Infrastruktur für Web Intelligent Systems

Thema: Business Intelligence –
Teil II: Data Mining & Knowledge Discovery

von Christian Merker

Betreuer: Dipl.-Inform. Marcus Flehmig

13. Januar 2004

Inhalt

1. Motivation

Im Zeitalter des Internets in dem alles schneller funktioniert, trifft man immer wieder auf neue Schlagworte, die bei Anbietern von Online-Diensten eine regelrechte Euphorie auslösen. Jeder schmückt sich und seinen Online-Dienst mit Aussagen wie „Wir verwenden ein Business-Intelligent-System“ oder „Unsere Web-Intelligent-Plattform verwendet die modernsten Verfahren“.

Doch es stellt sich die Frage: Was steckt hinter den Begriffen Business Intelligence bzw. Web Intelligence? Das Ziel der folgenden Seiten ist es einen groben Einblick in die Welt der Web-Intelligent-Systeme zu geben. Wir werden uns auch nur auf die Web-Intelligent-Systeme konzentrieren.

Zuerst werden wir uns mit dem Begriff Web Intelligence in Kapitel 2 näher beschäftigen. Wir werden versuchen eine Definition für den Begriff zu geben und die grundlegenden Ideen dieses Konzepts herauszustellen.

Anschließend werden wir in Kapitel 3 näher auf die Verfahren, die in Web-Intelligent-Systemen verwendet werden und deren Leistungsfähigkeit eingehen. Das Kapitel 3 gliedert sich in drei größere Blöcke. Beginnen wollen wir mit einer Überlegung, welche Charakteristik die Daten von Web-Intelligent-Systemen haben und woher diese Daten kommen. Anschließend betrachten wir vier Verfahren, die aus diesen Daten Ähnlichkeitsbeziehungen berechnen. Zum Abschluss dieses Kapitels werden wir noch zwei konkrete Anwendungsbeispiele vorstellen, welche die vorher beschriebenen Verfahren verwenden.

In Kapitel 4 beschäftigen wir uns mit Prefetching, d. h. dem Vorladen von Daten die in naher Zukunft angefordert werden. Auch hier werden wir zuerst eine Begriffsklärung vornehmen und im Anschluss mehrere verschiedene Möglichkeiten aufzeigen, an welchen Stellen Prefetching sinnvoll ist. Wir betrachten dabei das Prefetching auf Server-Seite, ebenso wie auf Client-Seite bzw. in Proxies zwischengelagert. Dabei werden wir immer wieder Bezug auf unsere Erkenntnisse aus den vorherigen Kapiteln nehmen, um die Einsatzmöglichkeiten von Web Intelligence Verfahren im Bereich Prefetching darzustellen.

2. Web Intelligence

Business Intelligence und Web Intelligence sind Schlagworte, die in der heutigen Zeit häufig von Firmen verwendet werden, um das Interesse von Käufern, insbesondere anderen Firmen die Online-Verkaufsplattformen anbieten, zu erhalten. Es hat sich mittlerweile schon fast zu einem Trend entwickelt: Jeder bietet Business- und Web – Intelligence-Lösungen an, doch es stellt sich immer die Frage: Was Web- und Business-Intelligent-Systeme sind und warum sie eingesetzt werden und wie sie funktionieren. Im Folgenden werden wir speziell auf Web-Intelligent-Systeme eingehen. Dieses Kapitel versucht die ersten beiden Fragen zu beantworten. Auf die Methoden zur Realisierung wird im folgenden Kapitel ausführlich eingegangen.

2.1. Definition

Eine Definition für Web Intelligence zu geben, stellt sich als recht schwierig dar. Als Laie findet man eine Vielzahl von Definitionen, die sich zwar in ihren Formulierungen und ihrer Aussage sehr unterscheiden, jedoch kann man in allen einen gemeinsamen Grundgedanken entdecken: Die Idee hinter Web-Intelligent-Systemen ist, Daten über Besucher von Internet-Seiten zu sammeln und diese Daten aktiv zu verwenden, um auf die Interessen und Vorlieben des Besuchers zu schließen und damit ein Profil des Besuchers zu erhalten. Dadurch kann sich ein klarer Wettbewerbsvorteil ergeben [Vinc01, Tan03]. Eine weitere Einsatzmöglichkeit für Web-Intelligent-Systeme sind so genannte Internet-Portale. Bei diesen Portalen ist es dem Benutzer möglich den Benutzerbereich nach seinen Bedürfnissen anzupassen, um so schnell an die von ihm gewünschten Informationen zu gelangen. Ein Web-Intelligent-System besteht aus Komponenten, die Algorithmen zur Ähnlichkeitssuche implementieren. Die Komponenten sind dafür zuständig die Daten über die Benutzer zu sammeln, diese Daten auszuwerten und daraus Benutzerprofile zu erstellen. Die erzeugten Benutzerprofile werden vom Web-Intelligent-System anschließend verwendet um Ähnlichkeitsvergleiche durchzuführen. Ein Web-Intelligent-System stellt die oben genannten Funktionen bereit und kann in bestehende Systeme, beispielsweise in E-Commerce-Systeme integriert werden.

Es ist leicht ersichtlich warum vor allem Anbieter von Online-Verkaufsplattformen ein reges Interesse an solchen Systemen haben. Web-Intelligent-Systeme werden daher meistens als Marketing-Instrument eingesetzt, um damit das gesammelte Wissen über Besucher zu verwenden, und die Besucher auf interessante Produkte bzw.

Angebote aufmerksam zu machen. Ein Beispiel hierfür ist z. B. der Onlinebuchhandel Amazon.com[1]. Dort wird Besuchern beim Kauf, bzw. schon bei Auswahl eines Produkts, gleich eine Liste mit Produkten mitgeliefert, die ebenfalls für den Besucher von Interesse sein könnten. Auf die dahinter stehenden Verfahren gehen wir in Kapitel 3.2. nochmals ausführlich ein. Mit Hilfe der gesammelten Daten ist es möglich eine Personalisierung durchzuführen, d. h. den Benutzer mit Informationen zu versorgen, die speziell für seine Person von Interesse sind.

2.2. Personalisiertes Web

Nachdem nun bekannt ist, was ein Web-Intelligent-System ist und für welche Zwecke es eingesetzt wird, wissen wir noch immer nicht, woher die Daten über die Benutzer herkommen.

Diese „persönlichen" Daten erhalten die Firmen zum einen durch aktive Beteiligung des Benutzers, z. B. durch Bewertungsfragebogen, die der Benutzer nach dem Kauf eines Produkts ausfüllen kann und dem Unternehmen zuschickt. Im Falle von Web-Intelligent-Systemen geschieht dies fast ausschließlich über das Medium Internet. Allerdings gibt es auch Unternehmen, die Fragebögen mit der Bestellung mitschicken und der Käufer diese dann per Post gebührenfrei zurückschicken kann.

