Personal Picture Finder: Ein Internet-Agent zur wissensbasierten Suche nach Personenphotos

Christoph Endres, Markus Meyer, Wolfgang Wahlster
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI)
Stuhlsatzenhausweg 3
66123 Saarbrücken
{endres,meyer,wahlster}@dfki.de

Zusammenfassung

Es wird die Funktionsweise eines Netbot vorgestellt, der nach Eingabe eines Personennamen im World Wide Web (WWW) nach Photos dieser Person sucht und die gefundenen Bilder zusammen mit einem Hinweis auf die Fundstelle ausgibt. Dazu werden zunächst durch eine parallele Metasuche relevante Webseiten gefunden, aus deren HTML-Code dann alle Bilddateien extrahiert werden. Der Netbot nutzt dann eine Kombination verschiedener Heuristiken aus seiner Wissensbasis, um die Bilddateien mit Personenphotos aufgrund effizient zu berechnender Bildmerkmale zu extrahieren und alle anderen Bilddateien wegzufiltern. Durch verschiedene Lerntechniken, die als Eingabe die Bewertungen von Suchergebnissen durch eine große Zahl von Benutzern haben, adaptiert der Netbot seine Bildfilter und erhöht damit die Precision-Werte beim Retrieval. Durch die Verwendung der Parallel Pull-Technologie und eine Kombination von Java Scripts, Applets und Servlets wird eine hohe Performanz des Systems erreicht.

Inhaltsverzeichnis

1 Motivation

Persönliche Netzassistenten sind als Softwaresysteme realisiert, die sich im Internet bestens auskennen, Informationswünsche des Anwenders richtig interpretieren, in gezielte Suchanfragen umsetzen und die relevanten Rechercheergebnisse systematisch zusammenfassen, filtern oder intelligent verknüpfen. Man nennt solche Netzassistenten auch Netbots (für Inter**net** Ro**bots**). Denn sie übernehmen geistige Arbeit des Benutzers bei der Suche im WWW, indem sie diverse Internet-Dienste im Auftrag des Anwenders ansteuern, bedienen und nutzen. Netbots arbeiten nach der Devise "Weniger ist mehr". Sie suchen gezielt die Internet-Information heraus, die für den jeweiligen Benutzer wirklich relevant ist. In diesem Papier wird die Funktionsweise eines neuartigen Netbots mit dem Namen Personal Picture Finder vorgestellt.
Der Personal Picture Finder wurde im Rahmen des Projektes PAN[1] am DFKI entwickelt. Ziel des Projektes PAN ist die Entwicklung planbasierter Informationsassistenten für Benutzer des Internets. Hauptaufgaben dabei sind die Generierung und Ausführung von Plänen in dynamischen Umgebungen, die Verwaltung des zugrundeliegenden Domänenmodells und die Realisierung kooperativen und adaptiven Verhaltens des Assistenten. Im Projekt PAN wird derzeit ein Werkzeug entwickelt, mit dessen Hilfe Informationsassistenten nach persönlichen Bedürfnissen konfiguriert und auf die Extraktion relevanter Information trainiert werden können (*InfoBean-Konzept*; [BD99]). Ein Schwerpunkt liegt dabei auf der einfachen Anpaßbarkeit dieser Assistenten an sich verändernde Informationsstrukturen und neue Domänen. Der Personal Picture Finder ist eine beispielhafte Anwendung der TrIAs-Technologie[2] [BD98], die im Projekt PAN konzipiert und entwickelt wurde.
Als persönlicher Internet-Assistent kann der Personal Picture Finder beispielsweise Bildjournalisten unterstützen, die im Internet auf der Suche nach Bildern von Persönlichkeiten sind. Er durchforstet das gesamte WWW nach Photos der angefragten Person und stellt diese in geeigneter Form dar. Aber auch für einen Illustrator, der das Photo einer bekannten Persönlichkeit in eine zu erstellende Graphik einbinden möchte, ist der Personal Picture Finder ein zeitsparendes Werkzeug. Interessant ist auch die Anwendungsmöglichkeit, den Personal Picture Finder zu verwenden, um mögliche Verletzungen des Copyrights an Bildern durch Betreiber von Informationsdiensten festzustellen. Da der Netbot sämtliche Instanzen eines Bildes mit Quellverweisen aufspürt, läßt sich leicht feststellen, ob ein vom Benutzer mit Copyright eingestelltes Bild einer Person an anderer Stelle unerlaubt wiederverwendet wird. Interessant für den privaten Einsatz ist auch die Vorbereitung von Verabredungen mit Personen, die man bislang noch nie getroffen hat, etwa zur visuellen Identifikation z.B. bei der Abholung eines Gastes am Flughafen. Auch für Fanclubs und einzelne Fans von Medienstars dürfte der Personal Picture Finder sehr hilfreich sein, da man sicher sein kann, mit ihm weltweit alle relevanten Photos aufzuspüren. Schließlich sei auch auf das Anwendungspotential für Fahndungszwecke verwiesen. Selbst wenn die meisten Täter kaum eine Homepage mit Photo anlegen werden, kann der Personal Picture Finder wertvolle Dienste leisten, da er auch Bilder aus dem WWW extrahiert, auf denen die gesuchte Person bei anderen Anlässen z.B. auch in Gruppen abgebildet ist, wenn eine

[1] PAN ist das Akronym für **P**lanning **A**ssistant for the **N**et.
[2] TrIAs ist das Akronym für **Tr**ainable **I**nformation **As**sistants.

