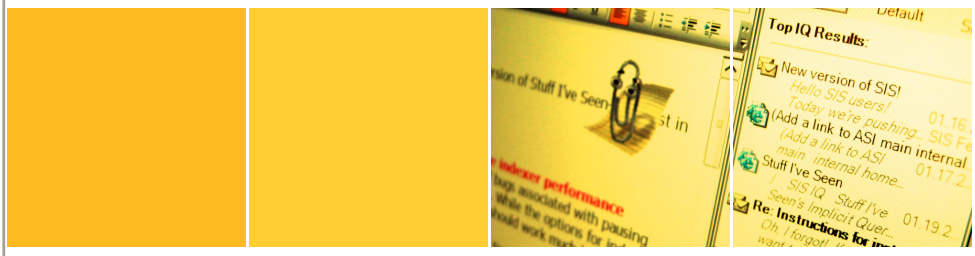
Andreas Henrich
Karlheinz Morgenroth (Hrsg.)



Kontextbasiertes Information Retrieval

Methoden und Anwendungen

Beiträge des Seminars
im Sommersemester 2004
an der Universität Bamberg

Lehrstuhl für Medieninformatik
Universität Bamberg

Andreas Henrich
Karlheinz Morgenroth (Hrsg.)

Kontextbasiertes Information Retrieval

Methoden und Anwendungen

Prof. Dr. Andreas Henrich
Dipl.-Wirtsch.Inf. Karlheinz Morgenroth

Lehrstuhl für Medieninformatik
Fakultät für Wirtschaftsinformatik und Angewandte Informatik
Universität Bamberg
Feldkirchenstraße 21
96045 Bamberg

ISBN 3-00-015171-0

Bibliographische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie.
Detaillierte bibliographische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Inhalt

Vorwort

Gegenstand des Information Retrieval ist die Suche nach Dokumenten. Traditionell handelt es sich dabei im Allgemeinen um Textdokumente. In neuerer Zeit kommt aber verstärkt auch die Suche nach multimedialen Dokumenten (Bilder, Audio, Video, Hypertext- Dokumente) hinzu. Ferner hat das Gebiet des Information Retrieval insbesondere auch durch das Aufkommen des Internets und World Wide Webs an Bedeutung und Aktualität gewonnen.

Im Detail geht es im Information Retrieval darum, aus einer Kollektion von Dokumenten zu einem von einem Nutzer explizit formulierten Informationswunsch relevante Dokumente zu ermitteln. Hierzu werden Techniken eingesetzt, die weit über eine einfache zeichenkettenbasierte Suche hinausgehen. So versucht man, von der konkreten Wortwahl in einem Dokument zu abstrahieren und stattdessen die Semantik des Dokumentes zu adressieren.

Kontextbasiertes Information Retrieval nutzt Informationen über die aktuelle Situation, den Kontext, in der sich ein Nutzer befindet, um daraus auf einen möglichen Informationsbedarf des Nutzers zu schließen. Der Kontext, in dem sich ein Nutzer befindet kann die aktuelle Tätigkeit oder auch seinen Aufenthaltsort umfassen. Ein entsprechendes kontextbasiertes Information Retrieval System beobachtet kontinuierlich einen Nutzer, erfasst entsprechende Daten über seinen Kontext und liefert zu diesem Kontext entsprechend relevante Informationen aus einer oder mehreren Datenquellen. Die erfassten Informationen über den Kontext eines Nutzers können neben der automatischen Anfragestellung auch zur Verfeinerung manuell gestellter Anfrage Verwendung finden.

In den vergangenen Jahren wurden von verschiedenen Autoren kontextbasierte bzw. berücksichtigende Information Retrieval Systeme und Ansätze vorgeschlagen. Diese Ansätze setzten jeweils auf verschiedene Quellen für die Gewinnung von Kontextinformationen über die aktuelle Arbeitsumgebung oder die Aufgaben, die eine Person zu einem bestimmten Zeitpunkt vollführt. Ebenfalls verwenden sie verschiedene Quellen für die Gewinnung von relevanten Informationen oder Dokumenten, die einem Benutzer in seiner aktuellen Arbeitssituation nützlich sein könnten.

Dieser Sammelband stellt die im Rahmen des Seminars “Kontextbasiertes Information Retrieval” im Sommersemester 2004 an der Otto-Friedrich Universität Bamberg entstandenen Beiträge zusammen. Diese Beiträge präsentieren jeweils Ansätze und Systeme, die sowohl dem Forschungsbereich des Information Retrieval als auch verwandte Gebieten entstammen.

Andreas Henrich, Karlheinz Morgenroth

Empfehlungen für lokale Dokumente

Christoph Reuter
christoph.reuter@stud.uni-bamberg.de

Abstract: Gegenstand dieser Arbeit sind Anwendungssysteme, die den Anwender beim Lesen und Verfassen von Dokumenten durch kontextbasiertes Information Retrieval unterstützen. Ziel ist es, dem Benutzer zur Erfüllung seiner Aufgabe Verweise auf nützliche Dokumente aufzuzeigen. Die Programme stellen dazu Anfragen an Suchmaschinen und präsentieren dem Anwender die aufbereiteten Ergebnisse. Charakteristisch für diese Art von Information Retrieval Agenten ist es, dass sie selbstständig im Hintergrund agieren. Im folgenden wird das grundsätzliche Vorgehen der Systeme im Allgemeinen erläutert. Anhand zweier konkreter Implementierungen soll auf die dabei eingesetzten Techniken des Information Retrieval eingegangen werden.

1 Einleitung

Mit der stetig steigenden Bedeutung des Internets und der damit immer größer werdenden Informationsflut wächst auch der Aufwand, um den in aller Regel sehr schlecht strukturierten Informationsbestand zu durchsuchen.

Allgemein befasst sich Information Retrieval damit, dem Anwender Informationen bereit zu stellen, die dessen Informationsbedarf abdecken [FB92]. Eines der Hauptziele ist es, Werkzeuge zu erschaffen, mit denen er möglichst einfach und schnell eine Anfrage (Query) an einen Datenbestand (Korpus) stellen kann. Die Antwort der Suchmaschine bildet dann eine Liste mit Dokumenten, die relevant sein könnten – z.B. weil der/die angegebene(n) Begriff(e) im Dokument enthalten sind.

Kontextbasiertes Information Retrieval versucht dem Benutzer dabei noch mehr Arbeit abzunehmen. Um möglichst nur relevante Dokumente vorzuschlagen, wird der lokale Kontext in der Anfrage explizit berücksichtigt.
An folgendem Beispiel soll verdeutlicht werden, welche Bedeutung dem Kontext dabei zukommt. Das Wort „Plasma“ zum Beispiel findet im alltäglichen Leben in sehr vielen Bereichen Verwendung. Eine große Rolle spielt es etwa in der Biologie sowie in der Medizin. Eine einfache Anfrage „plasma“ würde aber auch Ergebnisse aus den Bereichen Technik und eShopping (man denke etwa an Plasmabildschirme) liefern.

Eine Query kann nun präzisiert werden, indem man das Wissen über die aktuelle Tätigkeit und die Umwelt des Nutzers mit in die Anfrage einfließen lässt. In Abhängigkeit davon, was der Benutzer zur Zeit liest oder verfasst kann sein Informationsbedarf antizipiert und eine erweiterte Anfrage formuliert werden. Zusätzlich kann bereits eine Vorentscheidung getroffen werden, an welche Informationsquellen die Anfrage überhaupt gesendet werden soll. Da sich im Internet immer mehr domänenspezifische Portale etablieren, ist dies ein vielversprechender Ansatz.

Beim sogenannten „Just-In-Time Information Retrieval“ geschieht der oben beschriebene Prozess in regelmäßigen Abständen und vollautomatisch (pro-aktiv). Software Agenten versuchen den Kontext eigenmächtig zur erkennen. Der Benutzer muss diesen nicht explizit definieren, wie etwa bei der kontextbasierten Meta-Suchmaschine Inquirus 2 [GL99]. Im Anschluss wird die im Hintergrund generierte Suchanfrage an eine oder mehrere angebundene Quellen geschickt. Die gefundenen Dokumente werden dem Anwender vorgeschlagen noch bevor er selbst eine Query formulieren muss.

Bei der Präsentation der Ergebnisse soll der Anwender seine aktuelle Tätigkeit möglichst nicht unterbrechen müssen. Gleichzeitig soll es möglichst wenig Aufwand erfordern, die gefundenen Resultate zu überfliegen und gegebenenfalls einen interessant klingenden Verweis zu öffnen. Hier muss eine entsprechende Kompromisslösung gefunden werden, da sich beides auf Grund der begrenzten Wahrnehmungsfähigkeit des Menschen gegenseitig ausschließt [Br58].

Im weiteren Verlauf sollen nun zwei IR Agenten vorgestellt werden. Diese versuchen die oben skizzierte Aufgabenstellung zu lösen. Es handelt sich dabei um die Systeme „Remembrance Agent“ (RA) und Watson.

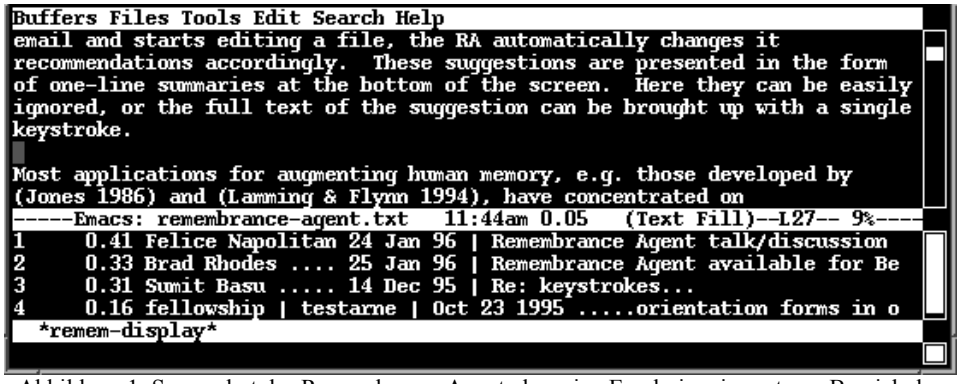


Abbildung 1: Screenshot des Remembrance Agent, der seine Ergebnisse im unteren Bereich des Bildschirms präsentiert

Der **Remembrance Agent** [Rh00m] stammt aus der Schmiede des MIT Media Labs. Er stellt eine Erweiterung des sehr mächtigen Texteditor EMACS dar. In einem eigenen Buffer (entspricht einem Ausschnitt des Anzeigefensters) präsentiert der RA zeilenweise seine Empfehlungen. Der Anwender kann einem Verweis mit speziellen Tastenkombinationen oder einem einfachen Mausklick nachgehen. Vorgestellt wurde der Remembrance Agent erstmals 1996 [RS96]. Die aktuellste Version des Produkts steht im Internet frei zum Download bereit[1].