Eine andere Möglichkeit ist, die „Digitalen Fußspuren" des Benutzers zu analysieren und daraus Daten zu erhalten. Mit „Digitalen Fußspuren" bezeichnet man die Aktivitäten, die ein Benutzer auf einer Internetseite oder mehreren Internetseiten durchführt. Solche Aktivitäten sind beispielsweise die Hyperlinks, die ein Benutzer auf einer Seite anklickt, die Verweildauer die er auf der Seite verbringt oder wie oft er die Seite in einem bestimmten Zeitraum besucht hat usw. [BS01].

Bei dieser Art der Datenerhebung ist dem Benutzer nicht bekannt, welche Daten über ihn gesammelt werden. Mit diesen Daten ist es allerdings auch nur möglich das statistische Verhalten des Benutzers als Profil auszuwerten.

Wie schon in Abschnitt 2.1 erwähnt, werden die gesammelten Daten meist dafür verwendet, den Besucher auf einer Seite persönlich anzusprechen und die Seite nach seinen Wünschen zu gestalten, d. h. der Benutzer erhält beispielsweise beim Aufruf einer Internetseite mit Börsendaten alle Aktienkurse, die er bei vorherigen Besuchen angeschaut oder ausgewählt hatte. Durch den Einsatz beispielsweise von sogenannten Cookies ist es möglich den Benutzer eindeutig im System zu identifizie-

[1] http://www.amazon.com

ren. Diese Cookies werden auf Client-Seite gespeichert und können Daten über den Benutzer enthalten, oder Informationen über die besuchten Internetseiten des Benutzers. Beim Aufruf der Internetseite werden dann die Daten aus dem Cookie ausgelesen und vom System verarbeitet.
Ziel ist es, dem Benutzer die Daten, die er benötigt und die für ihn wichtig sind, schnell und benutzerfreundlich zur Verfügung zu stellen. In einigen Systemen, so genannten personalisierten Portale, ist es sogar möglich, die Oberfläche nach den persönlichen Bedürfnissen anzupassen. Dadurch erhält der Benutzer die Möglichkeit schnell zu den Funktionen und Inhalten zu gelangen, die für ihn interessant sind [INet03].
Allerdings gibt es auch hier Schattenseiten. Wenn z. B. durch Bewertungsfragebogen die Adressen oder E-Mail Adressen von Benutzern vorhanden sind, werden diese Informationen ebenfalls dafür benutzt, um den Benutzer mittels E-Mail oder Postwurf auf interessante Produkte hinzuweisen. Des Weiteren werden diese Benutzerprofile auch oft zwischen verschiedenen Firmen ausgetauscht oder teilweise sogar verkauft [BS01]. Dies kann dann zur Folge haben, dass ein Benutzer der bei Unternehmen A eine Bestellung für Gartengeräte aufgegeben hat, nach einer Weile von weiteren Unternehmen mit E-Mails überflutet wird, die alle darauf abzielen ihre Gartengeräte bzw. ihre Produkte für den Garten an den Benutzer zu verkaufen.

2.3. Infrastruktur und Datenhaltung

Um Web-Intelligent-Systeme professionell und erfolgreich zu betreiben benötigt man eine leistungsfähige Infrastruktur. Ein Benutzer erzeugt durch seine Aktivitäten auf einer Internetseite große Datenmengen in Form von so genannten „Click-Stream“-Daten. Wenn man bedenkt, dass eine Internetseite eines großen Anbieters von Tausenden von Benutzern gleichzeitig besucht wird, kann man sich leicht vorstellen, dass dadurch riesige Datenmengen in Form von „Click-Streams“ entstehen, die verarbeitet und gespeichert werden müssen. Hierarchische Datenstrukturen helfen hier, diese Daten sinnvoll und nach Zugriffshäufigkeit zu speichern. Weiterhin können diese Datenmengen durch Herausfiltern unnötiger Daten oder durch Kompression reduziert werden [INet02]. Häufig verwendet man eine Speicherhierarchie, wie in Abbildung 2.1 dargestellt, bestehend aus Web-Server mit integrierter Datenbank, einem Data Warehouse und einem Near-Line-Speicher (Third-Level-Speicher).

Abbildung 2. 1

Der Web-Server ist dafür verantwortlich die aktuellen Daten über die Benutzer zu verwalten. Dabei wird beim Web-Server besonderes Augenmerk auf den Durchsatz gelegt, da ansonsten für den Benutzer lange Wartezeiten beim Aufruf einer Seite entstehen und die Benutzerfreundlichkeit darunter leidet. In dieser Schicht befinden sich auch immer die aktuellsten Benutzerdaten, die meistens nicht älter als 24 Stunden sind. Um Speicherplatz zu sparen werden die eintreffenden „Click-Stream"-Daten analysiert, unnötige Daten herausgefiltert und anschließend in der Datenbank des Web-Servers gespeichert.

In regelmäßigen Abständen, meistens einmal am Tag, werden die Benutzerdaten vom Web-Server in das Data Warehouse übertragen. Dabei werden diese Daten nochmals analysiert und überflüssige Daten entfernt. Durch die regelmäßige Einlagerung der Benutzerdaten ins Data Warehouse wird wieder Speicherplatz auf der Datenbank des Webservers geschaffen und es wird verhindert, dass dieser mit Daten über die Benutzer überschwemmt wird [INet02]. Im Data Warehouse verbleiben die Benutzerdaten im Schnitt zwischen 6 Monaten und einem Jahr.

Als letztes werden die Benutzerdaten auf einem Third-Level-Speicher übertragen. Da Speicherkapazität unbegrenzt ist und die Kosten für Speicher mittlerweile gering sind, ist es möglich diese Daten im Third-Level-Speicher dauerhaft d. h. über Jahre bzw. Jahrzehnte zu speichern.

Durch diese Speicherungshierarchie ist gewährleistet, dass der Zugriff auf benötigte Benutzerdaten immer schnell erfolgen kann. Benutzerdaten mit einer sehr hohen Zugriffswahrscheinlichkeit befinden sich im Web-Server bzw. in der angebundenen Datenbank. Das Data Warehouse verwaltet die Daten mit einer mittleren Zugriffswahrscheinlichkeit und alle Benutzerdaten, die eine geringe Zugriffswahrscheinlichkeit haben befinden sich im Third-Level-Speicher.