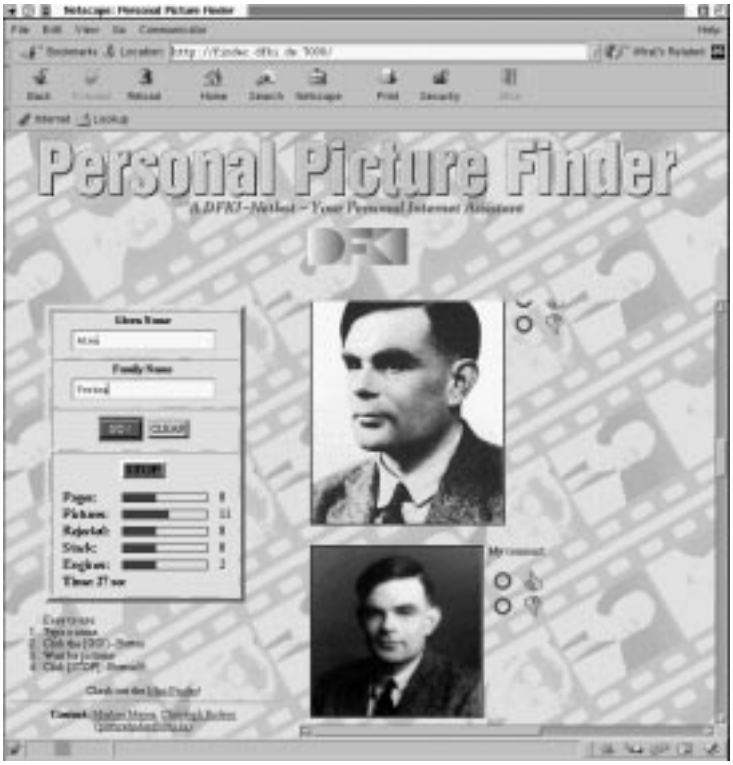

Abbildung 1: Ein Suchergebnis des Personal Picture Finder

entsprechende textuelle Bildannotation vorliegt.
Da bei diesem speziellen Informationsdienst eine Vielzahl von Suchanfragen parallel abgearbeitet werden muß, wird die *"Parallel-Pull"-Technologie* zur effizienten Metasuche verwendet. Ein weiterer Schwerpunkt bei der Konzeption dieses Netbots war die Konfiguration verschiedener Bildfilter, die benötigt werden, um beispielsweise Icons und Werbebanner auszufiltern.
Der Personal Picture Finder benutzt als Netbot selbst wieder andere Netbots, wie den Metacrawler (www.metacrawler.com) und Ahoy! (ahoy.cs.washington.edu:6060), um möglichst hohe Recall- und Precision-Werte beim Retrieval zu erreichen.
Im Gegensatz zu webbasierten Bildsuchdiensten, die auf Indices beruhen, arbeitet der Personal Picture Finder dynamisch und hat somit einen erheblich höheren Recall. Zudem nutzt er die beiden bekannten Bildarchive AltaVista Photo Finder (image.altavista.com) und den Lycos Bildkatalog (www.de.lycos.de) als weitere Ressourcen in seiner Metasuche.

2 Arbeitsweise des Personal Picture Finder

Der nun folgende Abschnitt erklärt anhand eines Beispiels die Idee und die Arbeitsweise des Personal Picture Finder.
Man stelle sich dazu die folgende Situation vor. Ein Bildjournalist ist auf der Suche nach Photos von Alan Turing. Mit den herkömmlichen Suchverfahren im Internet fragt dieser

Abbildung 2: Anfrageformular und Anzeigen

Journalist nun zunächst eine Suchmaschine an. Als Ergebnis bekommt er eine Liste mit Webseiten zurückgeliefert, die mit dem angegebenen Suchbegriff (hier: Alan Turing) in Verbindung gebracht werden. Jetzt beginnt für den Journalisten die eigentliche Arbeit. Die zurückgelieferten Webseiten müssen zunächst alle einzeln in den Browser geladen und nach geeignetem Bildmaterial durchforstet werden. Nach einigen Minuten Arbeit verbunden mit erheblichem Aufwand findet der Journalist unter Umständen auch eine Webseite von Alan Turing. Auf dieser Seite befindet sich zwar ein Bild von ihm, aber auch eine große Menge an Information, die nicht benötigt wird. So werden beispielsweise Icons oder Banner mitangezeigt, die das Herunterladen der Seite aus dem Web erheblich verlangsamen. Außerdem hat der Journalist jetzt erst ein einziges Bild gefunden, er möchte aber lieber mehrere Photos zur Auswahl haben.
Man sieht, daß das beschriebene Vorgehen sehr zeitintensiv und mühsam ist. Aus dieser Situation heraus ist die Idee des Personal Picture Finder entstanden. Man hätte gern einen Internet-Agenten, dem man nur den Vor- und den Nachnamen der gesuchten Person angibt, und der dann autonom im Internet möglichst schnell nach Photos dieser Person sucht und diese dem Benutzer zur Auswahl gibt. Genau diese Aufgabe erfüllt der Personal Picture Finder. Wie in Abbildung 1 dargestellt, gibt der Benutzer den Namen der gesuchten Person an und erhält nach kurzer Zeit mehrere Photos zur Auswahl. Zusätzlich hat er die Möglichkeit, durch Mausklick auf das entsprechende Bild zu der Webseite zu gelangen, von der es stammt. Der Benutzer kann außerdem sein Feedback zu den einzelnen Ergebnissen abgeben.
Der Benutzer kann den Verlauf der Suche anhand der Anzeigen im Applet nachvollziehen (siehe Abbildung 2). Eine Uhr zählt die Sekunden, die seit Beginn der Anfrage verstrichen sind. Die weiteren Anzeigen sind:

- *Pages*: Bisherige Anzahl der Adressen von Webseiten, die zu dem gesuchten Namen

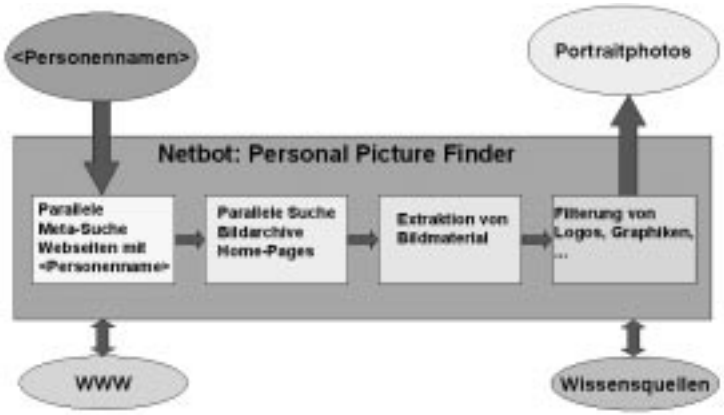

Abbildung 3: Arbeitsweise des Personal Picture Finder

gefunden wurden.

- *Pictures*: Anzahl der bisher angezeigten Bilder.
- *Rejected*: Anzahl der bisher aussortierten Bilder.
- *Stack*: Anzahl der Webseiten, die bereits angefragt, aber bisher noch nicht ausgewertet wurden.
- *Engines*: Anzahl der Suchmaschinen, die angefragt wurden und deren Ergebnis bisher noch aussteht.

Es folgt nun eine kurze Beschreibung der Arbeitsweise des Personal Picture Finder (siehe Abbildung 3): Der Benutzer gibt zunächst den Namen der gesuchten Person ein. Dieser Name wird dazu verwendet, mehrere zeitgleich laufende Anfragen an verschiedene Suchmaschinen im WWW zu starten. Das Ziel dabei ist, Webseiten zu finden, auf denen der angefragte Name vorkommt. Die von den Suchmaschinen zurückgelieferten Seiten werden anschließend parallel geladen. Außerdem werden verschiedene Bildarchive abgefragt. Aus den geladenen Webseiten wird dann das Bildmaterial extrahiert. Dieses Bildmaterial muß dann mehrere Filter durchlaufen. Dabei werden beispielsweise Icons, Graphiken und Werbebanner herausgefiltert. Das übrigbleibende Bildmaterial wird abschließend als Ergebnis (Portraitphotos) präsentiert.

3 Systemarchitektur

Die im vorigen Abschnitt beschriebene Arbeitsweise wird nun detaillierter betrachtet, und einige weitere Eigenschaften des Systems werden vorgestellt. Abbildung 4 zeigt die schematische Darstellung des Systems.
Der Personal Picture Finder besteht aus folgenden Modulen:

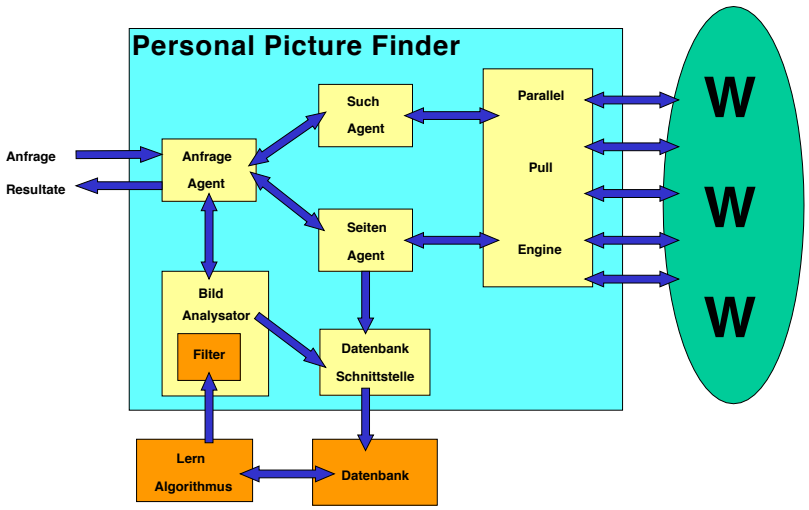

Abbildung 4: Architektur des Personal Picture Finder

- Anfrageagent
- Suchagent
- Seitenagent
- Parallel Pull Engine
- Bildanalysator
- Datenbank-Schnittstelle

