# Data-Warehouse-basierte Architektur für adaptive Online-Recommendations

Andreas Thor, Erhard Rahm

Universität Leipzig, Institut für Informatik, Abteilung Datenbanken
Augustusplatz 10-11, 04109 Leipzig, Germany
E-mail: {thor | rahm}@informatik.uni-leipzig.de

## Kurzfassung

Aufgezeichnetes Nutzungsverhalten von Websites stellt Wissen bereit, wie Nutzer innerhalb der Website navigieren. Mittels effizientem Wissensmanagement können die Nutzer bei der Navigation z.B. durch den Einsatz von Online-Recommendation-Systemen unterstützt werden. Die vorgestellte Arbeit zeigt eine Architektur, wie ein solches Recommendation-System flexibel und adaptiv gestaltet werden kann.

## 1 Einleitung

Kundenkarten wie z.B. Payback zeigen das Interesse des Einzelhandels, möglichst viele Informationen zum Kaufverhalten der Kunden zu generieren. Entsprechend versuchen die Betreiber von e-Shops, das Nutzungsverhalten innerhalb ihrer Websites zu analysieren, um Kundenbedürfnisse zu ermitteln und zu befriedigen. Aber auch für nicht-kommerzielle Webauftritte ist es interessant, die Navigation der Nutzer zu verstehen und zu unterstützen, um eine große Nutzerbindung zu erreichen.

Eine wichtige Anwendung des so erzeugten Wissens stellen Online-Recommendation-Systeme dar [11]. Der Online-Buchhandel Amazon ist dafür ein prominentes Beispiel. Hier werden dem Kunden zu einem ausgewählten Produkt Empfehlungen präsentiert, die ihm die Navigation im umfangreichen Produktsortiment erleichtern sollen. Dabei verwendet Amazon mehrere Empfehlungsverfahren bzw. -arten, z.B. Lieblingslisten anderer Kunden oder Produktpaare, die häufig gemeinsam gekauft wurden [6]. Die Empfehlungen der einzelnen Verfahren oder Recommender sind automatisch bestimmt, jedoch ist die Auswahl der eingesetzten Recommender statisch festgelegt. Die in zahlreichen E-Commerce-Websites verwendeten Applikations-Server unterstützen mittlerweile häufig die Erzeugung von Online-Recommendations. Allerdings sind bei Ansätzen wie IBM Websphere Personalization [3] nicht nur die anzuwendenden Recommender, sondern auch die von ihnen gelieferten Empfehlungen oft manuell (statisch) festgelegt. Das Navigationsverhalten eines Nutzers auf der Website bleibt in den kommerziell eingesetzten Recommendation-Ansätzen derzeit meist unberücksichtigt.

Die bisherigen Erfahrungen zeigen die Notwendigkeit verschiedene Arten von Recommendations zu unterstützen. Weiterhin ist gerade für größere Websites mit sehr vielen Empfehlungsmöglichkeiten (zahlreiche Produkte, Dokumente etc.) die automatische Bestimmung der Recommendations essentiell, wozu eine Vielzahl von Verfahren nutzbar ist. Eine weitgehend ungeklärte Fragestellung ist jedoch, welche Art von Recommendations wann und für welchen Nutzer am effektivsten ist und daher eingesetzt werden sollte. Dies festzustellen erfordert eine Evaluierung der Effektivität unterschiedlicher Empfehlungsarten, was jedoch selbst bereits ein schwieriges Problem darstellt. In [2] wurden hierfür Testpersonen überwacht, die vorgegebene Aufgaben innerhalb der Website mit und ohne Recommendations lösen sollen. Dieser Ansatz ist sehr aufwändig, i.a. nicht repräsentativ und für Produktempfehlungen u.ä. weniger geeignet. Eine technische Schwierigkeit bei der automatisierten Bewertung von als gewöhnliche Links realisierten Recommendations ist die Unterscheidung deren Verwendung von einer normalen Navigation. Weiterhin ist den üblichen Aufzeichnungen von