3. Algorithmen zur Datenanalyse

Nachdem wir im letzten Abschnitt die allgemeinen Konzepte betrachtet haben um Benutzerdaten zu gewinnen, werden wir in diesem Abschnitt näher auf die Algorithmen zur Ähnlichkeitsbestimmung zwischen Benutzerprofilen eingehen. Die Algorithmen, die in Web-Intelligent-Systemen eingesetzt werden, können aus dem Information Retrieval übernommen werden. In beiden Systemen wird eine Ähnlichkeitssuche auf vorhandenen operationalen Daten bzw. analysierten Daten über Benutzerverhalten durchgeführt. Die Erfahrungswerte aus dem Information Retrieval, z. B. Leistungsmaße, Ähnlichkeitsmaße, können daher bei Web-Intelligent-Systemen verwendet werden. Der einzige Unterschied zwischen beiden Systemen ist, dass der Benutzer in Information-Retrieval-Systemen aktiv eine Anfrage stellt, diese Eingabe mit bestehenden Dokumenten oder Daten auf Ähnlichkeit untersucht wird und der Benutzer ein Ergebnis erhält. Bei Web-Intelligent-Systemen ist der Benutzer in gewissem Sinn passiv, er wählt zwar auch z. B. ein Produkt aus einen Produktkatalog aus, aber das System startet selbstständig eine Ähnlichkeitssuche um dem Benutzer weitere, für ihn interessante Produkte zu finden. Da beide Systeme Algorithmen zur Ähnlichkeitssuche verwenden, ist es nicht ungewöhnlich dass Algorithmen, die sich im Information Retrieval bewährt haben, auch für Web-Intelligent-Systeme interessant sind. Im Folgenden werden wir uns allerdings nur auf Algorithmen bzw. Verfahren beschränken die in Web-Intelligent-Systemen eingesetzt werden.

3.1. Datencharakteristika

Die Charakteristik der operationalen Daten spielt in einem Web-Intelligent-System eine wichtige Rolle. Man muss zwischen verschiedenen Typen von Daten unterscheiden

- Textdaten: Die operationalen Daten liegen in einem Textformat vor und können durch bekannte Retrievalverfahren für Text bearbeitet werden.
- Audiodaten: Hier beinhalten die operationalen Daten Tonaufnahmen, die durch Untersuchung von Frequenzverteilungen oder durch Vergleich der mittleren Amplitude mit anderen Audiodaten auf Ähnlichkeit hin untersucht werden.
- Bilddaten: Die operationalen Daten liegen in Form von Pixelmatrizen vor und müssen mit Verfahren wie Farbhistogrammvergleiche oder Textursuchen auf Ähnlichkeit überprüft werden.

- Multimediadaten: Hier bestehen die operationalen Daten aus einer Kombination der vorherigen 3 genannten Datentypen (z. B. Videodaten), d. h. für die Ähnlichkeitssuche müssen verschiedene Verfahren aus den anderen Datentypbereichen kombiniert werden, um vernünftige Ergebnisse zu erhalten.

In diesem Dokument werden wir nur Textdaten betrachten, da in der Praxis die „Click-Stream"-Daten fast ausschließlich nur aus Textdaten bestehen und die Verfahren für Textretrieval am weitesten entwickelt sind.

Um die riesigen „Click-Stream"-Daten zu verarbeiten, die durch die Benutzer auf einer Internetseite entstehen, können zwei Techniken eingesetzt werden [KPZ03]:

- Inkrementelle Algorithmen: Diese Algorithmen ermöglichen es neue Daten in die bereits bestehenden Daten bzw. Profile zu integrieren, ohne dass alle vorhandenen Daten nochmals analysiert und bearbeitet werden müssen. Diese Algorithmen werden sehr häufig im Bereich des maschinellen Lernens eingesetzt.
- Datenfilterung: Diese Technik erzeugt aus den vorhandenen Daten eine repräsentative kleinere Datenmenge, mit der anschließend gearbeitet wird. Die Daten werden für eine spätere Bearbeitung und Veränderung vorbereitet, d. h. unnötige Daten und Rauschen werden entfernt. Allerdings verlieren die Daten durch diesen Prozess auch an Qualität, was zu Verfälschungen der Ausgabe des Algorithmus führen kann.

3.2. Algorithmen und Methoden

Es gibt eine Reihe von Algorithmen zur Bestimmung von Ähnlichkeit zwischen zwei oder mehreren Datenbeständen. Diese Algorithmen lassen sich in 4 Klassen einteilen [LSY03]:

- Kollaborative Filter [Heyl01]
- Cluster-Verfahren [Rijs99]
- Suchbasierende Verfahren
- Item-to-Item Collaborative Filtering [LSY03]

In den nachstehenden Kapiteln werden wir auf die einzelnen Klassen explizit eingehen und einen Vergleich zwischen den Klassen betrachtet.

3.2.1. Kollaborative Filter

Bei den traditionellen kollaborativen Filteralgorithmen werden die gesammelten Daten über einen Benutzer (z. B. bereits gekaufte Produkte) als ein N-dimensionaler Vektor dargestellt, wobei N die Anzahl der unterschiedlichen Produkte in einem Produktkatalog angibt. Alle Produkte, die der Benutzer bereits gekauft und für die er eine positive Bewertung abgegeben hat, erhalten einen positiven Wert; negative Bewertungen erhalten im Vektor einen negativen Wert. Noch nicht gekaufte Produkte werden weggelassen oder erhalten den Wert „0“. Die Idee zur Ähnlichkeitssuche besteht nun darin, dass man die Annahme trifft, dass zwei Benutzer sich ähnlich sind bzw. ähnliche Interessen besitzen, wenn sie ähnliche Bewertungen für die gleichen Produkte abgegeben haben. Um die Ähnlichkeit zwischen zwei Benutzern zu bestimmen muss man lediglich die beiden Benutzervektoren vergleichen. Wenn das Ergebnis einen vorher definierten Schwellwert überschreitet, geht man davon aus, dass die Benutzer ähnliche Interessen besitzen. Üblicherweise verwendet man zum Ähnlichkeitsvergleich zweier Vektoren das Cosinus-Maß.

Das Cosinus-Maß ist wie folgt definiert:

$$\cos\left(\vec{A},\vec{B}\right)=\frac{\vec{A}\cdot\vec{B}}{\left\|\vec{A}\right\|\cdot\left\|\vec{B}\right\|}$$

Abbildung 3. 1

Wobei A und B zwei Benutzervektoren sind deren Ähnlichkeit verglichen werden soll. Das Produkt der beiden Vektoren wird durch die euklidischen Längen der Vektoren A und B geteilt. Dadurch lässt sich die Formel auch als einfaches Skalarprodukt der normierten Vektoren interpretieren [Rijs99]. Der Wertebereich der Funktion liegt im Intervall [-1,1]. Je größer das Ergebnis ist, desto ähnlicher sind die Vektoren, d. h. bei Gleichheit der beiden Vektoren erhält man maximale Ähnlichkeit, was einem Wert von 1 entspricht. Die Ergebnisse der Formel aus Abbildung 3.1 lassen sich graphisch wie in Abbildung 3.2 veranschaulichen. Dabei bilden alle Punkte gleicher Ähnlichkeit eine Hyperebene und durch die Normierung der Vektoren erhalten wir nur Werte, die sich auf dem Einheitskreis befinden.