Der *Anfrageagent* bearbeitet die Anfrage des Benutzers, indem er die verschiedenen Aufgaben delegiert und Teilergebnisse sammelt oder weiterleitet. Beendet der Benutzer die Anfrage, so sorgt der Anfrageagent dafür, daß alle noch laufenden Threads ordnungsgemäß terminieren. Damit wird die Stabilität des Systems auch bei vielen Zugriffen gewährleistet.
Der Anfrageagent delegiert Teilaufgaben an Suchagent, Seitenagent und Bildanalysator.
Der *Suchagent* formuliert aus der Benutzeranfrage die korrekte Anfrage für die angesprochenen Suchmaschinen und leitet sie an die *Parallel Pull Engine* weiter.
Die Parallel Pull Engine[3] ist das Kernstück des Personal Picture Finder. Sie verwaltet und bearbeitet mehrere Anfragen gleichzeitig. Dabei wird ständig darauf geachtet, daß die Anzahl der nebenläufigen Threads nicht zu hoch wird, da dadurch die Stabilität des gesamten Vorganges gefährdet würde.
Sobald HTML-Seiten zurückgeliefert werden, extrahiert der Anfrageagent die essentielle Information, also entweder Adressen von Homepages, die mit dem angegebenen Namen in Verbindung gebracht werden oder aber - bei Anfragen an Bilddatenbanken - Adressen

[3]Die Funktionalität einer Parallel Pull Engine wird im nächsten Abschnitt ausführlich am Beispiel des MultiHttpServers erläutert.

von Bildern im WWW.
Der *Seitenagent* fragt über die Parallel Pull Engine Webseiten an, deren Adressen er vom Anfrageagent bekommt, extrahiert die Adressen von Bildern und bewertet die Relevanz der Seite für die Benutzeranfrage. Die Bildadressen werden dem Anfrageagent übergeben, der sie an den Bildanalysator weiterleitet.
Im *Bildanalysator* wird sowohl anhand der Header-Information der gefundenen Bilder, als auch der Herkunft der Bilder entschieden, was dem Benutzer gezeigt wird und was nicht. Berücksichtigt wird dabei Information aus der Analyse der Seite, von der das Bild stammt, der Name des Bildes und natürlich auch Information über das Bild selbst, wie beispielsweise Kompressionsfaktor, Größe, Farbtiefe und Format.
Bei der Implementierung wurde viel Wert auf Geschwindigkeit gelegt. Deshalb kommen aufwendige Analysen und Bilderkennung nicht in Frage. Die bisher angewandten Heuristiken liefern trotzdem gute Ergebnisse. Ausgefiltert werden:

- Icons: Anhand der Größe sind Icons leicht zu erkennen und auszufiltern.
- Banner: Sie werden an ihrem spezifischen Format[4] erkannt.
- Zeichnungen: Im Gegensatz zu eingescannten Photos oder Bildern einer Digitalkamera besitzen Zeichnungen meist nur eine geringe Farbtiefe. Wir filtern Bilder mit weniger als fünf Bit Farbtiefe aus.
- Thumbnails und Previews: Aus Breite, Höhe und Farbtiefe eines Bildes läßt sich berechnen, wie groß die Datei in unkomprimierten Zustand sein müßte. Weicht die tatsächliche Dateigröße erheblich davon ab, kann das Bild ausgefiltert werden[5].

Um sowohl Zeit zu sparen als auch unnötige Netzlast zu vermeiden, erscheint es sinnvoll, bereits evaluierte Daten zu speichern. In Java bietet die JDBC[6] Schnittstelle eine komfortable Möglichkeit, auf Datenbanken zuzugreifen.
Neben der Architektur des Personal Picture Finder, die vollständig in Java realisiert ist, gibt es zwei weitere Komponenten. Zum einen ist dies eine Datenbank, die die Verwaltung der anfallenden Datenmengen schneller, effizienter und stabiler verwalten kann, als es uns mit den Datenstrukturen in Java mit vertretbarem Aufwand möglich gewesen wäre. Zum anderen wird momentan eine lernende Komponente getestet, die anhand von Feedback durch den Benutzer die Leistung der Filter verbessern soll (siehe Abschnitt 6).
Ein wichtiger Aspekt beim Entwurf eines Internet-Agenten ist die Frage der Stabilität. Sie garantiert auch bei vielen Anfragen gute Performanz. Unsere Designentscheidungen basieren auf folgenden Überlegungen:

1. Der Agent soll soweit wie möglich unabhängig von Browser oder Betriebssystem sein.

[4]Ein Banner ist mindestens doppelt so breit wie hoch.
[5]Der Schwellwert des Filters hängt vom Format der Datei ab. JPEG beispielsweise liefert auch bei sehr hoher Kompression noch brauchbare Ergebnisse.
[6]JDBC ist das Akronym für Java DataBase Connectivity.