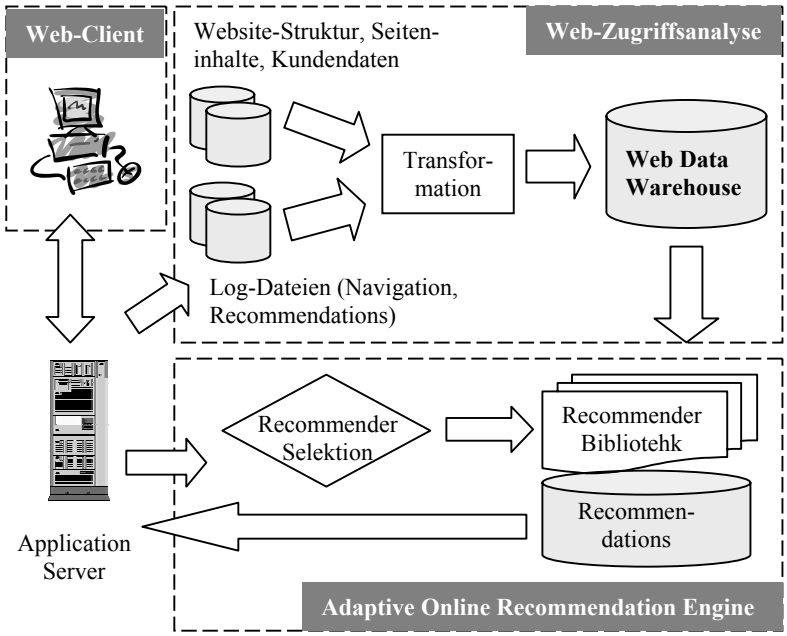


Abbildung 1: Architektur für adaptive Online Recommendations

Web-Zugriffen (Web Log) nicht zu entnehmen, wann welche (nicht akzeptierten) Recommendations angezeigt wurden.

Ziel unserer Forschungsarbeiten ist die Konzeption, Entwicklung und Evaluierung von leistungsfähigen Online-Recommendation-Systemen, welche eine Vielzahl von Empfehlungsarten (Recommender) und eine automatische Bestimmung von Empfehlungen unterstützen. Dabei soll erstmals die Auswahl der Recommender automatisiert und dynamisch erfolgen in Abhängigkeit des Wissens über den aktuellen Nutzer, der aktuellen Website-Inhalte und anderen Faktoren. Die Auswahl der Recommender soll adaptiv in Reaktion auf die Effektivität früherer Recommendations erfolgen. Hierzu soll eine automatisierte Bewertung der Verwendung von Empfehlungen und Recommender erfolgen und die Ergebnisse mit Verfahren des Machine Learning zur dynamischen Recommender-Auswahl genutzt werden.

Der vorgestellte Ansatz soll auch eine Möglichkeit zur Überwindung typischer Probleme im Bereich von Online-Recommendation-Systemen aufzeigen. Systeme, die Empfehlungen auf der Basis von Navigationsmustern berechnen, sind mit der Tatsache konfrontiert, dass neu in die Website eingebrachte Seiten zunächst in keinem Navigationsmuster auftreten (können). Damit kann der neue Inhalt zunächst nicht als Empfehlung präsentiert werden (*New Content Problem*). Ein üblicher Ansatz zur Lösung dieses Problems ist die Integration von Seiteninhalten in die Navigationsmuster [8], weitergehende Ansätze benutzen Ontologien [2]. Ebenfalls problematisch ist die *Selbstverstärkung* von empfohlenem Inhalt. Durch das Anzeigen einer Recommendation erhöht sich die Wahrscheinlichkeit, dass die entsprechende Seite angefordert wird. Damit tritt diese Seite verstärkt in Navigationsmustern auf, wodurch wiederum die Wahrscheinlichkeit steigt, dass diese Seite als Recommendation ermittelt wird. Dieses Verhalten kann u.U. erwünscht sein (z.B. für ein gewinnbringendes Produkt innerhalb eines e-Shops), kann jedoch auch die Qualität der

Recommendations verschlechtern, da bessere Recommendations zunehmend verdrängt werden.
Der Beitrag ist wie folgt aufgebaut: Abschnitt 2 stellt die entwickelte Data-Warehouse-basierte Architektur überblicksartig vor. Anschließend wird detailliert auf das eingesetzte Data Warehouse (Kapitel 3) und die Recommendation Engine (Kapitel 4) eingegangen. Abschließend wird eine kurze Zusammenfassung gegeben.