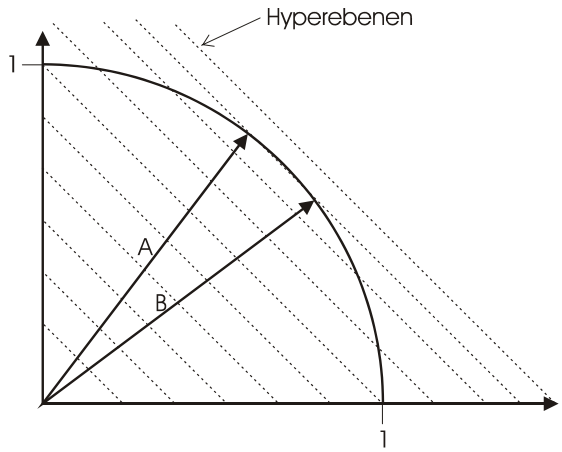

Abbildung 3. 2

Diese Ähnlichkeitsbetrachtung muss nun für alle Benutzer, die das System während der Laufzeit verwaltet bzw. während der Benutzer mit dem System interagiert, durchgeführt werden. Geht man davon aus, dass ein solches System M Benutzer und N Produkteinträge verwaltet, so entsteht im Worst-Case eine Komplexität von $O(M \cdot N)$. Nimmt man weiterhin an, dass im Normalfall die Größenordnungen für M bzw. N im Bereich $10^6 - 10^7$ bzw. $10^4 - 10^6$ liegen, so wird schnell klar, dass dieses Verfahren sehr aufwändig ist und für Web-Intelligent-Systeme aus Leistungsgründen eher ungeeignet ist.

In [LSY03] werden einige Möglichkeiten aufgezeigt, um dieses Verfahren zu optimieren, z. B. indem man die Datenmenge reduziert, d. h. man führt den Vergleich nicht mit allen Benutzern durch, sondern betrachtet nur die Benutzer, die eine bestimmte Mindestanzahl an Produkten bereits gekauft haben. Eine andere Möglichkeit ist es, die Anzahl der Produkte, die verglichen werden müssen zu verkleinern, indem man die Produkte in Kategorien oder Produktgruppen einteilt und nur noch diese Einteilung betrachtet, wodurch man den Faktor N in unserer Komplexitätsbetrachtung verkleinert. Allerdings haben alle diese Optimierungsideen auch einen entscheidenden Nachteil: Sie reduzieren die Qualität unserer Berechnung und führen daher zu den oben genannten Verfälschungen der Ergebnisse.

3.2.2. Cluster-Verfahren

Eine Verbesserung des in 3.2.1. beschriebenen Verfahrens bieten die Cluster-Verfahren. Das zugrunde liegende Verfahren ist zwar das gleiche welches bei den kollaborativen Filtern eingesetzt wird, allerdings werden hier die Benutzer in so genannte Cluster eingeteilt und anschließend wird der aktuelle Benutzer nur noch mit je einem Repräsentanten jedes Clusters verglichen. Ein Cluster ist eine Gruppe von

Objekten, in diesem Fall Benutzerprofile, die alle bestimmte Gemeinsamkeiten bzw. bestimmte Ähnlichkeit besitzen.
Die Einteilung der Benutzer in Cluster kann bereits im Vorfeld bzw. offline geschehen, um so während des Betriebs eine bessere Leistung zu erzielen. Als Cluster-Algorithmus werden meistens Formen von Greedy-Cluster-Verfahren verwendet. Der Algorithmus startet mit einer Menge von Objekten, die alle in verschiedene Klassen eingeteilt sind. Anschließend werden alle restlichen Objekte mit den bereits existierenden Clustern verglichen, und wenn das Objekt eine Ähnlichkeit besitzt, die größer dem definierten Schwellwert ist, wird es dem Cluster hinzugefügt.
Eine andere Variante des Algorithmus ist, dass die Objekte nicht eindeutig einem Cluster zugeordnet werden, sondern mittels einer Gewichtungsfunktion in Relation zu mehreren Clustern gesetzt werden.
Für jeden Cluster wird anschließend ein Zentroid erzeugt. Ein Zentroid entspricht dem Mittelpunkt eines Clusters, d. h. der Zentroid besitzt zu jedem anderen Objekt des Clusters minimalen Abstand. Während der Laufzeit wird nun der aktuelle Benutzer mit den Zentroiden verglichen und anschließend der Cluster mit dem ähnlichsten Zentroiden weiter untersucht. Dadurch muss nur noch ein Bruchteil der Benutzer des Systems zum Vergleich betrachtet werden und die Performanz steigt.
Obwohl die Leistung von Cluster-Verfahren wesentlich besser ist als die von kollaborativen Filterverfahren, so ist die Qualität der Ergebnisse doch schlechter. Die schlechte Qualität entsteht durch die Einteilung der Benutzer in Cluster. Dabei kann es passieren, dass eine Person durch den Vergleich mit den Zentroiden in einen Cluster gelangt, der nicht den optimal ähnlichsten Benutzer beinhaltet, wodurch ein optimales Ergebnis nicht mehr möglich ist.

3.2.3. Suchbasierte Verfahren

Einen ganz anderen Ansatz als 3.2.1. und 3.2.2. verfolgen die suchbasierenden Verfahren. Hierbei werden die von einem Kunden gekauften oder bewerteten Produkte untersucht und anhand von Produkteigenschaften (z. B. Autor, Darsteller, Produzent) wird eine Anfrage an das System gestellt. Diese Suchanfrage kann einer (SQL-) Anfrage an eine Datenbank entsprechen, wobei keine Ähnlichkeiten betrachtet werden, sondern nur solche Elemente zur Ergebnismenge hinzugefügt werden, die mit einem Schlüsselwort in der Anfrage exakt übereinstimmen.

Hat ein Benutzer zum Beispiel das Buch „Kampf dem Terror – Kampf dem Islam“ gekauft, so würde das System Bücher von Peter Scholl-Latour, sowie Bücher über das Thema Terror oder Islam heraussuchen und sie dem Benutzer als interessant anzeigen. Solange der Benutzer nur wenige Produkte gekauft hat funktionieren diese Verfahren sehr gut; hat der Benutzer allerdings schon hunderte verschiedene Produkte gekauft, so ist es ineffizient eine Anfrage mit eventuell Tausenden von Schlüsselworten an die Datenbank zu stellen. Die Ergebnismenge wird dann unter Umständen sehr groß, da alle Schlüsselworte mit logischem Oder verknüpft sind. Abhilfe kann hier eine Zusammenfassung von Schlüsselworten, oder eine Reduzierung der Schlüsselworte bringen. Durch die Reduzierung der Schlüsselwortmenge können aber eventuell prägnante Schlüsselworte wegfallen und die Ergebnismenge wird dadurch verfälscht, bzw. die Qualität des Ergebnisses wird schlechter.