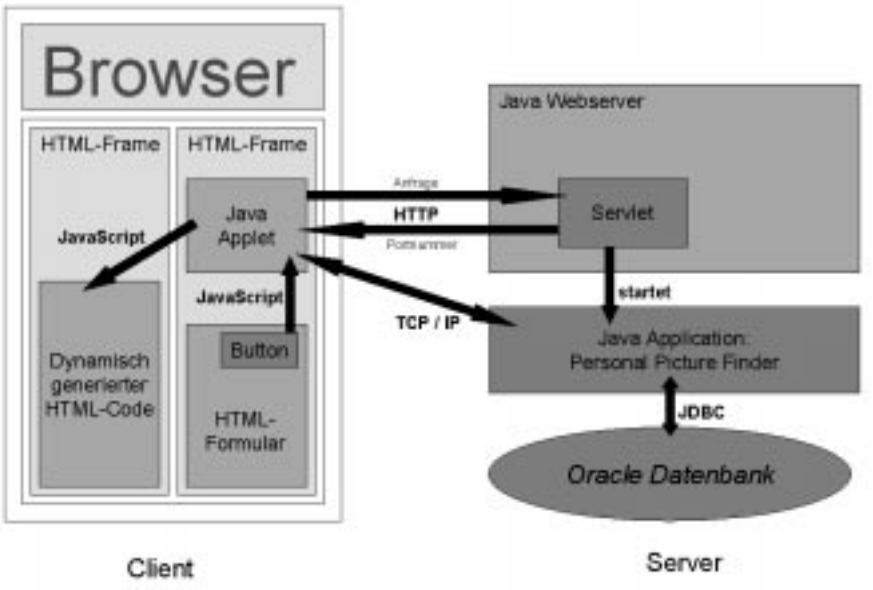

Abbildung 5: Datenfluß im Personal Picture Finder

2. Auf Client-Seite soll möglichst wenig gerechnet werden. Der Datenfluß zum Browser ist minimal.

3. Zugriffe in das WWW sollen ohne Vorbereitungen auf der Client-Seite möglich sein. Deswegen werden beispielsweise keine signierten Applets benutzt.

4. Probleme oder Fehler bei einer Anfrage dürfen keine Auswirkungen auf zeitgleiche oder spätere Anfragen haben.

Der Datenfluß ist in Abbildung 5 beschrieben.
Die Benutzeranfrage aus dem HTML-Formular wird von einer JavaScript-Funktion an das Applet übergeben. Ein vom Applet aufgerufenes Servlet startet einen "Personal Picture Finder"-Prozeß über eine TCP/IP-Schnittstelle und gibt dem Applet die Portnummer dieser Schnittstelle zurück. Das Applet öffnet eine Verbindung und liest von diesem Port. Die Information wird geparst und entsprechend visualisiert, und zwar entweder, indem die Anzeige in dem Control-Panel des Applet verändert wird, oder aber indem ein Bild angezeigt wird. Soll ein Bild angezeigt werden, ruft das Applet eine JavaScript-Funktion auf, die den entsprechenden HTML-Code erzeugt und in den Ausgabeframe schreibt. Der Benutzer kann den Prozeß jederzeit über das Applet terminieren. Diese Architektur läuft derzeit auch bei hoher Belastung stabil und unabhängig von Plattform und Browser[7]. Probleme treten lediglich auf, wenn der Benutzer hinter einer *Firewall* sitzt, die keine TCP/IP-Verbindung gestattet.

[7]Dies bezieht sich auf die neueren Versionen der populärsten Browser. Veraltete Browserversionen werden nicht unterstützt.

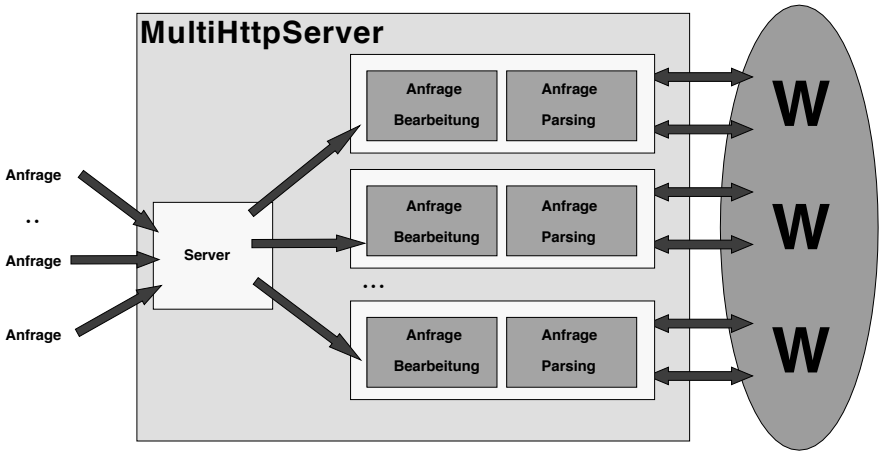

Abbildung 6: Architektur eines einzelnen MultiHttpServers

4 Parallel Pull Engine

Wie bereits im letzten Abschnitt erwähnt, ist die Parallel Pull Engine ein wesentlicher Bestandteil des Personal Picture Finder. In diesem Abschnitt wird ihre Funktionsweise am Beispiel des *MultiHttpServer* erläutert. Der MultiHttpServer wurde im Projekt PAN entwickelt und erfolgreich in mehreren Anwendungen eingesetzt. Der MultiHttpServer wird ausführlich in [End99] dokumentiert.