## 2 Architektur

Abbildung 1 zeigt unsere Architektur zur Erstellung und zum Einsatz adaptiver Online-Recommendations. Wesentlich sind drei eng interagierende Komponenten: der Applikations-Server, das Web Data Warehouse zur Web-Zugriffsanalyse sowie die Recommendation Engine. Der Applikations-Server ist für die Umsetzung der Website zuständig und realisiert die Schnittstelle zum Client. Er erweitert den Web-Server um die Fähigkeit, Webseiten dynamisch zu erstellen und nimmt eigene Aufzeichnungen der Web-Zugriffe vor. In der derzeitigen Prototyp-Realisierung werden PHP-Skripte zur Steuerungs- und Präsentationslogik verwendet.
Die dargestellte Web-Zugriffsanalyse ist eine Erweiterung des in [12][7] vorgestellten Data-Warehouse-Einsatzes. Der Data-Warehouse-Ansatz wurde gewählt, damit eine zentrale Datengrundlage für die Web-Zugriffsanalyse und für alle Recommender zur Verfügung steht. Das Warehouse integriert dabei das aus den Web-Zugriffen (Web-Logs) ermittelte Navigationsverhalten (inklusive einer Wiedererkennung von Nutzern) und die Beschreibung von Inhalt und Struktur der Website. Eine wesentliche Erweiterung gegenüber bisherigen Arbeiten ist die Verwendung des Warehouse für die automatische Evaluation der Recommendation-Nutzung. Die präsentierten, aber nicht akzeptierten Recommendations werden in einer gesonderten Log-Datei aufgezeichnet und ebenfalls in das Warehouse eingebracht.
Die Recommendation Engine enthält eine modular erweiterbare Bibliothek mit verschiedenen Recommendern. Dabei handelt es sich um eigenständige Algorithmen, die für einen aktuellen Nutzer und die angeforderte Seite unter Einbeziehung der Daten des Data Warehouse entsprechende Recommendations erzeugen. Einfache Recommender empfehlen z.B. die am häufigsten angeforderten Seiten, die aktuellsten Inhalte oder die Ausstiegsseite der letzten Sitzung bei einem wiederkehrenden Nutzer. Weiterhin sind komplexere Recommender vorgesehen, die sich Data-Mining-Techniken (z.B. Assoziationsregeln) bedienen und/oder weitergehender Informationen zum Nutzer, den Inhalten und des Navigationsverhaltens nutzen (z.B. Empfehlung ähnlicher Produkte oder von Favoriten von Nutzern mit ähnlichen Interessen). Um eine schnelle Verwendung der Recommendations zu gewährleisten, werden diese durch die Recommender weitgehend periodisch vorberechnet und dauerhaft gespeichert.
Von besonderer Bedeutung in der Recommendation Engine ist die Selektionskomponente zur dynamischen Auswahl der Recommender für einen konkreten beim Applikations-Server eingegangenen Request. Im Rahmen des Prototyps sollen mehrere Selektionsstrategien unterstützt und verglichen werden, insbesondere zufällige, manuelle und adaptive Strategien. Die zufällige Auswahl dient als Vergleichsverfahren zur Evaluation. Bei der manuellen Auswahl wird durch manuell zu erstellende Regeln festgelegt, welcher Recommender in Abhängigkeit vom aktuellen Nutzer und seinem Request ausgewählt wird. Besondere Bedeutung kommt den adaptiven Strategien zu. Diese nutzen die im Data Warehouse vorliegenden Zugriffsdaten und bewerten für einen Nutzer (oder eine Nutzergruppe), welche Recommender besonders häufig vom Nutzer ausgewählte Recommendations produzierten. Damit können bei der Auswahl die in der Vergangenheit für den Nutzer effektivsten Recommender verwendet werden und diese Entscheidung kann sich selbsttätig an die Bedürfnisse der Nutzer anpassen. Zur Realisierung dieser Aufgabe sollen Verfahren des maschinellen Lernens eingesetzt werden.