Watson [BH00] ist ein Windows-basiertes System. Der auch als Information Management Assistant (IMA) bezeichnete Softwareagent unterstützt eine Vielzahl von Anwendungen wie beispielsweise Microsoft Word®, Internet Explorer® und Netscape Navigator®. Die Ausgabe der Ergebnisse erfolgt in einem eigenen Fenster, d.h. unabhängig von der jeweiligen Applikation. Watson entstand aus einem Projekt des Intelligent Information Laboratory der Nothwestern University.

[1] http://www.remem.org/

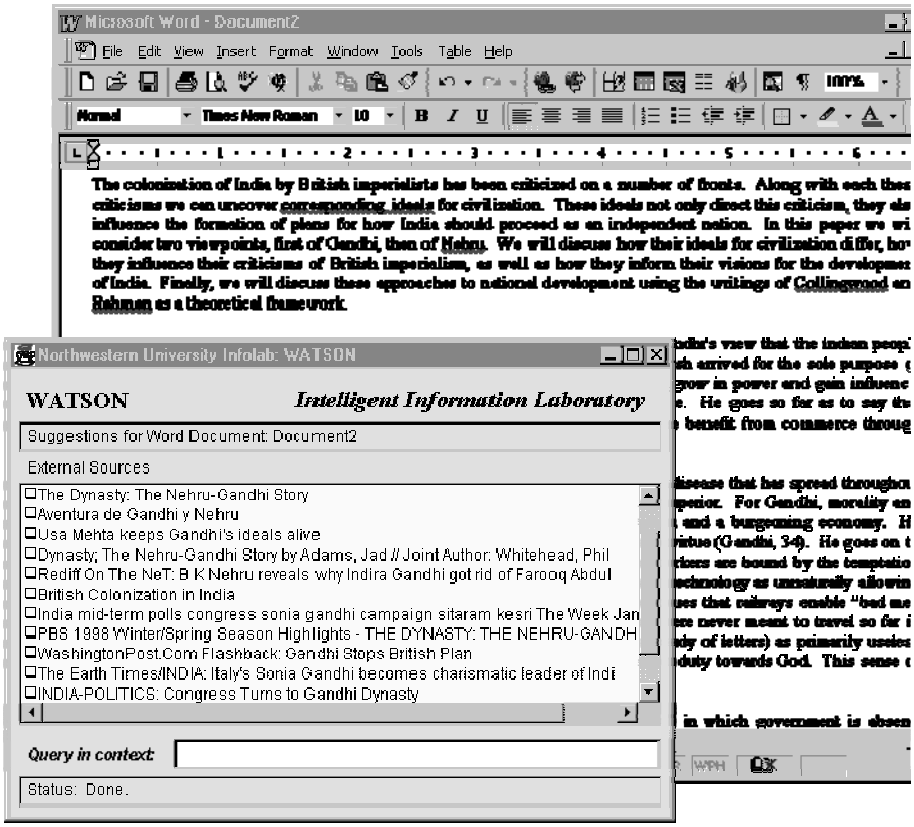


Abbildung 2: Screenshot des Information Management Assistant Watson mit Ergebnisfenster. Letzteres bietet dem Benutzer auch die Möglichkeit eine Query manuell zu formulieren.

2 Das Innenleben der Systeme

2.1 Kontext

Konkret stellt sich nun die Frage wie ein Rechner Informationen über den Kontext generieren und verarbeiten kann. Die hier vorgestellten Systeme beschränken sich dabei weitestgehend auf das Dokument, welches der Anwender gerade liest oder verfasst.

Der „Remembrance Agent“ zum Beispiel verwendet lediglich eine Textpassage – definiert als eine bestimmte Anzahl von Zeichen relativ zur aktuellen Cursorposition – um den Kontext zu charakterisieren. Mit dem Vektorraummodell wird seine Ähnlichkeit mit den einzelnen Dokumenten des Korpus bewertet.

Zudem erfolgt beim RA eine Aufspaltung des Dokuments in einzelne Textfelder durch die Technik des Template Matching. Eine E-Mail wird zum Beispiel in die Felder Empfänger, Betreff und Body aufgesplittet. Dazu existiert eine große Sammlung von Templates für die unterschiedlichsten Dateiformate (HTML, LaTex, E-Mail, u.a.). Unwichtige Felder und auch offensichtlich unwichtige Strukturen werden bereits an dieser Stelle eliminiert und von der weiteren Verarbeitung ausgeschlossen. Dazu zählen u.a. die gängigen Abkürzungen „Fwd:“ oder „Re:“ im E-Mailbetreff oder Signaturzeilen. Die Erkennung erfolgt durch „Reguläre Ausdrücke“ in Perl-Syntax, die fest in den Quellcode einprogrammiert sind. Eine Anpassung und Erweiterung des Systems ist dennoch recht einfach. Sie erfordert lediglich geringe Perl Kenntnisse [Si99].

Im Anschluss können auf die einzelnen Felder unterschiedliche Information Retrieval Methoden in Kombination angewendet werden. Man spricht in diesem Zusammenhang auch von der sogenannten „Data Fusion“ [Le95].

Bei längeren Textfeldern – wie z.B. dem Body einer E-Mail – erfolgt zudem eine gezielte Filterung. Hierzu zählen beim RA vor allem das „Stop-word Removal“ und das „Word Stemming“ [Po80]. Unter Stop-words versteht man die Wörter, die faktisch in jedem längeren Text vorkommen, also zum Beispiel Präpositionen, Artikel oder Pronomen. Diese Begriffe, die in einer Liste vorab definiert werden müssen, werden vor der Weiterverarbeitung eliminiert, da sie keine inhaltliche Bedeutung besitzen. Beim „Word Stemming“ (Stammformreduktion) werden unterschiedliche Wortformen auf ihren Wortstamm zurückgeführt. Es werden also z.B. „gelacht“ und „lachend“ einheitlich durch „lachen“ ersetzt. Dadurch können beim Gewichten der mehrfach auftretenden Begriffe (Term Weighting) wesentlich bessere Ergebnisse erzielt werden.

Während beim RA ausschließlich das Dokument als Input für den Kontext dient, geht Watson noch einen Schritt weiter. Dieser Agent berücksichtigt auch die Anwendung die der Anwender benutzt. Anhand der Art von Applikation wird versucht die Aufgabe des Benutzers zu antizipieren. Das System unterscheidet etwa ob der Benutzer eine E-Mail verfasst, ein Dokument oder ob er im Internet recherchiert. Abhängig vom Informationsbedarf wird dann vom „Ressource Selector“ vorab eine Auswahl der Informationsquellen getroffen, an welche die Suchanfrage geschickt werden soll. Durch „Application Adapter“ kann dieses System beliebig an weitere Anwendungen angepasst werden und deren jeweilige Dokumenten-Repräsentation verstehen. Im Application Adapter wird der Text mit den Formatierungen der Applikation in eine interne Repräsentation gebracht. Diese umfasst nur noch die Kategorien Normal, Hervorgehoben, Herabgestuft (de-emphasized) und Listeneintrag. Jeder Begriff wird anhand seiner Formatierung und Positionierung im Dokument in eine dieser Kategorien eingeordnet.

Parallel dazu betreibt Watson noch eine Art Mustererkennung, mit der bestimmte Strukturen erkannt und spezifisch behandelt werden können. Ein Bespiel dafür ist das Auftreten einer Adresse, zur der Watson automatisch einen Link zu einer Landkarte der entsprechenden Umgebung präsentiert.

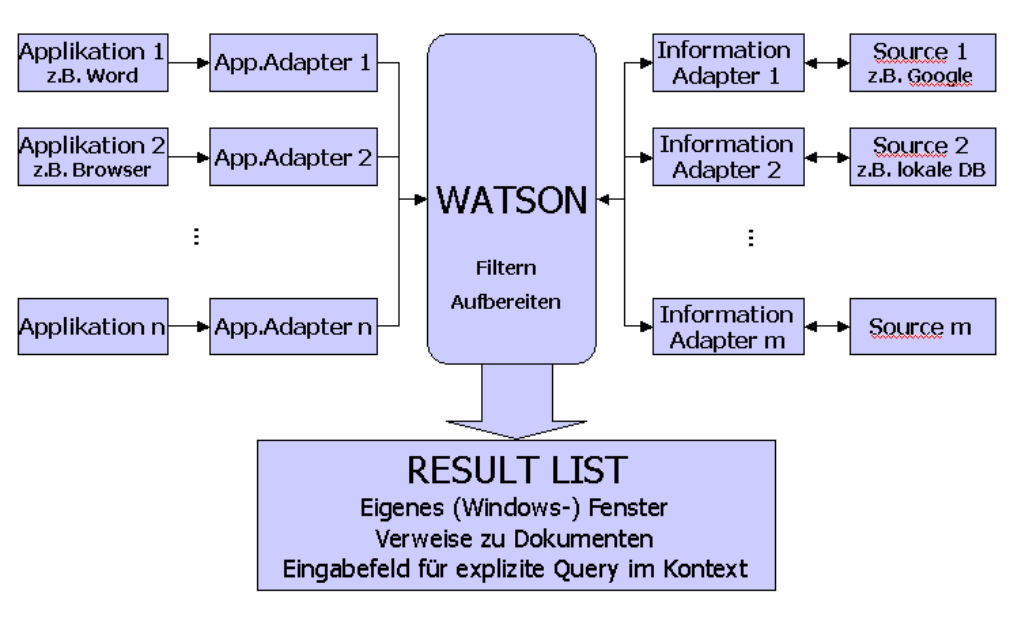


Abbildung 3: Durch den schematischen Aufbau wird die Skalierbarkeit des Agenten deutlich. Sowohl auf der Seite der Benutzeranwendungen als auch auf der Seite der zugreifbaren Informationsquellen lassen sich durch zusätzliche Adapter neue Systeme integrieren.