Abbildung 6 zeigt die Architektur eines einzelnen MultiHttpServer. Er besteht aus zwei Hauptkomponenten. Der Server nimmt die Anfragen vom Benutzer über eine TCP/IP-Schnittstelle an. Für jede geöffnete Verbindung wird eine Instanz eines *Servicemodules* erzeugt. Diese besteht wiederum aus zwei Teilen: Ein *Anfrage Parsing-Modul* übernimmt den Dialog mit dem Client nach einem festgelegten Protokoll, parst dessen Anfragen und übergibt sie zur Ausführung an das *Anfragebearbeitungs-Modul*. Dieses leistet die eigentliche Arbeit und übernimmt die Anfragen ins WWW. Da die MultiHttpServer-Instanz nebenläufig ist, können mehrere Anfragen gleichzeitig bearbeitet werden. Die Zeitersparnis bei parallelen Zugriffen gegenüber sequentiellen Zugriffen ist essentiell für die Performanz des Internet-Agenten.
Obwohl Java durch die Benutzung der *green threads* unabhängig ist von den Systembeschränkungen[8], sollten sinnvollerweise nicht zuviele Threads zeitgleich laufen. Ist eine größere Performanz nötig, kann eine neue Instanz des MultiHttpServer erzeugt werden. Diese Funktion übernimmt ein *Scheduler*, wie in Abbildung 7 dargestellt. Der Client fragt nun nicht mehr direkt beim MultiHttpServer an, sondern zuerst beim Scheduler, der ihm

[8]Solaris Betriebssyteme erlauben einem einzelnen Prozess defaultmäßig nur 64 Threads. Diese Schranke wird in /etc/system festgelegt und kann vom Systemverwalter geändert werden. Aus technischen Gründen ist es allerdings nicht sinnvoll, diese Schranke zu weit hochzusetzen. Die daraus resultierenden Nebeneffekte können die Leistungsfähigkeit des Systems gefährden.

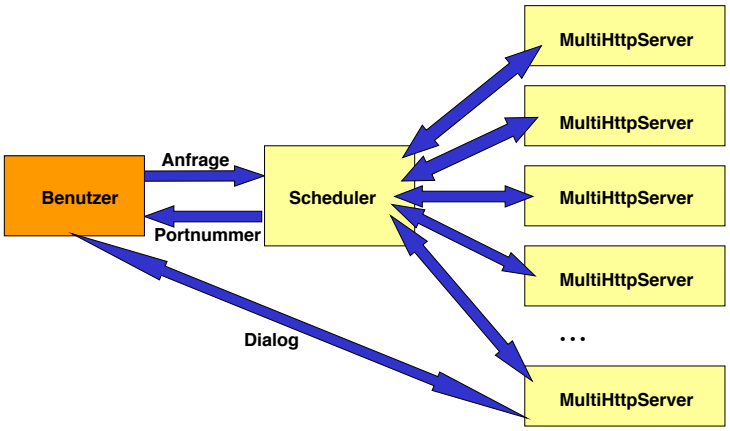

Abbildung 7: Scheduling von Anfragen (mehrere MultiHttpServer)

einen MultiHttpServer zuweist und dessen Adresse angibt.

Der Client verbindet sich zu dem ihm zugewiesenen MultiHttpServer und läßt seine Anfrage bearbeiten. Durch den ständigen Dialog mit den einzelnen MultiHttpServer-Instanzen kann der Scheduler erkennen, wann die Gesamtauslastung zu hoch ist, und er eine neue MultiHttpServer-Instanz generieren muß. Ebenso kann er nicht mehr benötigte MultiHttpServer-Instanzen terminieren. Sollte trotz aller Vorsichtsmaßnahmen eine MultiHttpServer-Instanz überlastet werden und ausfallen, kann der Scheduler sie terminieren und eine neue Instanz erzeugen. Der Scheduler kann über eine Konfigurationsdatei an die Systemressourcen und die Anforderungen des Benutzers angepaßt werden.

5 Statistische Auswertung

Die Leistungsfähigkeit eines Suchdienstes im Internet zu beurteilen, ist sehr schwierig, da viele nicht vorhersehbare Umstände die Qualität der Ergebnisse beeinflussen können. Zwei klassische Maßeinheiten, die sich auch für Netbots als nützlich erwiesen haben, sind *precision* und *recall* [SLE97]:

- Precision mißt das Verhältnis von relevanter zu irrelevanter Information , die während einer Suchperiode gefunden wird.
- Recall mißt das Verhältnis von relevanter Information, die gefunden wird, zu überhaupt existierender relevanter Information im Internet.