## 3 Data Warehouse

### 3.1 Datenquellen

Die wichtigste Datenquelle für das Data Warehouse ist das aufgezeichnete Nutzungsverhalten der Website-Besucher. Üblicherweise wird dazu die vom Web-Server automatisch erzeugte Log-Datei verwendet. Dabei wird jeder Zugriff auf die Website in einem Log-Satz gespeichert, welches i.A. im Common Log Format (CLF) dargestellt wird. Neben dem Zeitpunkt des Requests und der angeforderten Datei werden u.a. auch die Hostadresse des Nutzers und der Referrer gespeichert. Letzterer bezeichnet die URL der Seite, von der aus der Request abgesetzt wurde. Ist diese URL insbesondere eine Suchmaschine (z.B. Google), so sind dem Referrer i.A. auch die vom Nutzer eingegebenen Suchbegriffe zu entnehmen.
Die Verwendung einer solchen Log-Datei verlangt eine aufwändige Aufbereitung der Daten [1]. Da in der Log-Datei die Requests aller angeforderten Dateien aufgezeichnet sind, muss zunächst eine Pageview[1]-Identifikation durchgeführt werden. Dazu müssen u.a. sämtliche Bilder oder Javascript-Dateien eliminiert werden. Weiterhin ist eine Sitzungsidentifikation erforderlich. Dabei werden diejenigen Pageviews zusammengefasst, die innerhalb einer Sitzung für einen Nutzer aufgetreten sind. Zur Bestimmung der einzelnen Sitzungen wird davon ausgegangen, dass die Hostadresse des Nutzers und der Name des verwendeten Browsers innerhalb eines definierten Zeitraums (z.B. 30 Minuten) gemeinsam einen Identifier bilden. Da diese Annahme nicht immer korrekt ist (z.B. wenn der Nutzer einen Proxy-Server benutzt), ist die Sitzungsidentifikation fehlerbehaftet. Zusätzlich kann die Referrer-Information genutzt werden, um verschiedene Nutzer mit gleichem Identifier zu unterscheiden.
Zur Reduzierung der Nachteile wird innerhalb der vorgestellten Architektur eine vom Application Server generierte Log-Datei verwendet. Sie wird innerhalb der Website durch PHP-Skripte erzeugt, die für jeden Pageview genau einmal ausgeführt werden. Damit enthält die so erstellte Log-Datei für jeden Pageview nur einen Log-Satz, so dass eine Pageview-Identifikation nicht mehr notwendig ist.
Die Log-Datei erweitert das CLF um einen Session- und einen User Identifier. Zur Speicherung dieser beiden Werte wird je ein Cookie eingesetzt. Dabei handelt es sich um kleine Textdateien, die auf dem Rechner des Nutzers abgelegt werden. Der Session Identifier dient zur Sitzungsidentifikation, weshalb das zugehörige Cookie bei Beendigung der Browser-Nutzung gelöscht wird (Session Cookie). Um eine Nutzerwiedererkennung zu realisieren, ist eine dauerhafte Speicherung des User Identifiers innerhalb eines persistenten Cookies nötig.
Aus Sicherheitsgründen ist es möglich, dass der Nutzer keine Cookies zulässt. Dann ist eine allgemein zuverlässige Nutzerwiedererkennung nicht möglich, da sich nur noch Nutzer mit eindeutig zugeordneter (fester) IP-Adresse erkennen lassen. Werden auch Session Cookies abgewiesen, bleibt zur Sitzungsidentifikation nur das vorher dargestellte heuristische Verfahren. Eine Analyse des Nutzungsverhaltens auf http://dbs.uni-leipzig.de ergab jedoch, dass ca. 85% der Nutzer Session Cookies zulassen. Die eingesetzte Cookie-Technik vereinfacht und verbessert daher die Sitzungsidentifikation in den meisten Fällen.
Als zweite Datenquelle des Data Warehouse dient die Website selbst. Für ein effektives Knowledge Management ist es notwendig, die Inhalte der Website einheitlich zu beschreiben. Hierfür werden zunächst einfache Ontologien in Form hierarchischer Kategorisierungen der Seiten erstellt. Diese Einteilung kann nach mehreren Gesichtspunkten geschehen. Zusätzlich werden alle Seiten mit weiteren Daten annotiert, die dann für die spätere Berechnung der Recommendations genutzt werden können. Beispiele hierfür sind z.B. die Bestimmung relevanter Schlüsselwörter des Seiteninhaltes sowie die Unterscheidung zwischen einer normalen und einer empfohlenen Seite, d.h. einem Pageview, der durch eine präsentierte Recommendation entstanden ist

[1] Als Pageview bezeichnet man eine vollständig angeforderte, im Browser dargestellte Seite. Diese ist i.A. aus mehreren Dateien (z.B. Bilder, HTML- und Javascript-Dateien) zusammengesetzt.

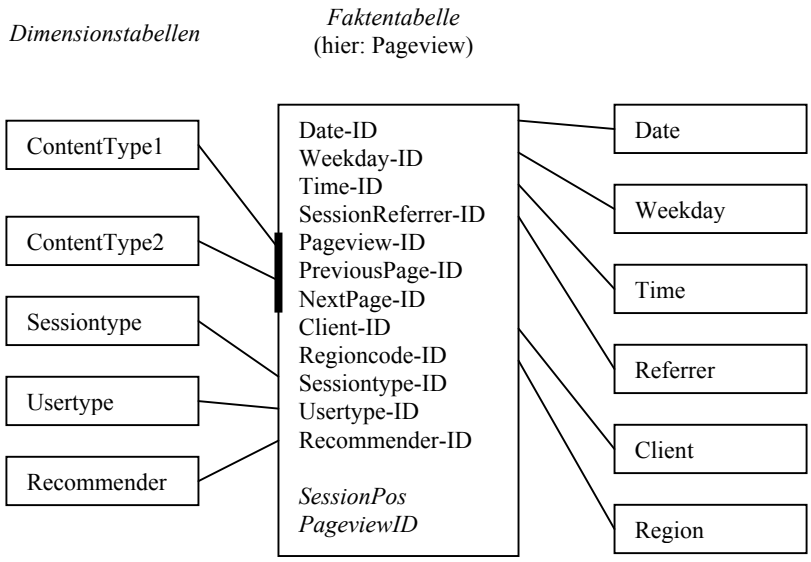


Abbildung 2: Sternschema der Faktentabelle Pageview

### 3.2 Data-Warehouse-Schema

Die zur Verfügung stehenden Daten werden periodisch in ein Data Warehouse importiert. Die Daten werden dabei primär in Faktentabellen abgelegt. Zur Beschreibung der Eigenschaften der Fakten werden Dimensionstabellen genutzt, die über Fremdschlüsselbeziehungen mit der Faktentabelle verbunden sind. Die Daten aus den beschriebenen Datenquellen werden innerhalb der Dimensionstabellen strukturiert abgelegt. Abbildung 2 zeigt exemplarisch das Stern-Schema für die Speicherung der aufgezeichneten Pageviews. Dabei werden z.B. die bereits angesprochenen Kategorisierungen des Inhaltes in den Dimensionen „ContentType“ abgebildet. Zusätzlich wird für jeden Pageview durch die Dimension „Recommender“ festgehalten, ob es sich um eine Navigation auf Grund einer Empfehlung handelt und (falls ja) welcher Recommender zur Berechnung genutzt wurde. Einfache Berechnungen, wie z.B. die Bestimmung des Wochentages aus dem Datum, können ebenfalls beim Einbringen der Daten in das Data Warehouse durchgeführt werden.

