschnittliche Konsistenz des vorhersehbaren Verhaltens ist aus wahrscheinlichkeitstheoretischer Sicht (*Gesetz der großen Zahlen*) auf Serverseite höher.

Dagegen kann die Clientseite die Erfahrungen aus der Kommunikation zwar nicht mit vielen Benutzern, dafür aber mit anderen Servern berücksichtigen. Allerdings sind hier die Wiedererkennungsprobleme (5.2) bei komplexen Anfragen enorm schwer zu bewältigen.

¹¹ Dabei kann auch ein Cookie inhaltlich relevant werden, wenn etwa „die nächsten 10 Datensätze“ einer Datenankabfrage angefordert werden und (nur) im Cookie kodiert ist, welche Datensatznummern nun tatsächlich gewünscht sind.

Gegen das Paradigma, dass ein Client eine Verbindung zu einem Server herstellen kann - nicht aber umgekehrt – ist anzuführen, dass innerhalb von Intranets oder mittels persistenter Verbindungen diese Unterscheidung verwischt wird [CDF98].

Es ergibt sich somit, dass das individuelle Benutzerverhalten selbst die entscheidende Rolle spielt. Je stringenter, konsequenter und stabiler das Anforderungsverhalten des einzelnen Anwenders ist, um so effizienter lässt sich ein Prediction-Szenario auf der Clientseite anbringen. Andererseits kann die Serverseite auch ein eher zufälliges Anforderungsverhalten des Einzelnen noch sinnvoll verarbeiten, wenn sich die Zugriffe der Nutzergemeinde insgesamt gemäß den in Kapitel 2 dargelegten Voraussetzungen 1 bis 3 quantifizieren lassen.

Ein aktueller Forschungsschwerpunkt im Bereich der Prediction liegt gerade in einer derartigen Benutzer-Verhaltensklassifikation.

5.4 Verwendung von Zusatzinformationen

Das voraussagbare Benutzerverhalten, wie in der ersten Voraussetzung proklamiert, hat zumeist konkrete Ursachen. Aus psychologischen Erwägungen ist leicht nachvollziehbar, dass (menschliche) Anwender dazu neigen, die Flut der Reize nicht ständig anwachsen zu lassen, sondern auch Wiederholungen anzustreben. Wenn eine Auswahl zwischen mehreren gleichwertigen Datenquellen über dieselben Grundinformationen (Wetter, Börsendaten etc.) besteht, werden die meisten Menschen fast immer dieselben Adressen benutzen, wenn sie mit der Darbietung der Information zufrieden sind. Auch Börseninteressierte fordern zumeist Informationen über Wertpapiere aus dem persönlichen Portefeuille an. Hier stellt sich die Frage, ob solche *Zusatzinformationen* nicht dazu verwendet werden sollten, eine Anforderungsvorhersage zu erstellen. Dies trifft sicherlich zu, doch sind juristische und sicherheitstechnische Fragen im Zusammenhang mit derartigen Lösungsmodellen sorgfältig zu beachten.

Vielversprechend scheint auch der umgekehrte Weg: inhaltliche Zusammenhänge der Datenmenge lassen sich über Mechanismen der Prediction sehr einfach gewinnen.

6. Zusammenfassung und Ausblick

Die Vorhersage von Benutzeranforderungen im Internet scheint ein vielversprechender Weg zu sein, um das Antwortzeitverhalten aus Sicht des Anwenders zu verbessern. Nach einer grundlegenden Darstellung der elementaren Möglichkeiten, um Vorhersagen über künftige Anforderungen zu treffen, wurden die Risiken und Gefahren, die mit dem gewagten Schritt über die Grenzen der Zeit einhergehen, kritisch beleuchtet. Es zeigt sich, dass unter Verwendung von defensiven Strategien die Vorteile der Vorhersagetechniken die Nachteile überwiegen und so eine Verbesserung der Gesamtperformance für die Internet-Gemeinde zu erzielen ist.

Die aktuellen Forschungen im Bereich der Prediction zielen neben der in Kapitel 3 angesprochenen Verfeinerung des Kostenmodells, das auch eine dynamische Ermittlung von optimierbaren Kosten vorsehen sollte, ebenso auf die Klassifikation der Anforderungen selbst. In Abhängigkeit von den individuellen Bedürfnissen des menschlichen Benutzers und dem zur Verfügung stehenden Informationsangebot sollte die Serverseite eine implizite oder explizite Klassifikation vornehmen, um zufällige, nicht vorhersehbare Anforderungen in jedem Einzelfall von solchen zu unterscheiden, die aufgrund der drei genannten Voraussetzungen durchaus vorhersehbar sind.

Die Theorie der Vorhersage für Benutzerrequests wird sich zunehmend von den verwandten Gebieten lösen müssen, um die Herausforderungen einer immer weiter wachsenden Datenflut der Informationsgesellschaft meistern zu können.

Literaturangaben

- [BA96] Azer Bestavros, Speculative Data Dissemination and Service, Proceedings of ICDE96, March 1996
- [BA95] Azer Bestavros, Using Speculation to Reduce Server Load and Service Time on the WWW, Proceedings of CIKM95, November 1995
- [BC98] Paul Barford, Mark Crovella, Generating Representative Web Workloads for Network and Server Performance Evaluation, SIGMETRICS 1998
- [BG98] Wolfgang Benn, Ingo Gringer; Zugriff auf Datenbanken über das World Wide Web; Informatik-Spektrum 21, S. 1-8
- [CB98] Mark Crovella, Paul Barford, The Network Effects of Prefetching, IEEE Infocom 1998,
- [CDF98] Ramón Cáceres, Fred Douglis, Anja Feldmann, Gideon Glass, Michael Rabinovich, Web Proxy Caching: The devil is in the details, Workshop on Internet Server Performance, June 1998, Madison, WI
- [CI98] Pei Cao, Sandy Irani, Cost-Aware Proxy Caching Algorithms, 1998
- [CKR98] Edith Cohen, Balanchander Krishnamurthy, Jennifer Rexford, Improving End-to-End Performance of the Web Using Server Volumes and Proxy Filters, 1998
- [FAR98] Jim Farley, JAVA - Distributed Computing, O'Reilly & Associates, 1998
- [HTML] HTML - HyperText Markup Language, <http://www.w3.org/MarkUp/>
- [Java] Java, <http://www.javasoft.com/>
- [JK98] Zhimei Jiang, Leonard Kleinrock, An Adaptive Network Prefetch Scheme, IEEE J Sel Areas Commun, 1998
- [KW97] Achim Kraiss, Gerhard Weikum, Vertical Data Migration in Large Near-Line Document Archives Based on Markov-Chain Predictions, Proceedings of the 23rd VLDB Conference, 1997
- [KW98] Achim Kraiss, Gerhard Weikum, Integrated document caching and prefetching in storage hierarchies based on Markov-chain predictions, The VLDB Journal, Springer-Verlag, 1998
- [KLM97] Thomas Kroege, Darrell Long, Jeffrey Mogul, Exploring the bounds of web latency reduction from caching and prefetching, 22. USENIX, December 1997
- [RS94] W. Richard Stevens, TCP/IP Illustrates, Volume 1: The Protocol, Addison-Wesley Pub. Co. 1994
- [SKS98] Schechter, Krishnan, Smith, Using path profiles to predict HTTP requests, Proceedings of the World Wide Web Conference, 1998