Die Entwicklung eines Benutzerprofils findet in keinem der beiden Systeme statt. Ein derartiger Ansatz wurde nicht gewählt, da man davon ausgeht, dass sich die wechselnden Aufgaben sehr stark unterscheiden – unabhängig von der Person des Bearbeiters. Die Erstellung eines Benutzerportraits wäre bei der Suche nach Empfehlungen zum aktuell bearbeiteten Dokument nicht hilfreich. Es gibt aber durchaus auch Systeme, die ihren Kontext um Benutzerinformationen erweitern. Ein Beispiel dafür ist Suitor [MC01] welches unter „Verwandte Systeme“ kurz vorgestellt wird.

2.1 Korpus und Query

Der Remembrance Agent überprüft im Abstand von beispielsweise 5 Sekunden, ob Änderungen im Kontext aufgetreten sind. Gegebenenfalls erzeugt er dann nach dem oben geschilderten Verfahren eine neue Query und sendet diese an die angebundenen Quellen. Deren Einbindung findet in der Konfiguration des Agenten statt. In einer Liste werden die Einstellungen für jeden sogenannten Scope festgelegt. Ein Scope ist ein Savantprozess, der ein Dateiarchiv anspricht. Für einem Scope werden eine bestimmte Anzahl von Bildschirmzeilen definiert – und damit die Anzahl der Ergebnisse, die er zurückliefern soll.

Es ist nun möglich mehrere Anfragen zu einem Dokument automatisiert abzusetzen. Dies ist sinnvoll, da hierdurch zum einen unterschiedliche Quellen angebunden werden können, andererseits dieselbe Quelle mit unterschiedlichen Kontexten – und dadurch mit unterschiedlichen Anfragen – angebunden werden. Des weiteren wird für jeden Scope auch die Zeit definiert, nach der eine Überprüfung auf Änderungen im Kontext stattfinden soll. Im folgenden Beispiel werden die Anpassungsmöglichkeiten verdeutlicht.

Man kann z.B. einen Scope definieren, der eine Anfrage basierend auf den letzten 200 Wörtern (relativ zur Position des Cursors) generiert und ihm zur Präsentation der Ergebnisse 3 Zeilen einräumen. Ein weiterer Scope könnte an dasselbe Archiv eine Anfrage stellen, die auf den letzten 50 Begriffen basiert aber in kürzen Zeitabständen wiederholt wird, sofern sich der Cursor weiter bewegt hat.

Auch an dieser Stelle geht Watson noch einen Schritt weiter. In Abhängigkeit des Kontextes wählt er gezielt die jeweiligen Quellen aus. Diese werden durch so genannte „Information Adapter“ angekoppelt. Letztere stellen neben Metadaten zur Quelle auch den einheitlichen Zugriff sicher. Mit ihrer Hilfe kann zum Beispiel sowohl auf Internetsuchmaschinen als auch auf lokale Datenbanken zugegriffen werden. Eine weitere Funktionalität der Information Adapter besteht darin, die allgemeine Query in eine quellenspezifische Anfrage zu transformieren. Dies sichert auch an dieser Stelle gute Anpassbarkeit hinsichtlich unterschiedlichster Quellen.

Nun soll erläutert werden, wie Watson aus der Repräsentation eines vorliegenden Dokuments eine allgemeine (also quellenunspezifische) Query erzeugt.

Schrittweise werden unwesentliche Begriffe entfernt und die restlichen Begriffe gewichtet. Dazu dienen die folgenden heuristischen Regeln.

1. Entfernen der „stop words“:
   (Erläuterung vgl. Remembrance Agent)

2. Gewichten der Häufigkeit des Auftreten eines Begriffes:
   Es wird dabei davon ausgegangen, dass die häufiger vorkommenden Wörter signifikanter für den Inhalt des Dokumentes sind.

3. Stärkeres Gewichten von Hervorhebungen:
   Begriffe in Überschriften oder hervorgehobenen Textpassagen gelten als besonders repräsentativ.

4. Stärkeres Gewichten von Begriffen, die früher im Text auftreten:
   Weil Lesen ein linearer Prozess ist, geht man davon aus, dass die Begriffe der ersten Absätze von größerer Bedeutung für den Inhalt sind als die später vorkommenden Ausdrücke.

5. Abwerten von Herabgestuften Begriffen:
   Ausdrücke die immer nur mit kleinerem Schriftgrad auftreten, erhalten eine entsprechend geringe Gewichtung, da sie als nicht repräsentativ für den Inhalt des Textes gelten.

6. Ordnung der Begriffe einer Liste vernachlässigen:
   Listen gelten als Sonderfall hinsichtlich der Regel Nr. 4 und werden gleichmäßig gewichtet.

7. Vernachlässigen von irrelevanten Dokumentbausteinen:
   Ausdrücke die zum Beispiel zur Navigation einer umfangreichen Webseite gehören, haben keine inhaltliche Bedeutung für den eigentlichen Text und werden daher von der eigentlichen Analyse ausgeschlossen.

Die anschließende Gewichtung der restlichen Begriffe folgt einem Algorithmus, der durch folgenden Pseudo-Code beschrieben wird:

```
for each word w collected
        for each (position p, style s) in pos(w)
                weight  :=  1 + (numTerms² * δ)/p²
                if (weight > maxCount) or
                        (s is the list item style) or
                        (s is the de-emphasized style) then
                                weight  :=  1
                else if (s is the emphasized style)  then
                        weight  :=  2 * weight
                weight(w)  :=  weight(w) + weight
```

Erläuterungen:

δ:	Konstanter Faktor, z.B. 0,2 - er legt fest wie stark der Einfluss der Position p des Begriffs innerhalb des Dokumentes sein soll
maxCount:	Anzahl des Auftreten des häufigsten Begriffes
numTerms:	Anzahl der Begriffe des Dokumentes (*nach* Anwendung der oben genannten Regeln)

Zum Schluss werden die Begriffe entsprechend ihres errechneten Gewichts sortiert. Die Query umfasst die zwanzig am stärksten gewichteten Begriffe in der ursprünglichen Reihenfolge ihres Auftretens im Originaldokument. Innerhalb des jeweiligen Information Adapter kann die allgemeine Query unter Umständen noch individuell auf die Quelle zugeschnitten werden. Ein Beispiel sind Suchmaschinen die zusätzliche Bool'sche Operatoren erlauben.

2.3 Verarbeitung der Anfrage

Da bei Watson die Durchführung der Anfrage sehr stark von dem Information Adapter und der jeweiligen Quelle abhängig ist, soll dies hier nur für dass RA System betrachtet werden. Der Remembrance Agent greift dabei einheitlich auf eine Information Retrieval Engine namens Savant zurück.

Die beiden Hauptaufgaben dieses Systems bestehen zum einen in der automatischen Indizierung des Corpus sowie zum anderen in der Durchführung des eigentlichen Retrievals.

Bei der Indizierung geht es darum, alle bereitgestellten Dokumente aufzubereiten. Die besondere Stärke von Savant liegt dabei in einer ausgefeilten Template Matching Engine, durch die unterschiedlichste Dokumentformate (also z.B. E-Mailarchive, HTML- oder LaTeX-Dokumente) in einzelne Textfelder untergliedert werden. Verschiedene Dokumentformate werden dadurch besser vergleichbar hinsichtlich ihrer Ähnlichkeit (Similarity), also zum Beispiel der Titel einer HTML-Seite (<TITLE>...</TITLE>) mit dem Betreff einer E-Mail (Subject: ...).

Das eigentliche Term-Weighting und damit die Bewertung der Relevanz des vorliegenden Textabschnittes mit den Dokumenten, die im Korpus verfügbar sind, erfolgt wie bereits erwähnt entsprechend dem Vektorraummodell.

Um die Dokumente im Korpus in eine Reihenfolge zu bringen, geht man erstens davon aus, dass innerhalb eines Dokumentes, diejenigen Begriffe besonders signifikant sind, die am häufigsten vorkommen (NTF, Normal Term Frequency). Zweitens gelten innerhalb des Korpus aller Dokumente diejenigen Wörter, die in nur wenigen Dokumenten auftreten, als besonders auszeichnend (iDF, inverse Document Frequency) [Ha92].

In Savant sind zwei Algorithmen implementiert die im folgenden erläutert werden.