Mit Hilfe der Nutzerwiedererkennung kann jedem Nutzer auch die Anzahl seiner Sitzungen zugeordnet werden. Diese Werte können nun gruppiert werden, so dass z.B. jeder Nutzer in die Kategorie „Anfänger“ (nur eine Sitzung) oder „Erfahrener Nutzer“ (mehr als eine Sitzung) eingeordnet werden kann (Dimension „UserType“). Diese Typisierung der Daten wird im Data Warehouse durch verschiedene Hierarchiestufen innerhalb der Dimensionen unterstützt. Des Weiteren lassen sich u.a. die einzelnen Sitzungen bezüglich ihrer Länge (d.h. der Anzahl der Pageviews) unterteilen (Dimension „SessionType“).

Durch den i.A. hierarchischen Aufbau der Dimensionen sind nun für jeden Pageview alle relevanten Informationen in verschiedenen Detailstufen verfügbar. Damit sind u.a. auch Data-Warehouse-gestützte Auswertungsverfahren anwendbar. Ein Beispiel hierfür sind mehrdimensionale OLAP-Analysen (Online Analytical Processing). Dabei lassen sich durch die Benutzung verschiedener Auswahlkriterien auf verschiedenen Detailstufen komplexe Auswertungen zum Nutzungsverhalten durchführen [12]. Innerhalb dieses Forschungsprojektes werden diese Daten zur Berechnung von Recommendations genutzt.

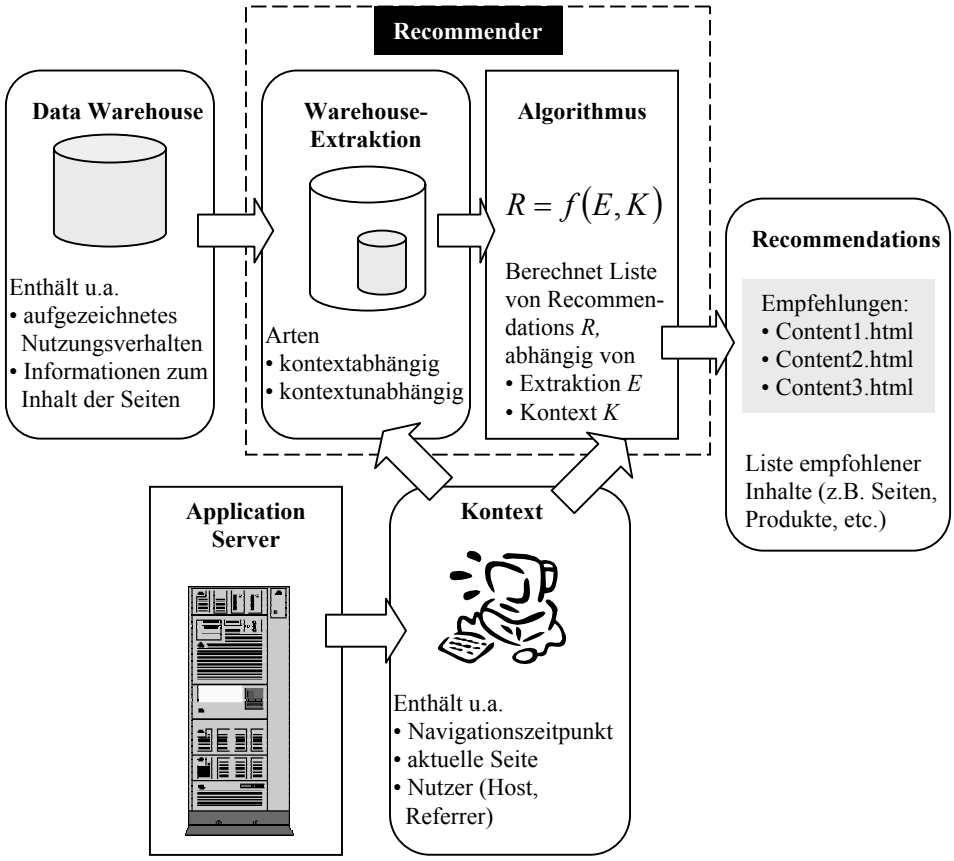


Abbildung 3: Aufbau und Funktionsweise eines Recommenders (schematisch)

## 4 Recommendation Engine

### 4.1 Recommender-Bibliothek

Die Recommender-Bibliothek umfasst die Definition und Beschreibung aller zur Verfügung stehenden Recommender. Abbildung 3 zeigt den schematischen Aufbau der eingesetzten Recommender.