Es ist sehr schwierig, eine Recall-Bewertung für den Personal Picture Finder vorzunehmen, da man nur messen kann, wieviel relevante Information gefunden wird, nicht jedoch, wieviel überhaupt im Internet existiert. Wir beschränken uns deshalb im folgenden auf die Precision-Bewertung. Abbildung 8 zeigt eine Statistik der Ergebnisse des Personal

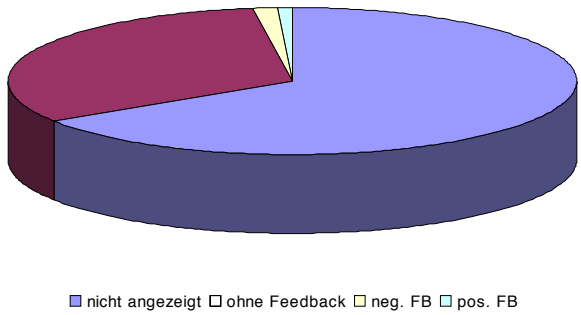

Abbildung 8: Angezeigte und aussortierte Bilder

Picture Finder. Von 26556 gefundenen Bildern werden nur 8676 angezeigt. Das entspricht etwa 33 Prozent. Die restlichen Graphiken werden von den Filtern aussortiert.
Abbildung 9 zeigt einen Vergleich der wichtigsten bisher benutzten Filter. Der größte Teil der aussortierten Bilder wird von dem Icon-Filter abgelehnt, etwa 83 Prozent. Als Thumbnails werden etwa 79 Prozent der aussortierten Bilder erkannt. Zeichnungen (49 Prozent) und Banner (48 Prozent) filtern jeweils knapp die Hälfte der Bilder. Man sieht hier, daß im Durchschnitt jedes abgelehnte Bild von mehr als zwei Filtern verworfen wurde.

6 Selbstoptimierung des Netbots durch maschinelles Lernen

Die positiven und negativen Beurteilungen des Benutzers werden im Personal Picture Finder genutzt, um durch verschiedene Lernverfahren[Mit97] die Systemleistung laufend zu verbessern. Die einfachste Form der Selbstoptimierung ist dadurch realisiert worden, daß in einer Datenbank alle Verweise auf Bilddateien zu einer bearbeiteten Suchanfrage gespeichert werden, die zwar als Suchergebnis angezeigt, dann aber vom Benutzer verworfen wurden. Wenn die gleiche Suchanfrage (meist wohl von einem anderen Benutzer) nochmals gestellt wird, werden die in der Datenbank mehrfach als unpassende Bilddateien gekennzeichneten Verweise vom System erst gar nicht mehr weiter verfolgt. Andererseits werden die Verweise auf überwiegend positiv bewertete Bilddateien natürlich herangezogen, um einen Caching Effekt zu erzielen und die aktuell neu gefundenen Bilder durch garantierte Erfolgskandidaten anzureichern. Liegen weder eindeutig positive noch negative Urteile über eine einmal gefundene Bilddatei vor, so nutzt der Netbot den entsprechenden Datenbankverweis für einen Memoization Effect, um die Filterkosten zu minimieren. Interessanter ist eine im Personal Picture Finder zusätzlich realisierte Form des sog. Reinforcement Learnings. Diese Form des Lernens ist dadurch charakterisiert, daß bei der initialen Implementierung des Netbots keine Trainingsdaten für das Erlernen der Zielfunktion vorliegen. Erst bei der Anwendung des Netbots entstehen durch die vom Benutzer erbetenen Evaluationen der Suchergebnisse inkrementell Trainingsdaten, die für maschinelle

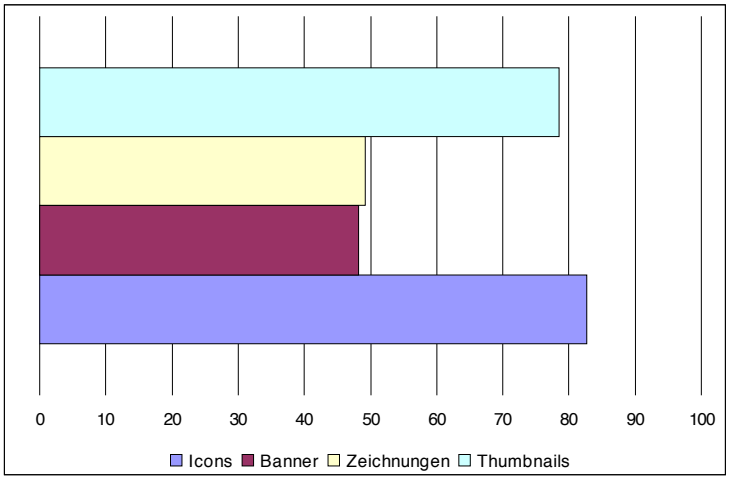

Abbildung 9: Häufigkeit der Filteranwendungen

Lernverfahren verwertbar sind. Für jede ausgeführte Suchaktion gibt es eine Belohnung in Form positiver Benutzerbeurteilungen oder eine Bestrafung durch Negativurteile. Das Lernziel des Agenten ist es, eine Suchstrategie durch Optimierung und Gewichtung der Bildfilter zu finden, die eine optimale Belohnung sichert.
Abbildung 10 zeigt einen Ausschnitt von einem aus Benutzerbeurteilungen gelernten Entscheidungsbaum. Dabei kam das klassische Induktionsverfahren ID3 zum Einsatz. Der Entscheidungsbaum entsteht durch eine Diskretisierung von Schwellwertintervallen für die verschiedenen Filter. So ist in Abbildung 10 für die Farbtiefe z.B. eine Intervallaufteilung in Zeichnung, Cartoon, Photo und Truecolor zu sehen. Die im Entscheidungsbaum angezeigten Fehlerabschätzungen beziehen sich bereits auf anhand von Testdaten ermittelte Ergebnisse nach einer durch Lernen optimierten Filterkaskade. Insgesamt können Induktionsverfahren dazu verwendet werden, um Schwellwerte in Filtern neu zu justieren, neuartige Kombination von Filterergebnissen zu finden oder auch neue Gewichtungen für Filter zu lernen. Die Experimente dazu werden derzeit noch durchgeführt.