Der erste Algorithmus basiert auf den Überlegungen von [CH79] und [Ha86]

$$IDF_T = \log\left(\frac{(N - n_T)}{n_T}\right) \quad \text{[CH79]}$$

$$NTF_T = \frac{\log(freq_{dT} + 1)}{\log(length_d)} \quad \text{[Ha86]}$$

$$query\ normalization = \frac{1}{C}$$

$$similarity = \sum_{T \in Q} (NTF \cdot IDF_T \cdot C)$$

Bei dem zweiten Algorithmus handelt es sich in der derzeitigen Version um das Okapi weighting scheme ohne Relevance Feedback [Ro95].

$$similarity = \sum_{T \in Q} \frac{W \cdot freq_{dT} \cdot freq_{qT} \cdot (k_1 + 1)\ (k_3 + 1)}{(K + freq_{dT})\ (k_3 + freq_{qT})}$$

Erläuterungen:

Q Vektor mit den Begriffen T der Anfrage

W "Inverse Document Frequency", berechnet mit

$\log(N - n_T + 0{,}5) - \log(n_T + 0{,}5)$

N Gesamtzahl der Dokumente

n_T Anzahl der Dokumente die den Begriff T beinhalten

$freq_{dT}$ Auftrittshäufigkeit von T innerhalb des Dokumentes

$freq_{qT}$ Auftrittshäufigkeit von T innerhalb der Query

k_1 Tuningparameter, mit steigendem Wert steigt auch die Gewichtung von $freq_d$

Vorgeschlagener Wert: 1,2 (in Anlehnung an [Wa98])

k_3 Tuningparameter, mit steigendem Wert steigt auch die Gewichtung von

$freq_q$

Vorgeschlagener Wert: 100 (in Anlehnung an [Wa98])

K $K = k_1 \cdot \left((1-b) + \frac{b \cdot dl}{avdl}\right)$

b Tuningparameter, mit steigendem Wert werden große Dokumente „bestraft“

Vorgeschlagener Wert: 0,75 (in Anlehnung an [Wa98])

dl Länge des Dokumente (Anzahl der Begriffe)

avdl Durchschnittliche Dokumentenlänge

Im Anschluß an die Berechnung erfolgt die Normalisierung der Ergebnisse unter Bezugnahme auf die Länge der Query.

$$normalization\ factor = \frac{C}{length_q}$$

C Konstante um den Faktor im Intervall [0; 1] zu positionieren (ein Standardwert für C ist 3,0)

$length_q$ Anzahl der verschiedenen Begriffe in der Anfrage

Die Normalisierung der Query ermöglicht es, Empfehlungen auch im Zeitverlauf, also bei sich ändernden Anfragen, vergleichbar zu halten. Der RA wird also ältere Empfehlungen nicht unmittelbar ersetzen, wenn durch neuere Anfragen nur Dokumente geliefert werden, die lediglich sehr geringere Relevanzwerte erzielen.

2.4 Ergebnisverarbeitung und -präsentation

Bei der Ergebnisverarbeitung ist das primäre Ziel doppelte Empfehlungen zu erkennen und gegebenenfalls auszusortieren. Schließlich müssen die Ergebnisse noch dem Anwender präsentiert werden. Diese - auf den ersten Blick eher einfache - Aufgabe, hat jedoch weitreichende Konsequenzen bezüglich der beabsichtigten Unterstützung des Nutzers. Eine zu aufdringliche Präsentation birgt die Gefahr, dass sich der Benutzer durch ständige Unterbrechungen gestört fühlen könnte. Dieser Eindruck könnte sich noch verstärken, wenn viele Empfehlungen erscheinen, die für ihn praktisch keinen Nutzen haben. Wählt man andererseits den Weg die Ergebnisse sehr unauffällig zu präsentieren, kann es passieren, dass Aktualisierungen vom Anwender nicht wahrgenommen werden. Der Agent leistet auch in diesem Fall nicht die gewünschte Unterstützung. Zuletzt muss noch erwähnt werden, dass auch der Aufwand, der mit dem Öffnen einer vielversprechenden Empfehlung möglichst klein gehalten werden sollte.

Beim RA geschieht das Filtern der doppelten Ergebnisse bereits auf der Ebene von Savant. Da der Benutzer mehrere Scopes definieren kann und jeder Scope einen eigenen Savant-Prozess startet, kann es doch zu doppelten Einträgen kommen. Dieses Problem ist bisher noch ungelöst. Zur Darstellung der Ergebnisse wird auf ein sogenanntes „Ramping Interface“ gesetzt. Der Name rührt daher, dass die präsentierte Information schrittweise verdichtet wird. Im ersten Schritt wird innerhalb einer Zeile der Titel, evtl. der Autor eines Dokuments sowie den errechneten Relevanzwert präsentiert. Erst wenn das Interesse des Nutzers geweckt ist und er mehr wissen möchte, kann er den nächsten Schritt veranlassen. Bevor er das Dokument an sich öffnet, kann er sich durch das Überfahren der Zeile mit dem Mauszeiger die wichtigsten Begriffe des Dokuments anzeigen lassen. Es wird dem Benutzer also immer ein kleines bisschen mehr Aufwand abverlangt, um zu sehen, ob er (noch) an der Empfehlung interessiert ist.

Bei Watson erfolgt das Filtern im sogenannten „Result Processor“ zentral über sämtliche zurückgelieferten Verweise. Er prüft zur Erkennung der Duplikate erstens auf Gleichheit des Dokumententitels und zum zweiten auf Gleichheit des URLs. Bei Letzterem reicht bereits eine Übereinstimmung der Verzeichnisstruktur, womit der im Internet häufig anzutreffenden Spiegelung kompletter Seiten Rechnung getragen werden soll. Die Präsentation der Empfehlungen findet wie bereits erwähnt in einem eigenen Fenster statt.

3 Evaluation

Zur Evaluation der beiden Systeme wurden mehrere Studien durchgeführt. Die Ergebnisse sollen hier sehr knapp vorgestellt werden. Detaillierte Ausführungen können zu Watson in [BH00] und zum Remembrance Agent in [RH00m] und [RS96] gefunden werden.

In einer Testreihe zum Remembrance Agent waren 27 Studenten dazu angehalten, einen Aufsatz zu einem vorgegebenen Thema zu schreiben. Die Zeitspanne zur Erledigung der Aufgabe wurde auf 45 Minuten festgelegt, was nach Aussage der Teilnehmer ausreichend war. 13 Studenten wurden in der Aufgabestellung explizit dazu aufgefordert, bei der Erledigung auf den RA zurückzugreifen. Die restlichen 14 Teilnehmer bildeten die Kontrollgruppe und wurden nur mit EMACS und Zugriff auf eine Internetsuchmaschine ausgestattet. Im Ergebnis wird deutlich, dass die Mitglieder der RA-Nutzergruppe etwa die 2,5-fache Anzahl an Textdokumenten aufriefen als die Mitglieder der Kontrollgruppe. Auch die Anzahl der von ihnen in der Arbeit zitierten Literaturverweise war bei den Teilnehmer mit Nutzung des RAs durchgängig höher.

In einer weiteren Studie wird jedoch deutlich, dass der Anteil der für den Anwender interessanten Empfehlungen noch sehr gering ist (gemessen an der Gesamtheit der unterbreiteten Verweise). So ergab eine Langzeitstudie, dass von den 186.480 Empfehlungen, die der RA machte, die Anwendergruppe nur etwa 197 (0,1%) nachging. Anschließend erfolgte eine Bewertung des Links (von 1 für „nicht hilfreich“ bis 5 für „sehr nützlich“), die allerdings auf Grund des zusätzlichen Arbeitsaufwands nicht verpflichtend war und nur in 49 der 197 Fälle (25%) wahrgenommen wurde. Daraus ergab sich ein näherungsweise gleichverteiltes Ergebnis.

Zusammenfassend lässt sich feststellen, dass durch die Kontexteinbeziehung unter gelieferten Empfehlungen auch relevante und nützliche Dokumente waren, deren Auffinden beim Benutzen konventioneller Suchmaschinen einige Mühe bedeutet hätte.

Zur Evaluation des IMAs Watson wurde im Rahmen einer Studie zehn Wissenschaftler dazu angehalten, ihre letzte wissenschaftliche Arbeit einzureichen. Diese wurden in Word® geöffnet und die Liste der Empfehlungen, die Watson daraufhin generierte wurde an die Forscher zurückgeschickt. Jeder sollte dann für die einzelnen Verweise seiner Liste beurteilen, ob sie zum Verfassen des Dokuments nützlich bzw. hilfreich gewesen wären. Immerhin 8 der 10 Beteiligten bewerteten mindestens eine Empfehlung als nützlich. Vier davon räumten des weiteren ein, dass ihnen die gefundenen Arbeiten bisher nicht bekannt waren und möglicherweise in zukünftige Arbeiten mit einfließen könnten. Die meisten Texte jedoch waren zwar lexikalisch ähnlich, hatten aber inhaltlich nicht viel mit der Arbeit zu tun. Watson kommt an dieser Stelle sehr zu gute, das er in seinem Ergebnisfenster sehr viele Verknüpfungen präsentieren kann. Allerdings ist es für den Benutzer sehr aufwändig diese alle zu überfliegen.

Zum Vergleich des Agenten mit der Nutzung herkömmlicher Suchmaschinen wurde eine weitere Untersuchung durchgeführt. Sechs Studenten, die angaben mit der professionelle Nutzung von Altavista® vertraut zu sein, bekamen die Aufgabe zu einem vorgegebenen Dokument weitere relevante Dokumente zu suchen. Bei der Durchführung schnitt Watson in 15 der insgesamt 19 Fälle besser oder mindestens genauso gut ab wie die menschlichen Kandidaten. Im Durchschnitt waren 5 von 10 seiner Empfehlungen „relevant“, während dies bei den Probanden durchschnittlich nur auf 3 von 10 Dokumenten zutraf.

Abschließend lassen sich folgenden Rückschlüsse ziehen. Zum einen hängt die Nutzung des Agenten sehr stark davon ab, ob und wann der Benutzer die gelieferten Ergebnisse überhaupt betrachtet. Eine wesentliche Stellschraube bei der Entwicklung derartiger Systeme ist also die Ergebnispräsentation.

Andererseits stellt sich die Frage, wie gut sich mit den Mitteln des Information Retrieval tatsächlich der Informationsbedarf des Anwenders bestimmen lässt. Zwar wird es in der Regel für einen Autor eines Dokumentes über z.B. die Entwicklung der Kaufkraft im osteuropäischen Wirtschaftsraum von Nutzen sein Dokumente präsentiert zu bekommen, die Ähnlichkeit mit diesen Schlagworten haben. Aber will dies auch, derjenige, der nur dabei ist, ein Kapitel eines vor ihm auf dem Schreibtisch liegenden Buches zusammenzufassen? Es kann auch keine Aussage darüber getroffen werden, ob der Verfasser die präsentierten Dokumente eventuell schon kennt, ob diese seinen qualitativen Ansprüchen genügen oder ob er nicht vielleicht schon zu viel Material gesammelt hat und mit der Sichtung bereits überfordert ist.