Die Aufgabe eines Recommenders besteht darin, für eine angeforderte Seite eine Liste von Empfehlungen (d.h. anderer Seiten innerhalb der Website) zu berechnen, welche dann dynamisch innerhalb der Seite dargestellt werden können. Ausgangspunkt für diese Berechnung ist der Application Server, welcher zunächst den aktuellen *Kontext* liefert. Dieser umfasst Informationen, die zum Navigationszeitpunkt zur Verfügung stehen, d.h.

- Nutzer (Host-Adresse, User Identifier)
- Referrer-Information
- Datum und Uhrzeit
- aktuelle Seite
- aktuelle Sitzung (Session Identifier)

Dieser Kontext reicht jedoch i.A. nicht für eine Berechnung von Recommendations aus. Daher werden weitere Teile des Data Warehouse hinzugezogen, die durch eine *Warehouse-Extraktion* bestimmt werden. Diese Extraktion kann grundsätzlich sowohl kontextabhängig als auch kontextunabhängig erfolgen. Kontextabhängige Extraktionen sind z.B. das Nutzungsverhalten der

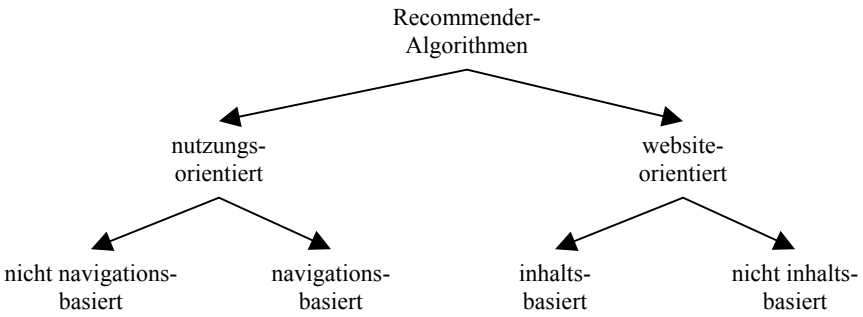


Abbildung 4: Grobklassifizierung von Recommendern unter Berücksichtigung der genutzten Daten

letzten sieben Tage (abhängig vom Zeitpunkt) oder alle bisher aufgezeichneten Sitzungen des aktuellen Nutzers. Bei kontextunabhängigen Extraktionen besteht kein Bezug zum aktuellen Kontext. So ist z.B. das Nutzungsverhalten aller Nutzer, die bereits mindestens drei Sitzungen innerhalb der Website absolviert haben, eine kontextunabhängige Extraktion. Innerhalb einer Extraktion können auch verschiedene Extraktionsarten miteinander kombiniert werden.

Neben Kontext und Warehouse-Extraktion spielt der anzuwendende *Algorithmus* eine entscheidende Rolle. Abbildung 4 zeigt eine hinsichtlich der für den Algorithmus relevanten Daten hierarchische Gliederung möglicher Verfahren[2]. Man kann grundsätzlich zwei Arten unterscheiden: nutzungs- und website-orientierte Algorithmen. Erstere verwenden zur Berechnung von Recommendations das bisher aufgezeichnete Nutzungsverhalten der Website. Demgegenüber nutzen website-orientierte Algorithmen ausschließlich Aufbau, Inhalt und Struktur der Website.

Nutzungsorientierte Algorithmen lassen sich in zwei Untergruppen zerteilen. Navigationsbasiertes Verfahren verwenden die in der Log-Datei aufgezeichneten Navigationspfade. Daraus lassen sich z.B. Navigationsmuster erstellen, die dann in die Berechnung der Recommendations eingehen. Ein Algorithmus, der diejenigen Seiten empfiehlt, die am häufigsten mit der aktuellen Seite innerhalb einer Sitzung zusammen aufgetreten sind, ist ein Beispiel für einen navigationsbasiertes Verfahren. Hingegen betrachten nicht navigationsbasierte Verfahren die einzelnen angeforderten Seiten isoliert. Der konkrete Navigationspfad (z.B. wie man auf die Seite gelangt ist) spielt dabei keine Rolle. Ein Beispiel dafür ist ein Algorithmus, der die Anzahl der Pageviews für jede Seite zählt und die am häufigsten angeforderten Seiten empfiehlt.