3.2.4. Item-to-Item Collaborative Filtering

Dieses Verfahren ist eine Weiterentwicklung des suchbasierten Verfahrens aus 3.2.3. Dabei werden alle Produkte im Katalog im Vorfeld miteinander auf Ähnlichkeit überprüft. Für jedes Produkt wird eine Matrix erstellt in der die Ähnlichkeitswerte zu allen anderen Produkten eingetragen werden. Die Berechnung der Matrizen geschieht nicht zur Laufzeit. Als Grundlage für die Ähnlichkeitsbestimmung können bei Büchern z. B. Titel, Kurzbeschreibungen oder auch Volltexte verwendet werden. Dabei wird eine Schlagwortliste aus den vorhandenen Objekten erstellt und durch Anwendung von linguistischen Verfahren (Synonymbeseitigung, Eliminierung von Groß- und Kleinschreibung, Grundformreduktion, Eliminierung von Stoppworten) wird die Schlagwortliste kleiner und die wichtigsten, den Text kennzeichnenden Worte, erhalten eine höhere Gewichtung. Nachdem die Schlagwortlisten erstellt wurden, können die Objekte bzw. Dokumente, wie schon in 3.2.1. beschrieben, als N-dimensionaler Vektor dargestellt werden. N gibt die Anzahl der Worte in der Schlagwortliste an. Das Verfahren bestimmt nun mit Hilfe des Cosinus-Maßes die Ähnlichkeitswerte aller Dokumentpaare und speichert diese in einer Matrix bzw. mehreren Matrizen. Geht man davon aus, dass der Produktkatalog 1 Million Einträge enthält, so wird die Berechnung extrem aufwändig und benötigt viel Speicherplatz. Eine Optimierung dieses Verfahrens bietet der Pseudo-Code in Abbildung 3.3 [LSY03]:

For each item in product catalog I1
For each customer C who purchased I1
For each item I2 purchased by customer C
Record that a customer purchased I1 and I2
For each item I2
Compute the similarity between I1 and I2

Abbildung 3. 3

Durch diesen Algorithmus werden nur noch die Produkte in die Matrix eingetragen, die von mindestens einem Benutzer gekauft wurden. Da es viele Produkte gibt, die noch von keinem Benutzer gekauft wurden wird die Matrix kleiner als beim vorherigen Algorithmus. Der Algorithmus ist zeitaufwändig: im Worst-Case $O(N^2 \cdot M)$. Dabei bezeichnet N die Anzahl der Produkte im Produktkatalog und M die Anzahl der Benutzer, die vom System verwaltet werden. In der Praxis erhält man allerdings im Mittel eine Laufzeitkomplexität von $O(N \cdot M)$. Der Algorithmus muss nicht zur Laufzeit durchgeführt werden, d. h. während der Laufzeit muss nur noch auf die Matrix bzw. Matrizen zugegriffen werden. Es wird dabei nur auf die Matrizen zugegriffen, die Produkte enthalten, die der Benutzer bereits gekauft hat. In den Matrizen werden dann die Produkte ausgewählt, die den höchsten Ähnlichkeitswert zum Referenzprodukt dieser Matrix besitzen. Da die Berechnung der Matrizen im Vorfeld geschieht und während der Laufzeit nur noch auf die Matrix bzw. Matrizen zugegriffen wird, ist dieses Verfahren äußerst schnell und hängt in seiner Geschwindigkeit nur noch von der Anzahl der Produkte, die der Benutzer bereits gekauft hat, ab.

3.2.5. Vergleich der Verfahren

In den letzten vier Abschnitten wurden verschiedene Verfahren vorgestellt, die alle die Ähnlichkeit zwischen zwei oder mehreren Objekten bestimmen, um so dem Benutzer eine Auswahl von für ihn interessanten Objekten zu liefern.

Das kollaborative Filterverfahren ist zwar auf der einen Seite relativ einfach zu implementieren, allerdings ist es nur für kleine bis mittlere Kataloggrößen ($10^2 - 10^4$ Einträge) sinnvoll. Bei größeren Katalogen wird das Verfahren ineffizient, da für die Ähnlichkeitsberechnung sehr viel Zeit benötigt wird und die Ähnlichkeitssuche ausschließlich zur Laufzeit geschieht.

Eine Leistungssteigerung kann durch Cluster-Verfahren erzielt werden. Bei diesem Verfahren werden ähnliche Objekte nicht zur Laufzeit zu Gruppen zusammengefasst

und durch einen Zentroiden repräsentiert. Allerdings liefern diese Verfahren bei den Ergebnissen nicht die Qualität der kollaborativen Filterverfahren.
Bei den suchbasierenden Verfahren wurde nur eine Exakt-Match-Suche auf den Daten der Datenbank durchgeführt, d. h. aus den vom Benutzer gekauften Produkten wurden beispielsweise Titel, Autor und Produzent extrahiert und eine (SQL-) Anfrage an eine Datenbank gestellt. Dieses Verfahren berücksichtigt allerdings keine Ähnlichkeit im eigentlichen Sinne, sondern nur die logische Oder-Verknüpfung aller Suchworte. Ein weiterer Nachteil dieses Verfahrens ist, dass sehr lange Anfragen entstehen wenn der Benutzer bereits viele (unterschiedliche) Produkte gekauft hat.
Das letzte vorgestellte Verfahren ist das Item-to-Item Collaborative Filtering. Dieses Verfahren erzeugte im Offline-Betrieb mehrere Tabellen bzw. Matrizen, in denen für jedes Produkt eine Ähnlichkeitsbewertung zu allen anderen Produkten durchgeführt wurde. Während der Laufzeit wird dann nur noch auf die Tabellen zugriffen, die sich auf Produkte die vom Benutzer gekauft wurden, beziehen und die Einträge entnommen, die den höchsten Ähnlichkeitswert besitzen. Dieses Verfahren ist zur Laufzeit sehr effizient und liefert eine gute Qualität der Ergebnismenge. Es ist auch weiterhin für sehr große Benutzerzahlen $\left(10^6 - 10^7\right)$ und große Kataloge $\left(\approx 10^6\right)$ geeignet. Dieser Algorithmus findet unter anderem bei Amazon.com Einsatz [LSY03].