4 Verwandte Systeme

Zum Schluss soll eine Reihe weiterer Systeme vorgestellt werden, die ähnliche Ziele verfolgen.

Radar [CS98] ist eine Variante des Remembrance Agent für die Windows-Welt und Microsoft Word®. Ähnlich zum Watsonsystem werden auch hier die Ergebnisse in einem eigenen (Windows-) Fenster präsentiert.

Des weiteren gehören zu den verwandten Systemen die „**Margin Notes**“, ein Agent, der ebenfalls auf dem vom RA genutzten IR-Engine Savant aufbaut. Das Programm beschränkt sich jedoch darauf, Empfehlungen zur aktuell betrachteten Internetseite zu geben. Die Präsentation weitergehender Links erfolgt – wie der Name des Systems andeutet – in einem künstlich an die Seite angefügten „Rand (für Anmerkungen)“. In diesem wird auf der Höhe jedes neu beginnenden Absatzes des HTML-Dokumentes eine Empfehlung angezeigt. Unabhängig von den Farben der jeweiligen Seite wird der Rand als schwarzer Streifen mit weißer Schrift präsentiert, wodurch eine klare Trennung zwischen der ursprünglichen Seite und den Empfehlungen erzielt werden soll. Eine detaillierte Vorstellung des Agenten erfolgt in [Rh00j].

Beim **Suitor** (Simple User Interest Tracker) handelt es sich um eine System, dass den Benutzer während seiner Tätigkeit ständig mit Empfehlungen aber auch aktuellen Nachrichten versorgt. Interessiert sich der Benutzer für einen Vorschlag, so wird dies vom System vermerkt und in Zukunft berücksichtigt. Den eigentlichen Kern des Suitor bildet eine Art „Schwarzes Brett“, auf dem eine Vielzahl von Agenten ihre Informationsangebote und –bedarfe (die sogenannten „Investigators“) anbieten. Durch sogenannte „Reflectors“ werden die Information mit dem Benutzerprofil abgeglichen und über ihre Relevanz entschieden. Durch die Gruppe der Actors kann schließlich eine Präsentation von Empfehlungen an den Benutzer erfolgen (z.B. in Form einer Scrollbar

ähnlich einem Börsenticker). Das System entstand im IBM Almaden Research Center rund um Paul P. Maglio und Christopher S. Campbell [Ma00]

Literaturverzeichnis

[BH99] J. Budzik, K. Hammond. Watson: Anticipating and Contextualizing Information Needs, in Proc. of the 62nd Annual Meetings of ASIS, 1999.

[BH00] J. Budzik, K. Hammond. User interactions with everyday applications as context for just-in-time information access. In Proceedings of the 2000 International Conference on Intelligent User Interfaces, New Orleans, Louisiana, ACM Press , 2000.

[Br58] D. E. Broadbent. Perception and Communication, 1958.

[CH79] W. Croft and D. Harper. Using Probabilistic Models of Document Retrieval Without Relevance Information, Documentation, 35(4), pp. 285-295, 1979.

[CS98] I.B. Crabtree, S. Soltysiak and M. Thint. Adaptive Personal Agents, Personal Technologies 2, No. 3, pp. 141-151, 1998.

[FB92] W. Frakes, R. Baeza-Yates (eds.), Information Retrieval: Data Structures and Algorithms, 1992.

[GL99] Eric J. Glover, Steve Lawrence, William Birmingham, and C. Lee Giles. Architecture of a metasearch engine that supports user information needs. In Eighth International Conference on Information and Knowledge Management, CIKM 99, pages 210-216, Kansas City, Missouri, November 1999.

[Ha86] D. Harman. An Experimental Study of Factors Important in Document Ranking, in ACM Conference on Research and Development in Information Retrieval, 1986.

[Ha92] D. Harman. Ranking Algorithms, in "Information Retrieval: Data Structures and Algorithms", W. Frakes and R. Baeza-Yates (eds.), 1992.

[Le95] Lee, J. Combining Multiple Evidence from Different Properties of Weighting Schemes, in SIGIR'95, pp. 180-188, 1995.

[Ma00] P. Maglio et al. SUITOR: An Attentive Information System, in The Proc. of IUI 2000, pp. 169-176, January 9-12, 2000.

[MC01] P. Maglio, C.S. Campbell, R. Barrett and T. Selker. An architecture for developing attentive information systems. Knowledge-based Systems 14, pp. 103-110, 2001.

[Po80] M. Porter. An Algorithm For Suffix Stripping, Program 14(3), pp. 130-137, July 1980.

[Rh00j] Bradley J. Rhodes. Margin Notes: building a contextually aware associative memory, in Proceedings of the International Conference on Intelligent User Interfaces (IUI'00), ACM Press, pp. 219-224, January 2000.

[Rh00m] Bradley J. Rhodes. Just-In-Time Information Retrieval. PhD thesis, MIT Media Laboratory, Cambridge, MA, May 2000.

[Ro95] S. Robertson et al. Okapi at TREC-3, in NIST Special Publication 500-226, 1995.

[RS96] Bradley J. Rhodes and Thad Starner. Remembrance Agent: A continuously running automated information retrieval system. In Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96), pages 487-495, 1996.

[Sa75] G. Salton et al. A vector space model for automatic indexing, CACM, 18, pp. 613-620, 1975.

[Si99] E. Siever et al. Perl in a Nutshell, pp. 63-70, 1999.

[Wa98] S. Walker et al. Okapi at TREC-6, in NIST Special Publication 500-240, 1998.

Empfehlung von Webseiten

Raiko Eckstein
mail@raiko-eckstein.de

Abstract: In diesem Artikel soll die Frage geklärt werden, inwiefern lernende Agenten Browsing im World Wide Web unterstützen können. Es wird das System WebWatcher vorgestellt. Der Fokus liegt dabei auf der Vorstellung der Techniken der Wissensgewinnung, einerseits durch das Lernen aus vergangenen Touren, und andererseits durch die Analyse der Hypertextstruktur mittels Reinforcement Learning. Danach werden diese Methodiken verglichen und einem Experiment, in dem Menschen die Aufgabe des Agenten übernehmen mussten, gegenübergestellt. Abschließend werden kurz einige weitere Projekte vorgestellt und ein Ausblick auf die zukünftige Entwicklung gegeben.

Einleitung

Das World Wide Web ist seit seinen Anfängen inzwischen zu einer riesigen Ansammlung von Dokumenten geworden, die lose miteinander verknüpft sind. Will sich ein Anwender in diesem Geflecht von Informationen orientieren, ist er auf Hilfsmittel angewiesen, die ihm bei der Suche nach für ihn relevanten Informationen leiten. Man kann verschiedene Herangehensweisen für die Suche nach Informationen unterscheiden. Zum Beispiel kann man auf eine der vielen Suchmaschinen, z.B. Google[2] zurückgreifen und dort seinen Informationswunsch eingeben sowie hoffen, dass in der Ergebnismenge relevante Dokumente zu finden sind. Internetkataloge, wie z.B. das Open Directory[3], in denen manuell versucht wird, interessante Seiten zu kategorisieren, erlauben dem Nutzer entlang von Cluster-Hierarchien zu „wandern“ und sich so spezielleren bzw. allgemeineren Themen zuzuwenden.

Eine weitere Möglichkeit der Informationssuche im WWW namens „Browsing“ besteht darin, ausgehend von einer Webseite einzelnen Links zu folgen und so weitere Informationen zu erhalten. Oft beinhalten Webseiten eine Vielzahl von Links, wodurch es relativ wahrscheinlich ist, dass der Benutzer relevante Verweise übersieht. Hier wäre es sinnvoll, wenn der Surfer einen Helfer zur Seite gestellt bekäme, der ihn auf relevante Informationen versprechende Seiten hinweist. Ein Beispiel für einen derartigen Helfer ist das Projekt WebWatcher[4].

---

[2] http://www.google.de
[3] http://www.dmoz.org
[4] http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html

WebWatcher

WebWatcher hat sich zum Ziel gesetzt, den Benutzer auf seinem Weg durch das World Wide Web zu begleiten. Oft „irrt“ ein Nutzer durch die Weiten des Internets mit einem Informationsbedürfnis, welches er nicht genau spezifizieren kann. Nach einer Klassifikation von Meadow [Me92] ist dieser Informationswunsch als „general-information search“ einzuordnen. Der Nutzer hat also eine sehr allgemeine Vorstellung des eigenen Interesses, welches schwer in Worte zu fassen ist. Die genaue Zielrichtung wird erst klar, während man durch das Internet surft. Gegensätzlich anzusiedeln ist die „known-item search“, also die Suche nach einem konkreten Thema, Fachgebiet o.Ä. Dies geschieht üblicherweise über eine konventionelle Suchmaschine. WebWatcher will den Nutzer an die Hand nehmen, und ihm auf seinem Weg zur Befriedigung seiner Informationswünsche behilflich sein.

Eine Tour mit WebWatcher

WebWatcher ist ein Agent, der sich zwischen den Nutzer und das Internet schaltet und als Proxy-Server agiert. Das heißt, WebWatcher lädt die vom Nutzer angeforderten Seiten aus dem Internet, fügt seine Änderungen und Empfehlungen auf der Seite ein und liefert die Webseite an den Nutzer aus. Dies hat den Vorteil, dass WebWatcher unabhängig vom Browser des Nutzers ist. Im Gegensatz zu anderen „Tour Guide Agents“ speichert WebWatcher kein Interessenprofil eines Nutzers, sondern orientiert sich an der Gesamtheit der in der Vergangenheit vermittelten Touren und deren Einschätzung durch den Nutzer. Der Nutzer kann WebWatcher mitteilen, ob die Tour erfolgreich war und er sein Informationsbedürfnis befriedigen konnte.