Die zweite große Gruppe der Recommender-Algorithmen sind website-orientierte Algorithmen. Sie vernachlässigen das aufgezeichnete Nutzungsverhalten und verwenden lediglich Informationen der Website, um ihre Berechnung durchzuführen. Hier lassen sich inhaltsbasierte und nicht-inhaltsbasierte Verfahren unterscheiden. Ein typischer Vertreter der erstgenannten Verfahrensklasse ist ein Algorithmus, welcher zur aktuellen Seite die ähnlichsten Seiten an Hand eines Ähnlichkeitsmaßes bestimmt. Ein anderer inhaltsbasierter Algorithmus kann z.B. angewendet werden, wenn es sich um einen Nutzer handelt, der von einer Suchmaschine (z.B. Google) auf die entsprechende Website gelangt ist. Dann können die dort verwendeten Suchbegriffe im Referrer ausgelesen werden und diese Information nun genutzt werden, um zu den Suchbegriffen passende Seiten zu empfehlen.

Nicht-inhaltsbasierte Verfahren nutzen nun weder das Nutzungsverhalten noch den Inhalt der einzelnen Webseiten. Ein solcher Algorithmus ist z.B. die Empfehlung der neuesten Seiten innerhalb der Website. Ebenfalls in diese Gruppe fallen Berechnungsverfahren, die die Empfehlungen zufällig bestimmen.

[2] Weitere mögliche Klassifizierungen von Recommendern sind in [5] dargestellt.

### 4.2 Recommender-Selektion

Der präsentierte modulare Aufbau von Recommendern führt zu einer Vielzahl von konkreten Berechnungsverfahren. Am Beispiel des Algorithmus, der aus einer Menge von Seiten (allgemein: Inhalten) diejenigen auswählt, die am häufigsten im Nutzungsverhalten aufgetreten sind, soll die Vielfalt kurz skizziert werden.

Zunächst wird eine Warehouse-Extraktion durchgeführt, die sich aus mehreren Teilextraktionen zusammensetzt. Diese definieren jeweils eine Bedingung an die zu extrahierenden Daten und können zeit-, nutzer- und inhaltsorientiert sein. So ist z.B. eine kontextabhängige Extraktion möglich, die den betrachteten Nutzungszeitraum einschränkt (z.B. die letzten sieben Tage oder die letzten vier Wochen). Diese kann mit einer kontextunabhängigen Extraktion kombiniert werden, die nur diejenigen Nutzer auswählt, die bereits mehr als eine bestimmte Anzahl von Sitzungen (z.B. drei) absolviert haben. Schließlich werden mit einer inhaltlichen Extraktion nur noch diejenigen Seiten betrachtet, die eine gleiche inhaltliche Kategorisierung wie die aktuelle Seite besitzen. Da es verschiedene Kategorisierungen gibt, die dann selbst wieder hierarchisch angeordnet sind, stehen auch hier eine Vielzahl von Gruppen (z.B. Seiten der selben Hauptkategorie) zur Verfügung.

Neben der dargestellten Vielfalt für eine Warehouse-Extraktion stehen auch noch verschiedene Algorithmen zur Verfügung. Es ergibt sich somit allgemein die Problematik, dass für einen konkreten Kontext ein oder mehrere Recommender ausgewählt werden müssen. Ein Verfahren, welches diese Auswahl für jeden Kontext vornimmt, wird als *Selektionsstragie* bezeichnet. Der Schwerpunkt unseres Forschungsvorhabens liegt dabei in der Entwicklung einer Selektionskomponente zur automatischen Bestimmung geeigneter Recommender.

Grundsätzlich muss dabei zunächst definiert werden, was das Ziel der präsentierten Recommendations ist. Im kommerziellen Umfeld (z.B. e-Shop) ist dies natürlich die Gewinnmaximierung. Andere Website-Betreiber möchten die Nutzer möglichst lange auf ihrer Website halten, um dadurch eine größere Nutzerbindung zu erreichen. Zunächst gehen wir jedoch von einem domänenunabhängigen Ziel aus, d.h. sämtliche Strategien sollen die Anzahl der akzeptierten (d.h. angeklickten) Recommendations maximieren. Um somit die Qualtät einer Recommendation zu bestimmen, wird bei jeder Präsentation von Recommendations eines Recommenders gespeichert, ob eine der empfohlenen Seiten angeklickt wurde (positives Feedback) oder nicht (negatives Feedback).

Bei den Selektionsstrategien lassen sich feedback-basierte (d.h. adaptive) und nicht-feedback-basierte Verfahren unterscheiden. Ein Beispiel für die zweite Verfahrensklasse ist die zufällige Auswahl von Recommendern. Sie dient später als Vergleichsstragie bei einer späteren Evaluation. Adaptive Verfahren benutzen die ermittelte Qualtiät der bisher eingesetzten Recommender. Einfache Strategien können z.B. für jeden Pageview denjenigen Recommender auswählen, der für diese Seite das beste Feedback erhalten hat.