3.3. Anwendungsbeispiele

Nachdem wir nun verschiedene Verfahren kennen gelernt haben, die in Web-Intelligent-Systemen eingesetzt werden, wollen wir in diesem Abschnitt auf zwei konkrete Beispiele eingehen. Das erste System ist die Testplattform „myFreddy.com“. Das zweite in 3.3.2. vorgestellte System ist ein japanisches Forschungsprojekt namens „PalmAgent“.

3.3.1. myFreddy.com

Diese Plattform kann als eine Testplattform für Web-Intelligent-Verfahren angesehen werden. Mit Hilfe von Cookies werden die Benutzer eindeutig identifiziert und es entfällt eine Registrierung. Die angebotenen Daten dieser Seite beziehen sich ausschließlich auf lustige Witze, Sprüche und Bilder. Der Benutzer bekommt beim erstmaligen Besuch der Seite eine kleine Kollektion aus Standardbildern oder Witzen präsentiert und kann dem Bild bzw. Witz eine Punktbewertung zwischen 1 und 10 geben (Abbildung 3.4). Durch jede Bewertung erhält er ein weiteres Bild bzw. einen

Witz, den er wieder bewertet (Abbildung 3.5). Das System verwendet diese Bewertungen, um nach dem Verfahren des kollaborativen Filterns diejenigen Benutzer herauszufinden, deren Bewertungen ähnlich zum aktuellen Benutzer sind. Anschließend werden die Bewertungen der ähnlichsten Benutzer analysiert und aus der Kollektion die Objekte ausgewählt, die von anderen Benutzern hoch bewertet wurden und noch nicht vom aktuellen Benutzer bewertet wurden.

Abbildung 3. 4

Abbildung 3. 5

Da große Unternehmen wie Amazon.com oder andere ungern ihre Benutzerdaten nach außen preisgeben, wurde dieses Projekt gegründet, um eigenständig Testdaten über Benutzer zu sammeln und für Auswertungen zur Verfügung zu stellen. Des Weiteren bieten die Macher der Seite auch an, neue Verfahren für Web-Intelligent-Systeme auf ihrer Plattform zu testen bzw. ihre gewonnenen Testdaten weiteren Forschungen zur Verfügung zu stellen [KPZ03].

3.3.2. PalmAgent

Die ATR Media Integration & Communications Research Laboratories in Kyoto haben mit diesem Projekt ein Führungssystem für Touristen entwickelt. Es soll Menschen helfen, sich bei Ausflügen oder im Urlaub besser in fremden Städten zurechtzufinden. Das System funktioniert auf Basis von Personal Digital Assistants (PDAs), die die Personen mit sich führen. Jeder PDA besitzt einen Agenten, der die persönlichen Vorlieben des Besitzers kennt. Durch Funk oder mittels Informationsstationen, die mit Servern verbunden sind, erhält der PDA Informationen über Sehenswürdigkeiten, Veranstaltungen oder sonstige Informationen über die Stadt. Diese Informationen werden vom Agenten nach den Vorlieben des Benutzers bewertet und dem Benutzer auf dem Display angezeigt. Der Benutzer hat die Möglichkeit, den angezeigten Informationen eine Bewertung zu geben. Durch diese

zeigten Informationen eine Bewertung zu geben. Durch diese Bewertungen lernt der Agent ständig hinzu und seine Abschätzung über interessante bzw. uninteressante Informationen für den Benutzer wird besser.
Wird per Funk in der näheren Umgebung ein weiterer PDA-Nutzer erkannt, so beginnen die beiden Agenten ihr Wissen über Veranstaltungen und Informationen auszutauschen, wobei auch Benutzerprofile verglichen werden. Dabei fließen auch die Bewertungen des anderen Benutzers über Veranstaltungen in die Auswertung des Agenten mit ein.
Dieses Projekt steckt zwar noch in den Kinderschuhen, doch vor allem im Bereich von Touristenführung bietet dieses System ganz neue Möglichkeiten [SM01]. Ein ähnliches Projekt findet sich auch bei [Nexus].

4. Prefetching

In den vorangegangenen Kapiteln wurden Verfahren und Methoden aufgezeigt, die mittels Ähnlichkeitsbestimmung die für einen Benutzer interessanten Dokumente automatisch suchen und anzeigen. Dieses Wissen kann man nun auch in Bezug auf Prefetching von Web-Seiten einsetzen, um die Bandbreite von Internetverbindungen vom Server zu den Clients optimal zu benutzen. Durch das Prefetching kann die Latenzzeit vom Aufruf einer Seite durch den Benutzer bis zur vollständigen Darstellung der Seite in dessen Browser entscheidend verkürzt werden. Mit Prefetching bezeichnet man das Vorladen von Daten aus dem Server in einen Client. Ziel ist Wartezeit für den Benutzer zu verkürzen. Die Latenzzeit ist die reale bzw. wahrgenommene Antwortzeit eines Gerätes. In unserem Fall die Zeit die ein Signal oder Anfrage (z. B. einer Internetseite) von einem Punkt im Netzwerk (Client) zu einem anderen Punkt (Server) und wieder zurück benötigt [NetLex].

4.1. Die Idee des Prefetching

Wie schon oben angedeutet ist die Idee beim Prefetching, die Latenzzeit für den Benutzer zu verkürzen. Dazu müssen die Internet-Gewohnheiten des Benutzers bekannt sein, oder mit statistischen Untersuchungen geschätzt werden. Untersuchungen zeigen, dass die meisten Benutzer im Internet den Hyperlinks von einer Web-Seite auf eine andere folgen, welche sich auch meist auf demselben Server befinden. Diese Erkenntnis lässt sich dazu nutzen, dass die Referenzseiten von einer Web-Seite bei einer Anfrage gleich mit an den Benutzer geschickt werden. Eine weitere Möglichkeit um Prefetching zu betreiben ist, die Zeitspanne zwischen den Aufrufen zweier Seiten zu verwenden. Nach [CZ03] ruft ein Benutzer eine Internetseite auf, beginnt den Inhalt der Seite zu lesen und folgt anschließend den Hyperlinks oder ruft eine weitere Seite auf. Dieser Vorgang dauert im Schnitt zwischen 2 und 120 Sekunden und ist ausreichend um ein entsprechendes Prefetching durchzuführen. Bilder sollten nach [CZ03] ebenfalls sofort mit der HTML-Seite vorgeladen werden, da Untersuchungen ergeben haben, dass etwa ein Drittel aller Bilder weniger als 2 Sekunden nach Aufbau der HTML-Seite angefordert wurden. Das Prefetching der Bilder bietet also auch hier einen wesentlichen Optimierungsschritt.