7 Ausblick

Der Personal Picture Finder ist eine Modellanwendung, bei der die Basistechnologien zur Konstruktion von Netbots untersucht werden. Das System ist unter der URL finder.dfki.de:7000 seit Juni 1998 verfügbar und wird aufgrund verschiedener Presseberichte derzeit verstärkt benutzt. Für die Lernverfahren werden aber Massendaten mit entsprechender Evaluation benötigt, so daß in den nächsten Monaten noch erhebliche Verbesserungen durch die Selbstoptimierung zu erwarten sind. Um die Recall-Rate noch weiter zu verbessern, könnte analog zu dem Netbot Ahoy! ein wissensbasierter URL-Generator integriert werden, der anhand syntaktischer Variationen von typischen URL-Mustern gezielt nach nicht indizierten Bilddateien sucht. Die für das Training der Lernverfahren angelegte

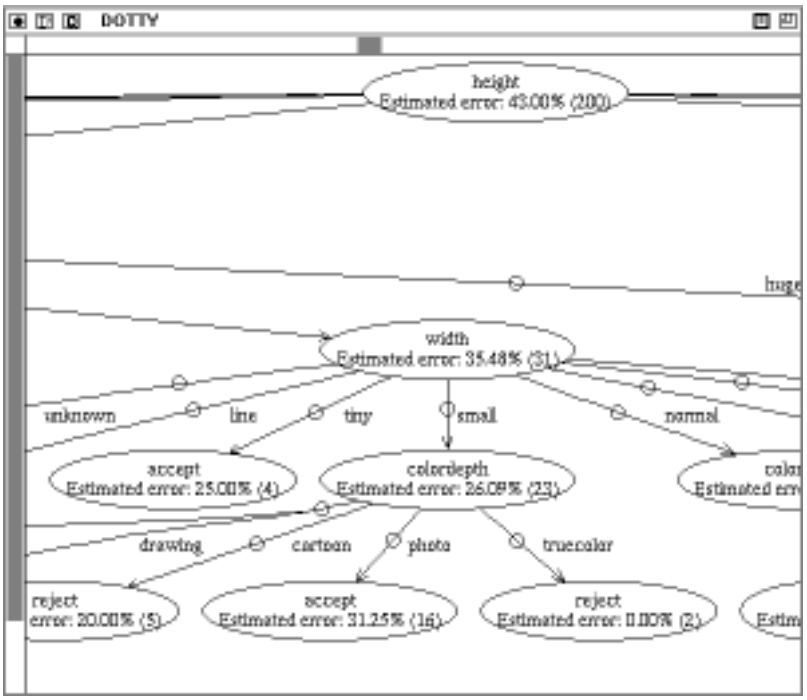

Abbildung 10: Aus Beurteilungen gelernter Entscheidungsbaum

Datenbank ist eine gute Ausgangsbasis zur Erkennung solcher Muster. Selbstverständlich könnte die Precision-Rate durch aufwendige Filtermechanismen noch erheblich verbessert werden. Im Extremfall könnten etwa Algorithmen zur Gesichtserkennung eingesetzt werden. Einem solchen Vorgehen steht natürlich die zu erzielende hohe Geschwindigkeit bei der Filterung im Wege. Es wäre aber denkbar, im Sinne eines benutzergesteuerten "Quality of Service" maximale Bearbeitungszeiten durch den Benutzer vorgeben zu lassen. Höchste Ansprüche an die Resultatsqualität führen dann dazu, daß der Netbot als Hintergrundprozeß arbeitet und den Benutzer nach Abschluß der Suche durch email oder andere Push-Technologien über das Ergebnis informiert.

Literatur

[BD98] Bauer, M. und D. Dengler: *TrIAs: An Architecture for Trainable Information Assistants*. In: *Proceedings of the Agents98 Workshop on "Personal Information Assistants"*, 1998.

[BD99] Bauer, M. und D. Dengler: *InfoBeans - Configuration of Personalized Information Assistants*. In: *Proceedings of the 1999 Conference on Intelligent User Interfaces*, 1999.

[End99] Endres, C.: *The MultiHttpServer - A Parallel Pull Engine*. Technischer Bericht Deutsches Forschungszentrum für Künstliche Intelligenz, 1999. Erscheint.

[Mit97] MITCHELL, T.: *Machine Learning.* McGraw-Hill, 1997.

[SLE97] SHAKES, J., M. LANGHEINRICH und O. ETZIONI: *Dynamic Reference Sifting: A Case Study in the Homepage Domain.* In: *Proceedings of the Sixth International World Wide Web Conference*, Seiten 189–200, 1997.