Zur Realisierung komplexer adaptiver Selektionsstragien soll u.a. auch das Prinzip des Collaborative Filterings genutzt werden. Dieser Ansatz wird bisher u.a. zur Bestimmung einzelner Recommendations verwendet. Dabei werdem dem aktuellen Nutzer diejenigen Nutzer zugeordnet, welche ein ähnliches Navigations- oder Kaufverhalten besitzen. Im Rahmen der Selektionsstrategie werden stattdessen Nutzer gesucht, die für die eingesetzten Recommender ein ähnliches Feedback gegeben haben. Diese Nutzergruppe dient als Grundlage zur Auswahl der Recommender für den aktuellen Kontext.

Der vorgestellte Ansatz soll sowohl das New-Content-Problem als auch die Selbstverstärkung von empfohlenem Inhalt umgehen. Grundsätzlich tritt das New-Content-Problem zunächst nur bei nutzungsorientierten Recommendern auf. Des Weiteren hat die Warehouse-Extraktion einen großen Einfluss, insbesondere die Auswahl des Zeitfensters (z.B. die Website-Nutzung der letzten 24 Stunden), in welchem das Nutzungsverhalten betrachtet wird. Die Adaptivität der Selektionsstrategie muss nun gewährleisten, dass für Nutzer, die häufig neue Inhalte anfordern, entsprechend nutzungsorientierte Recommender mit kurzen Zeitfenstern oder website-orientierte Recommender in den Vordergrund treten.

Zur Vermeidung von Selbstverstärkung geschieht die Anwendung einer Selektionsstrategie auf der Basis von Recommendern. Ebenfalls wird das protokollierte Feedback den Recommendern und nicht den einzelnen Recommendations zugeordnet. Der Verstärkungs- bzw. Lerneffekt innerhalb der Selektionsstrategie befindet sich daher auch auf Recommender-Ebene. Da ein Recommender zu verschiedenen Kontexten verschiedene Recommendations liefern kann, bleibt ein Selbstverstärkungseffekt von einzelnen konkreten Inhalten aus. Des weiteren kann im Data Warehouse zwischen normalen und empfohlenen Pageviews unterschieden werden. Die verstärkte Anforderung einzelner Seiten auf Grund von Recommendations kann dadurch für die Berechnung weiterer Empfehlungen herausgefiltert werden.

## Zusammenfassung

Es wurde eine Architektur zur flexiblen Realisierung von Online-Recommendations unter Nutzung von Informationen über das Zugriffsverhalten von Website-Nutzern vorgestellt. Als wesentliche Neuerung gegenüber bisherigen Ansätzen soll damit die Auswahl der eingesetzten Empfehlungsarten dynamisch erfolgen. Durch die automatisierte Bewertung der Recommendation-Nutzung auf Basis eines Data Warehouse wird eine adaptive Verfahrensweise angestrebt. Die prototypische Umsetzung des Ansatzes wurde bereits begonnen.

## Literatur


[1] Cooley, R., Mobasher, B., and Srivastava, J.: *Data preparation for mining world wide web browsing*. Journal of Knowledge Information Systems, 1(1), 5-32, 1999

[2] Dai, H., Mobasher, B.: *Using ontologies to discover domain-level web usage profiles.* Proc. of the 2nd Semantic Web Mining Workshop at PKDD, 2002

[3] *Documentation for IBM WebSphere Personalization*, http://www.ibm.com/software/webservers/personalization/library.html

[4] Heer, J., Chi, E.H.: Separating the Swarm: *Categorization Methods for User Sessions on the Web*. Proc. of Conference on Human Factors in Computing Systems, 2002

[5] Jameson, A., Konstan, J., Riedl, J.: *AI Techniques for Personalized Recommendation*. Tutorial presented at the 18th National Conference on Artificial Intelligence (AAAI), 2002

[6] Linden, G., Smith, B., York, J.: Industry Report: *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*. IEEE Distributed Systems Online 4(1), 2003

[7] Rahm, E., Vossen, G.: *Web & Datenbanken. Konzepte, Architekturen, Anwendungen*. Kap. 11: Data-Warehouse-Einsatz zur Web-Zugriffsanalyse. dpunkt 2003

[8] Mobasher, B., Dai, H., Luo, T., Sun, Y., Zhu, J. *Combining web usage and content mining for more effective personalization*. Proc. of the International Conference on ECommerce and Web Technologies (EC-Web), 2000

[9] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: *Analysis of recommendation algorithms for e-commerce*. Proc. of ACM E-Commerce, 2000

[10] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: *Item-based collaborative filtering recommendation algorithms*. Proc. of the 10th International World Wide Web Conference (WWW10), Hong Kong, May 2001

[11] Schafer, J. B., Konstan, J. A., Riedl, J.: *E-commerce recommendation applications*. Data Mining and Knowledge Discovery, 5(1/2), 2001

[12] Stöhr, T., Rahm, E., Quitzsch, S.: *OLAP-Auswertung von Web-Zugriffen*. Proc. GI-Workshop Internet-Datenbanken, Sep. 2000