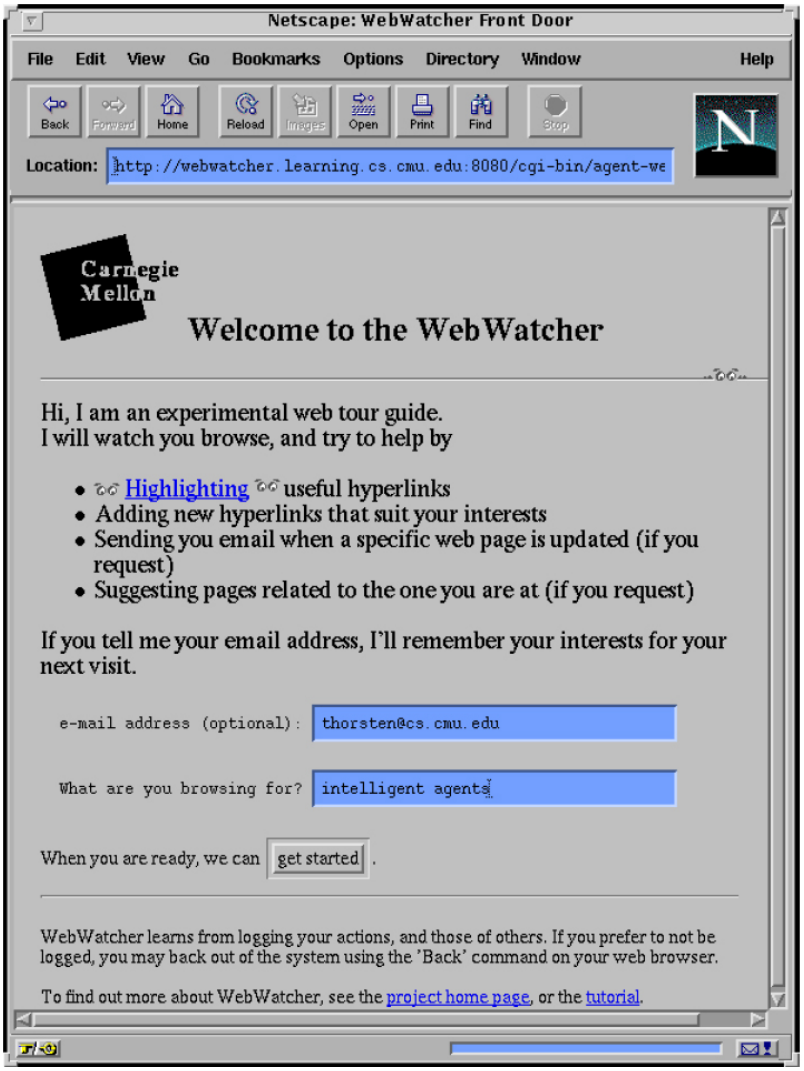


Abbildung 4: Start von WebWatcher. Hier kann der Nutzer sein Interesse eingeben. (Quelle: [WW97])

Eine WebWatcher Tour beginnt mit dem Aufruf der WebWatcher-Startseite[5], auf der man wie in Abbildung 4 zu sehen, mit der Eingabe des Interesses des Nutzers im Klartext in einem HTML-Formular fortfährt. Nach einem Klick auf den „Get Started“ Button beginnt die Tour in Begleitung von WebWatcher. Dieser macht sich durch ein paar Änderungen auf der Seite bemerkbar.

[5] Stand Juni 2004, nicht mehr online.

Wie in Abbildung 5 zu erkennen, fügt WebWatcher oberhalb der Originalseite ein Menü ein, in dem man mit WebWatcher kommunizieren und dem Agenten verschiedene Befehle zuweisen kann, u.a. ob das Ziel der Tour erreicht wurde, oder nicht. Darunter wird eine Liste von Links aufgeführt, die auf Seiten verweisen, die in vergangenen Touren von anderen Benutzern aufgerufen worden sind und die Worte des aktuellen Interesses beinhalten. Die Hauptänderung erfahren die Links, die WebWatcher empfiehlt. Sie werden links und rechts mit einem Augenpaar () markiert, was darauf hinweist, dass die mit den Links verknüpften Seiten gemäß WebWatcher dem Interesse des Nutzers am Besten entsprechen. Der Nutzer hat zu jedem Zeitpunkt der Tour die Möglichkeit, WebWatcher mitzuteilen, ob das Ziel erreicht wurde, oder nicht. Weiterhin kann man sich vom Agenten anzeigen lassen, wie oft die Links der gerade betrachteten Seite, von anderen Besuchern besichtigt worden sind. Außerdem kann man sich auch per E-Mail benachrichtigen lassen, sobald sich eine Seite, die man gerade besucht hat, verändert hat.

Abbildung 5: Die von WebWatcher veränderte Startseite der Tour. (Quelle: [WW97])

Wissensgewinnung

Durch die Breite und Tiefe des Internets einerseits und durch die schnellen Aktualisierungs- bzw. Veralterungszyklen andererseits wird schon seit langem jedweder Versuch, die Informationen und deren Verknüpfung über Webseiten hinweg manuell erfassbar zu machen, zunichte gemacht. Deshalb benötigt man für einen Agenten wie WebWatcher Automatismen, die das notwendige Wissen bereitstellen bzw. erzeugen können. Die Datenbasis für den Agenten muss automatisch erstellt werden. In diesem Dokument werden zwei grundsätzliche Möglichkeiten des maschinellen Lernens vorgestellt und bewertet.

Der erste Part setzt sich mit der Frage auseinander, wie man das Wissen aus vorherigen Touren für die Empfehlung von neuen Touren nutzen kann. Dazu wird versucht, anhand von Vergangenheitsdaten ein Verhalten für WebWatcher zu erlernen.

Die andere Herangehensweise nutzt das Wissen über die Dokumente selber und die Hypertextstruktur aus. Letzteres stellt dabei die Verknüpfung der Seiten untereinander dar. Gesucht wird ein Weg durch den Hypertext, der am Meisten dazu beiträgt, den Informationswunsch des Nutzers zu erfüllen.

Um Webseiten vorzuschlagen, benötigt WebWatcher eine Funktion, die die Links einer Seite gemäß dem Interesse des Nutzers bewertet und gegebenenfalls empfiehlt. Die Funktion sieht idealerweise folgendermaßen aus:

$$OptimalLinks?: Page \times Interest \times User \rightarrow 2^{Links(p)} \quad (1)$$

Dabei sollen für einen Nutzer mit Interesse $i \in Interest$ Links der Webseite $p \in Page$ vorgeschlagen werden. Dabei beschreibt der Ergebnisterm $2^{Links(p)}$ die Potenzmenge der auf der Seite p enthaltenen Links, d.h. jede mögliche Teilmenge der Links. Weiterhin ist es wünschenswert, wenn man auch Wissen über den Nutzer $u \in User$ zur Verfügung hätte, u.a. Hobbys, Beruf oder auch in der Vergangenheit besuchte Webseiten.

Die Komplexität der Funktion, welche sich z.B. durch sich ändernde Interessen des Nutzers ergibt, versucht man durch Annahmen zu vereinfachen. So unterstellt WebWatcher z.B. ein konstantes Interesse des Nutzers, und gibt ihm nur am Anfang der Tour die Möglichkeit, sein Interesse mitzuteilen.

Die approximierte vereinfachte Funktion $UserChoice$ ergibt sich wie folgt:

$$UserChoice?: Page \times Interest \rightarrow 2^{Links(p)} \quad (2)$$

Der Wertebereich – die Potenzmenge der Links auf Seite p – ergibt die Menge der Links, die ein Benutzer mit einem Interesse i auf der Seite $p \in Page$ wahrscheinlich folgt. Diese Funktion ist unabhängig vom Nutzer u.

Alternativ kann man die Formel auch folgendermaßen darstellen:

$$UserChoice? : Page \times Interest \times Link \rightarrow [0,1] \quad (3)$$

Dabei stellt die Funktion $UserChoice$ die Wahrscheinlichkeit dar, dass ein bestimmter Benutzer den Link bei einer gegebenen Seite mit einem gegebenen Interesse auswählt.

WebWatcher protokolliert automatisch die von ihm geführten Touren. Das System speichert für jede besuchte Seite die vom Nutzer verfolgten Links und das zu der früheren Tour gehörende Interesse. So erhält das System durch vorangegangene Touren sehr einfach Trainingsbeispiele für die oben angegebene Funktion $UserChoice$ (2), welche der Agent zum Erlernen eines Verhalten verwenden kann.

Wissensgewinnung durch Lernen aus vergangenen Touren

Die Funktion $UserChoice$ (2) muss für WebWatcher in folgendes Format transformiert werden:

$$UserChoice?_{p,l} : Interest \rightarrow \{+,-\} \quad (4)$$

Sie beschreibt jetzt für jede Seite p und jeden Link $l \in Links(p)$, ob ein Nutzer mit Interesse $i \in Interest$ dem Link folgt. Die Funktion wird also dichotomisiert.

Zur Wissensgewinnung greift WebWatcher jetzt auf die Protokolle der vergangenen Touren zurück. Zu jedem angeklickten Link wurde in einer Listenstruktur zusätzlich zu dem Linktext[6] das zu Anfang eingegebene Interesse der Tour der Linkbeschreibung zugefügt.

Zum Lernen werden all jene Beispiele aus vergangenen Touren herangezogen, bei denen der Nutzer auf Seite p Link l ausgewählt hat, die so genannten positiven Trainingsbeispiele. Nicht besuchte Links können bei dieser Methodik keine Hilfe leisten. Ziel des Algorithmus ist die Approximation der Funktion $UserChoice$ für alle Seiten p und Links l.

[6] Der von den HTML-Anchor-Tags (<a> …) eingeschlossene Text.