Effizientes Prefetching lässt sich aus diesen Erkenntnissen an drei Bedingungen festmachen [CZ03]:

- Die als nächstes angeforderten Dokumente wurden möglicherweise bereits mit Voraussicht geladen, bevor sie benötigt werden. Dies kann z. B. durch einen Blick in die Browserhistorie geschehen.
- Bei Anforderung stehen aktuelle Prefetching-Daten zur Verfügung.
- Es sollte ein genügend großes Zeitintervall zwischen zwei Seitenaufrufen bestehen, um die Daten vorzuladen.

4.2. Prefetching-Methoden

In der Literatur werden viele Möglichkeiten beschrieben wie Prefetching von Internet-Daten durchgeführt werden kann. In [CZ03] werden 4 Kategorien von Prefetchingansätzen beschrieben:

- Client-basiertes Prefetching
- Proxy-basiertes Prefetching
- Server-basiertes Prefetching
- Kooperatives Prefetching

Die folgenden Abschnitte gehen auf diese Ansätze explizit ein und stellen dabei immer wieder den Bezug zu den Erkenntnissen aus Kapitel 2 und Kapitel 3 her.

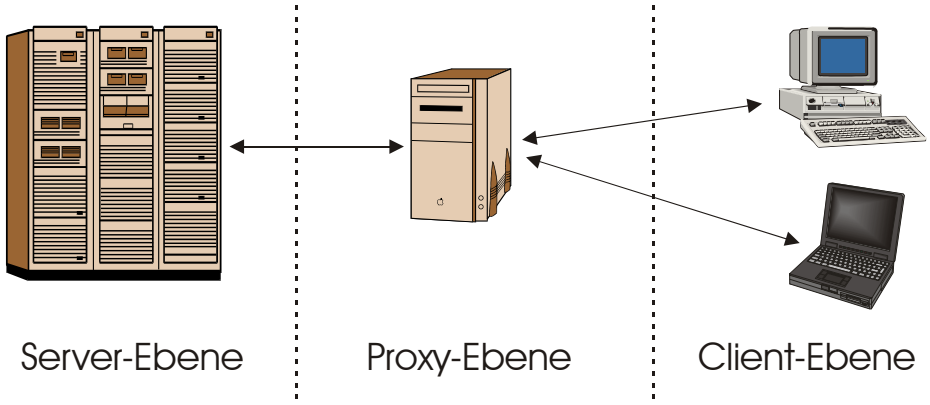

Abbildung 4. 1

4.2.1. Client-basiertes Prefetching

Der einfachste Weg um Prefetching zu betreiben ist, die Historie des Browers auf der Client-Seite zu untersuchen. Diesem Ansatz ist sehr viel Beachtung zu schenken, da aus der Historie Gewohnheiten des Benutzers am besten analysiert werden können. Eine weitere positive Eigenschaft ist, dass die Einträge der Historie sich nicht nur auf Seiten, die auf nur einem Server gespeichert sind, beziehen und dadurch auch ein Server-übergreifendes Prefetching möglich ist.

Die Client-basierten Prefetching-Verfahren lassen sich in Greedy- und Non-Greedy-Verfahren unterteilen. Bei den Greedy-Verfahren kann der Benutzer nur Parameter angeben, beispielsweise die für das Prefetching zur Verfügung stehende Größe der Ressourcen. Diese Art der Verfahren ist sehr einfach zu implementieren, bringt allerdings den Nachteil mit sich, dass sie nicht die Interessen des Benutzer beim Prefetching mit einbeziehen und z. B. alle Hyperlinks einer Seite gleich mit laden. Dadurch werden sehr viele unnötige Daten geladen und es entsteht ein sehr großer Overhead.
Verbesserung bringen da die Non-Greedy Verfahren mit sich. Diese Verfahren werten die Historieeinträge des Benutzers aus und merken sich wie oft eine Seite besucht wurde und bewerten Seiten mit häufigem Zugriff als interessanter als Seiten, die weniger häufig besucht wurden.
In Verbindung mit den in Kapitel 3 vorgestellten Verfahren ist es auch möglich die Non-Greedy Verfahren dahingehend zu erweitern, dass sie die Inhalte der besuchten Seiten auf Ähnlichkeit untersuchen und daraus das Benutzerverhalten noch besser bestimmen zu können, z. B. könnte dies dazu führen, dass eine weniger häufig besuchte Seite auch schon geladen wird, da sie eine häufig besuchten Seite sehr ähnlich ist und somit die Wahrscheinlichkeit hoch ist, dass sie in nächster Zukunft auch vom Benutzer angefordert wird.

4.2.2. Proxy-basiertes Prefetching

Beim Proxy-basierten Prefetching findet das Vorausladen von Seiten auf einem Rechner, der sich im Netz zwischen dem Server und dem Client befindet, statt. Der Vorteil des Proxy-basierten Prefetching liegt darin, dass über einen Proxy die Anfragen von mehreren Clients ablaufen und somit im Proxy das Verhalten mehrerer Clients untersucht werden kann. Wird im Proxy festgestellt, dass auf einer Seite bestimmte Hyperlinks sehr oft und von verschiedenen Benutzern aufgerufen werden, kann der Proxy darauf reagieren und bei der nächsten Anfrage auf diese Seite gleich die entsprechenden Seiten der Hyperlinks an den Benutzer mitschicken, da mit großer Wahrscheinlichkeit davon ausgegangen werden kann, dass dieser Benutzer ein ähnliches Verhalten wie andere Benutzer an den Tag legt.
Weiterhin ist es möglich, dass der Proxy die zwischengespeicherten Daten komprimiert und somit mehr Dateninformationen in kürzerer Zeit zum Client senden kann.

Die zwischengespeicherten Daten verbleiben in der Regel 24 Stunden auf dem Proxy, bevor sie aus Aktualitätsgründen bzw. aus Speicherplatzgründen gelöscht werden.

4.2.3. Server-basiertes Prefetching

Bei diesem Prefetching-Verfahren wird, wie der Name schon sagt, auf Server-Seite das Prefetching durchgeführt. Der Server besitzt eine signifikant größere Historie als der Proxy bzw. der Client. Es ist möglich für alle Seitenzugriffe auf diesem Server das Verhalten der Benutzer zu protokollieren, d. h. herauszufinden welche Seiten zuvor und welche Seiten danach angefordert wurden.
Die Server-basierenden Prefetching Verfahren kann man grob in zwei Klassen unterteilen:

- Push-Strategien: Bei dieser Strategie überträgt der Server (oder auch Proxy) seine am häufigsten angeforderten Seiten gleich mit. Häufig findet sich hier ein Top 10 Prefetching, d. h. die 10 am häufigsten referenzierten Seiten werden übertragen.
- Pull-Strategien: Der Server übergibt hierbei an den Client oder Proxy eine Liste mit Seiten, die entweder häufig referenziert wurden, oder die durch Analyse des Servers demnächst angefordert werden könnten. Diese Liste wird auf Client-Seite untersucht und diejenigen Seiten ausgewählt, die der Server noch übertragen soll. Die Liste wird anschließend wieder an den Server geschickt welcher dann die entsprechenden Seiten bereitstellt.