Zur Laufzeit bewertet WebWatcher bei der **Annotate** [7] genannten Methode die Links auf der anzuzeigenden Seite bezüglich des Nutzerinteresses. Dazu wird das Vektorraummodell von Salton [Sa91] verwendet. Man transformiert die Link- und Interessenbeschreibungen in Vektoren. Jede Vektorkomponente steht für einen Term aus der Indexierungssprache. Die Vektorkomponenten geben jeweils den Relevanzgrad des Wortes für den Link bzw. das Interesse an. Diese Gewichtung wird mithilfe der $TF \cdot IDF$ Heuristik berechnet. Eine Vektorkomponente w_{dk} ergibt sich folgendermaßen:

$$w_{dk} = TF(w,d) \cdot IDF(w) \quad (5)$$

Hierbei steht $TF(w,d)$ für die Termfrequenz von Term w, also wie oft das Wort w in der Link- bzw. Interessenbeschreibung d vorkommt. $IDF(w)$ ist die inverse Dokumentfrequenz und wie folgt definiert:

$$IDF(w) = \log \frac{n}{DF(w)} \quad (6)$$

$DF(w)$ besagt, in wie vielen Beschreibungen das Wort w mindestens einmal vorkommt. Durch Verwendung von $IDF(w)$ versucht man, die Trennschärfe zwischen den Termen zu berücksichtigen.

Für jede einzelne Interessenbeschreibung eines Links (die durch vergangene Touren gesammelt wurden) inklusive des Linktextes wird ein Vektor anhand Formel (5) bestimmt. Nun berechnet man die Ähnlichkeit zwischen dem Anfragevektor[8] und den einzelnen Interessenvektoren vergangener Touren mit dem so genannten Kosinusmaß:

$$\cos(L(\vec{d}_1, \vec{d}_2)) = \frac{\vec{d}_1 \cdot \vec{d}_2}{\sqrt{\vec{d}_1 \cdot \vec{d}_1}\sqrt{\vec{d}_2 \cdot \vec{d}_2}} \quad (7)$$

$L(\vec{d}_1, \vec{d}_2)$ gibt dabei Winkel zwischen den Vektoren $\vec{d}_1$ und $\vec{d}_2$ an. Durch die Normierung im Nenner (Division durch die Länge der Vektoren) ist die Unabhängigkeit der Vektoren von der Dokumentlänge gegeben.

[7] to annotate (engl.) – mit Anmerkungen versehen; kommentieren
[8] der hier das Interesse des Nutzers darstellt

Für jeden Link sortiert WebWatcher die Kosinusähnlichkeiten der Beschreibungen des Links zur Anfrage. Der Wert der Funktion $UserChoice$ für den gerade betrachteten Link ergibt sich einfach durch Mittelwertbildung der fünf ähnlichsten Beschreibungen. Liegt dieser oberhalb eines Schwellenwertes, schlägt WebWatcher den Link vor und markiert ihn mit dem Augensymbol. Dabei beschränkt sich der Agent auf maximal drei Links.

Diese Methodik setzt voraus, dass der Nutzer am Anfang der Tour sein Interesse eingibt. Ohne diese Daten kann WebWatcher keine Links empfehlen.

Wissensgewinnung durch Analyse der Dokumente und der Hypertextstruktur

Wie man beim vorangegangenen Ansatz von WebWatcher sehen konnte, ist die Entscheidungsregel - welcher Link soll hervorgehoben werden – an Vergangenheitsbeispiele gebunden. Ohne Trainingsbeispiele ist es daher nicht möglich, sinnvolle Ergebnisse zu liefern.

Der folgende Ansatz besitzt diese Einschränkung nicht und kommt ohne Vergangenheitsbeispiele aus. Hierbei werden die für die Entscheidung notwendigen Daten aus dem Text und der Hypertextstruktur gewonnen. Ein Anlernen des Softwareagenten ist deshalb nicht notwendig. Dieser Ansatz namens **Reinforcement Learning** (RL; Verstärkendes Lernen) wurde erfolgreich bei der Navigation von Robotern [Th95] eingesetzt, die sich autonom durch einen Raum mit Hindernissen bewegen mussten. Auch gelang es damit erstmals, dem Computer das Brettspiel Backgammon [Te92] beizubringen.

Das Grundkonzept des Reinforcement Learning geht von einem gerichteten Grafen aus, dessen Knoten Zuständen und dessen Kanten Aktionen entsprechen. Von einem Startzustand ausgehend sucht der Agent eine optimale Tour durch den Grafen. Der Graf enthält „Belohnungen“ $R(s)$, die man erhält, sobald man einen Knoten mit Belohnung erreicht. Die optimale Tour beschreibt den Weg, der die maximale Belohnung verspricht.

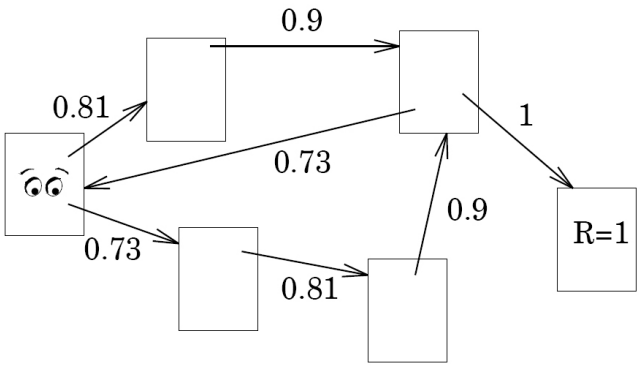


Abbildung 6: Ein Beispielgraf. Zustand rechts unten besitzt Belohnung von R=1. Die Kantenmarkierungen sind die Werte für Funktion $Q(s,a)$ (γ =0,9) der jeweiligen Zustände. Quelle: [Jo96]

Um diesen Weg zu finden, nutzt der Agent eine Bewertungsfunktion $Q(s,a)$. Diese beschreibt die zu erwartende Belohnung, wenn sich der Agent in Zustand s befindet, und Aktion a beschreitet und danach dem Rest der Tour optimal folgt.

$$Q(s,a) = R(s') + \sum_{i=1}^{\infty} \gamma^i R(s_i) \quad (8)$$

Die zu erwartende Belohnung im Falle, dass sich der Agent in Zustand s befindet und dort Aktion a ausführt, ergibt sich aus der Belohnung des Nachfolgezustands s' von s addiert zu den durch den Discount-Faktor γ gewichteten Belohnungen auf dem optimalen Restweg durch den Grafen. Der Discount-Faktor, der zwischen 0 und 1 liegen darf, erlaubt es, in dieser Formel potenzielle Belohnungen auf entfernten Knoten abzuschwächen. Je kleiner γ ist, desto weniger gehen die Belohnungen in die Bewertung ein.

In Abbildung 6 ist ein solcher Graf abgebildet. Darin erhält der Agent nur in dem Zustand rechts unten eine Belohnung von $R = 1$. In den sonstigen Zuständen erhält der Agent nichts. Die Kanten sind mit den Werten der Funktion $Q(s,a)$ beschriftet. Beispielsweise errechnet sich der Wert 0,73 unten links, wenn man vom Zustand mit dem Augensymbol nach rechts unten geht, folgendermaßen:

$$Q(s,a) = R(s') + \sum_{i=1}^{\infty} \gamma^i R(s_i) = 0 + \gamma^1 \cdot 0 + \gamma^2 \cdot 0 + \gamma^3 \cdot 1 = 0{,}73 \quad (9)$$

Setzt man $\gamma = 0{,}9$, ergibt sich der Wert von 0,73.

Mithilfe der dynamischen Programmierung kann man die Werte für $Q(s,a)$ approximieren. Beispielsweise verwendet man dazu den „synchronen Value-Iteration-Algorithmus“ von Bellman [Be57]. Dieser setzt voraus, dass die Discount-Faktoren γ kleiner 1 und die Belohnungen $R(s)$ endlich sind. Weiterhin ist per Annahme die Wahl einer Aktion in Zustand s unabhängig von den vorher besuchten Knoten.

Beim Start belegt man die Bewertungsfunktionen der Zustände im Grafen normalerweise mit $Q_0(s,a) := 0$. Mit folgender Formel berechnet man in mehreren Iterationen für alle Zustände s mit den möglichen Aktionen a neue Approximationen für $Q(s,a)$:

$$Q_{k+1}(s,a) := R(s') + \gamma \max_{a' \in Aktionen \cdot in \cdot s'} [Q_k(s',a'] \quad (10)$$

s' stellt den Nachfolgezustand von s dar, nachdem Aktion a beschritten worden ist. Somit ist $R(s')$ die Belohnung des Nachfolgezustands, zu der das diskontierte Maximum der Belohnung, die man von hier aus erwarten kann, addiert wird. Gemäß [Be87] konvergiert jedes $Q_\infty(s,a)$:

$$\forall s,a : \lim_{k \to \infty} Q_k(s,a) \to Q(s,a) \quad (11)$$

Dies bedeutet, dass man bei einer hohen Zahl k von Iterationen über die oben angegebene Formel (10) den gesuchten Wert für die Bewertungsfunktion $Q(s,a)$ erhält. Man bricht diesen Algorithmus ab, wenn die Differenz zwischen den $Q_{k+1}(s,a)$ und den $Q_k(s,a)$ unter eine Grenze ε fällt.

Reinforcement Learning und Tourenplanung im WWW

Der oben vorgestellte theoretische Ansatz der optimalen Tourenplanung wird jetzt auf das von WebWatcher zu lösende Problem angewandt. Das World Wide Web lässt sich als gerichteten Grafen darstellen. Die einzelnen Webseiten entsprechen den Knoten, und die Kanten den Hyperlinks, die semantische Verbindungen zwischen den Seiten herstellen. Das Optimierungsproblem für den Agenten stellt sich als Suche nach der optimalen Tour durch das Geflecht der Links dar. Optimal bedeutet hierbei, dass man als Zielkriterium die maximale Informationsausbeute in Hinsicht auf das Nutzerinteresse setzt.

Bewertungsfunktion für Links

WebWatcher benötigt eine Bewertungsfunktion, anhand derer er entscheiden kann, welche Hyperlinks der gerade besuchten Seite für das aktuelle Interesse des Nutzers relevant sind, also Informationen enthalten, die für den Nutzer wichtig sind. Im Gegensatz zur allgemeinen Einführung von Reinforcement Learning in Kapitel 0 gibt es für einen Knoten, also eine Webseite, mehrere Bewertungsfunktionen. Für jedes Wort w der Indexierungssprache, muss eine Funktion $Q_w(s,a)$ gelernt werden.