4.2.4. Kooperatives Prefetching

Die oben beschriebenen Verfahren bieten schon eine bestimmte Effizienzsteigerung, jedoch kann die Effizienz und Effektivität nochmals gesteigert werden, indem man kooperatives Prefetching verwendet. Bei diesem Verfahren findet das Prefetching auf allen Ebenen der Datenübertragung statt, d. h. auf Server-Seite, in den Proxies und auf Client-Seite. Es wird versucht alle Vorteile der vorherigen Verfahren auszunutzen. Zum einen kann man auf der Server-Seite den Proxies die Seiten im Voraus anbieten, die auf diesem Server am häufigsten angefordert werden. Zum anderen können die Proxies die Interessen der Benutzer bestimmen, die auf sie zugreifen und dann entsprechende Anfragen an die Server stellen um Daten vorzuladen. Die

Clients können die sehr genauen Angaben der Benutzerinteressen aus den Historien auslesen und diese vorab an die Proxies weitergeben. Dadurch entsteht eine rege Kommunikation zwischen den verschiedenen Ebenen und es entsteht ein effektives und effizientes Prefetching. Die hieraus resultierende Infrastruktur kann durch ein Web-Intelligent-System erweitert werden, d. h. man fügt auf allen Ebenen (Server, Proxy, Client) Komponenten ein, die die Internetseiten im jeweiligen Speicher miteinander auf Ähnlichkeit vergleichen.

5. Zusammenfassung

Nachdem wir nun verschiedene Verfahren für Web-Intelligent-Systeme betrachtet und Einsatzgebiete für diese Verfahren kennen gelernt haben, möchten wir an dieser Stelle noch einmal kurz die Erkenntnisse zusammenfassen.

Als Einstieg in das Thema haben wir in Kapitel 2 zuerst einmal eine Begriffsklärung vorgenommen um die Begriffe Web Intelligence und personalisiertes Web besser verständlich zu machen. Weiterhin haben wir in diesem Kapitel Infrastrukturen betrachtet, die für einen effizienten Einsatz von Web-Intelligent-Systemen Vorraussetzung sind.

Kapitel 3 beschäftigte sich dann explizit mit vier verschiedenen Ansätzen (Kollaborative Filter, Cluster Verfahren, Suchbasierende Verfahren, Item-to-Item Collaborative Filtering) zur Ähnlichkeitsanalyse zwischen Dokumenten bzw. Objekten im Allgemeinen. Dabei haben wir festgestellt, dass die ersten drei Ansätze sich nur effizient für kleine bzw. mittlere Datengrößen eignen und der vierte Ansatz auch in großen Datenbeständen noch effizient arbeitet und gute, bis sehr gute Resultate liefert. Weiterhin wurden in diesem Kapitel zwei Anwendungsbeispiele vorgestellt. Zum einen die Testplattform myFreddy.com, eine Plattform auf der neue Verfahren zur Ähnlichkeitsanalyse getestet werden können und deren Daten für Forschungszwecke zur Verfügung gestellt werden. Das zweite Beispiel behandelte ein Führungssystem für Touristen mit Hilfe von PDAs. Das System wählte nach den Vorlieben seines Benutzers und den Informationen, die es von außen durch Funkdaten empfangen hat, diejenigen Sehenswürdigkeiten bzw. Veranstaltungen aus, die am Besten zu den Benutzerinteressen passten.

Im letzten Kapitel wurde noch eine weitere Einsatzmöglichkeit für Verfahren zur Ähnlichkeitsbestimmung vorgestellt, das Prefetching. Wir haben das Prefetching auf verschiedenen Ebenen (Client-, Proxy- und Server-basiertes Prefetching) betrachtet und die Vor- und Nachteile der einzelnen Prefetching-Strategien herausgestellt. Unsere Ergebnisse führten uns zu der Erkenntnis, dass eine Kombination der verschiedenen Strategien, also ein Prefetching auf allen Ebenen im Mittel, zur besten Lösung führt.

Referenzen

BS01 Bleich, Holger; Schüler, Peter:
„Digitale Fußspuren“
in c’t Magazin für Computer Technik;
Verlag: Heise
S. 200 – 209; 8/2001

CZ03 Chen, Xin; Zhang Xiaodong:
„Web Document Prefetching on the Internet“
in Web Intelligence von Zhong; Liu; Yao
Verlag: Springer
S. 345 – 363; 2003

Heyl01 Hylighen, Francis:
„Collaborative Filtering“;
http://pespmc1.vub.ac.be/COLLFILT.html
2001

INet02 Wichtige Aspekte der E-Business-Infrastruktur:
http://www.sap.info/de/go/18573
2002

INet03 Personalisierte Portale:
http://h40047.www4.hp.com/solution/crm/telko-pportale.html
2003

KPZ03 Kalles, Dimitrios; Papagelis, Athanasios; Zaroliagis, Christos:
„Algorithmic Aspects of Web-Intelligent-Systems“;
in Web Intelligence von Zhong; Liu; Yao
Verlag: Springer
S. 323 – 344; 2003

NetLex Netlexikon:
http://www.net-lexikon.de
2003

LSY03 Linden, Greg; Smith, Brent; York Jeremy:
„Amazon.com Recommendations: Item-to-Item Collaborative Filtering“;
http://dsonline.computer.org/0301/d/wp1lind.htm
2003

Nexus Prof. Dr. Ing. Dr. h. c. Andreas Reuter
http://www.nexus.uni-stuttgart.de/
http://www.i-u.de/departments/admin.htm

Rijs99 van Rijsbergen, Keith:
„Information Retrieval Second Edition“;
http://www.dcs.gla.ac.uk/~iain/keith/index.htm
1999

SM01 Sumi, Yasuyuki; Mase, Kenji:
„Collecting, Visualizing and Exchanging Personal Interests and Experiences in Communities“;
in Web Intelligence, First Asia Pacific Conference ; proceedings von Zhong
Verlag: Springer
S. 163 – 174; 2001

Tan03 Tan, Ah-Hwee:
„Personalized Information Management for Web Intelligence“;
http://suez.cs.gsu.edu/~cscyqz/research/wcci2002-cwi/Tan.pdf
2003

Vinc01 Vincenti, Stefano:
„Web Intelligence: Beyond the Hype“;
http://www.novonordisk-it.com/primary+channels/press/news+archive/web-intelligence.htm
2001