Um die Belohnung für ein konkretes Wort zu berechnen, wird das Vektorraummodell von Salton [Sa91] verwendet. Dabei werden Dokumente mit einem Vektors, dessen Vektorkomponenten für einen Term im Dokument stehen, beschrieben. Die Vektorkomponenten geben jeweils den Relevanzgrad des Wortes im Dokument nach der unten beschriebenen Heuristik an und entsprechen gleichzeitig den Belohnungen $R_w(s)$.

Man verwendet die $TF \cdot IDF$-Gewichtung des Vektorraummodells:

$$R_w(s) = \frac{TF(w,s) \cdot IDF(w)}{\sqrt{\sum_{w'} (TF(w',s) \cdot IDF(w'))^2}} \quad (12)$$

$TF(w,s)$ steht für die Termfrequenz des Terms w, also wie oft das Wort w im Dokument/Zustand s vorkommt. $IDF(w)$ ist die inverse Dokumentfrequenz und wie folgt definiert:

$$IDF(w) = \log \frac{n}{DF(w)} \quad (13)$$

$DF(w)$ gibt an, in wie vielen Dokumenten das Wort w mindestens einmal vorkommt. Durch Verwendung von $IDF(w)$ versucht man, die Trennschärfe zwischen den Termen zu berücksichtigen. Der Nenner von Formel (12) dient der Normierung des Vektors auf 1.

Wichtig ist hierbei, dass sich die Gewichtungen allein aus dem Dokument ergeben, und man keine Historie des Nutzer- bzw. Surfverhaltens benötigt[9].

Die Funktion $Q_w(s,a)$ für ein konkretes Wort w wird darauf trainiert, dass die diskontierten $TF \cdot IDF$-Belohnungen beim Verfolgen einer Kette von Hyperlinks maximal werden.

[9] vgl. Annahme der dynamischen Programmierung.

Approximation der Bewertungsfunktion bei unbekannten Links

Kennt WebWatcher die Seiten, auf die die Hyperlinks zeigen, kann er die Bewertungsfunktionen $Q_w(s,a)$ exakt errechnen. Bedingt durch die inzwischen erreichte Größe des World Wide Web ist es aber beinahe unmöglich, alle möglichen Seiten zu indizieren. Da Nutzer von WebWatcher sehr wahrscheinlich über die Grenzen des für WebWatcher bekannten Gebiets surfen möchten, benötigt der Agent eine Approximation zur Bewertung der ihm noch unbekannten Links.

[Mi97] schlägt vor, einen k-Nearest-Neighbor-Funktionsapproximator zum näherungsweisen Bestimmen der Funktionswerte $Q_w(s,a)$ zu verwenden.

$$\tilde{Q}(s,a) = \frac{1}{k}\sum_{i=1}^{k} sim((s,a),(s_i,a_i)) \cdot Q(s_i,a_i) \quad (14)$$

Die Ähnlichkeit zwischen zwei Seiten und zwei Hyperlinks ist folgendermaßen definiert.

$$sim((s_1,a_1),(s_2,a_2)) = \frac{1}{3}[\cos\angle(\vec{s}_1,\vec{s}_2)) + 2\cdot\cos(\angle(\vec{a}_1,\vec{a}_2)] \quad (15)$$

Man berechnet jetzt die Ähnlichkeit zwischen einem unbekannten Hyperlink und einem auf einer schon bekannten Seite. Der approximierte Wert für $Q_w(s,a)$ wird als gewichteter Mittelwert über die k ähnlichsten Links und deren Webseite mit (14) berechnet. $L(\vec{s}_1,\vec{s}_2)$ gibt dabei Winkel zwischen den Vektoren $\vec{s}_1$ und $\vec{s}_2$ an.

Die Repräsentation der Seiten erfolgt wieder über das Vektorraummodell, wobei jetzt allein der Titel der Seite[10] ausschlaggebend ist. Der Vektor eines Links ergibt sich nach wie vor aus dem Linktext[11]. Die Gewichtung in Formel (15) unterstellt heuristisch, dass die Ähnlichkeit der Links wichtiger ist, als die Seitenbeschreibung im Titel.

Entscheidungsregel

Soll WebWatcher einem Nutzer Hyperlinks empfehlen, sind zwei Fälle zu unterscheiden. Umfasst das Nutzerinteresse nur einen Term w, besteht die einfache Aufgabe für WebWatcher, die Funktionswerte $Q_w(s,a)$ für alle verlinkten Seiten von Zustand s zu sortieren, und die k (meist 3) am höchsten bewerteten Seiten für den Nutzer zu markieren. Dazu kommt noch eine „unmittelbare Belohnung“, falls das Wort w im Linktext vorkommt.

---

[10] Text zwischen HTML-Tags <title> … </title>
[11] Text zwischen HTML-Tags <a> …

Hat der Nutzer mehrere Worte $w_1,\ldots,w_n$ für sein Interesse eingegeben, ergibt sich die Kennzahl für jede verlinkte Seite aus der Summe der Q_w-Werte. Durch Sortierung erhält man wieder eine Rangfolge von Links, aus der man die k Größten auswählt und hervorhebt.

Vergleich der Lernmethoden

Die folgende Auswertung gemäß [JD96] beruht auf 1777 Touren, die WebWatcher zwischen dem 2. August 1995 und dem 4. März 1996 absolviert hat. Bei jeder dieser Touren hat der Nutzer ein Interesse eingegeben und mindestens vier Hyperlinks verfolgt. In der folgenden Auswertung werden neben den oben vorgestellten Methoden Annotate und Reinforcement Learning drei weitere verglichen:

- **Random**
  WebWatcher wählt zufällig Hyperlinks der Seite aus. Diese Strategie soll eine untere Schranke definieren.

- **Popularity**
  Auch diese Strategie kommt ohne das Wissen über das Interesse des Nutzers aus. Der Agent zählt, wie oft Links in der Vergangenheit angeklickt worden sind, und empfiehlt dann die populärsten.

- **Match**
  Die Strategie kommt der von Annotate nahe, allerdings wird jetzt nur der reine Linktext[12] mit dem Benutzerinteresse gemäß $TF \cdot IDF$ Heuristik und dem Kosinusmaß verglichen.

Die Bewertungsfunktion $Q(s,a)$ der Seiten bei der Reinforcement Learning Methode besteht aus der diskontierten Summe der Belohnungen und der oben erwähnten „unmittelbaren Belohnung“. Diese entspricht dem Ähnlichkeitswert der Match Methode.

Es wurden fünf verschiedene Test / Training – Splits verwendet und ausgewertet. Die Ergebnisse in Tabelle 1 sind daher als Mittelwert (mittels Mikrobewertung) über die fünf verschiedenen Experimente zu verstehen.

[12] Der von den HTML-Anchor-Tags (<a> …) eingeschlossene Text.

	Accuracy (auf allen Seiten)	Accuracy (nur bekannte Seiten)	Accuracy (nur unbekannte Seiten)
Random	31,6%	23,8%	39,7%
Popularity	41,8%	43,7%	39,7%
Match	40,3%	33,8%	46,0%
Annotate	42,0%	37,0 %	46,0%
RL	44,5%	41,5%	47,8%

Tabelle 1: Ergebnisse für fünf verschiedene Lernmethoden

Accuracy beschreibt den Anteil der Links, die der Nutzer verfolgt hat, wenn diese von WebWatcher vorgeschlagen wurden. Bei diesem Experiment sollten die Lernmethoden immer Links vorschlagen, egal wie überzeugt WebWatcher von ihnen war.

Die ersten Werte geben die Accuracy über alle Seiten im Testset an. Es ist zu erkennen, dass Reinforcement Learning signifikant besser ist, als die vier anderen Methoden. Die beiden anderen Spalten geben die Accuracy bei bekannten bzw. unbekannten Seiten an. Bekannt ist eine Seite, wenn der Benutzer einem empfohlenen Link während einer Trainingstour gefolgt ist. Bei den unbekannten Seiten fällt auf, dass die Accuracy bei fast allen Methoden höher ist, als bei bekannten Seiten. Erklären lässt sich dies damit, dass die Anzahl der Links auf bekannten Seiten im Durchschnitt höher war, als bei unbekannten Seiten.

Der Ansatz „Random“ stellt eine untere Schranke für den Agenten dar, da dabei zufällig Links markiert werden. Zu beachten ist weiterhin, dass die Accuracy durch empfohlene Links beeinflusst wird, da der Nutzer davon ausgeht, dass sich WebWatcher schon „auskennen“ wird.

Vergleich mit Ergebnissen menschlicher Experten

Um die Testergebnisse besser einschätzen zu können, wurde die Aufgabe, die normalerweise WebWatcher übernimmt, in einem Experiment [Jo97] auf menschliche Experten übertragen. Die Aufgabe bestand darin, für ein gegebenes Interesse drei Hyperlinks aus insgesamt 18 auf der SCS-Front-Door-Seite zu benennen. Acht Experten wurden jeweils 15 Beispiele vorgelegt, die sie bearbeiten mussten.

	Accuracy
Menschen	**47,5%**
Annotate	42,9%
Random	22,4%

Tabelle 2: Ergebnisse der Methoden des maschinellen Lernens im Vergleich mit der Leistung von Menschen

Bei der Interpretation der Ergebnisse muss berücksichtigt werden, dass die Probanden alle Studenten bzw. Professoren der Informatik waren, und sie die Webseite kannten. Auch ist der Stichprobenumfang sehr klein.

Wie in Tabelle 2 zu sehen, erreichten die Versuchspersonen eine Accuracy von 47,5%. Die Annotate-Methode liegt mit 42,9% signifikant darunter. Nichtsdestotrotz scheint die Aufgabe des Empfehlens von Webseiten für Menschen trotz ihrem größeren Sprach- und Wissenshorizont schwierig zu sein.

Es erscheint daher sinnvoll, weitere Informationen über den Kontext des Nutzers zu sammeln, da die anfangs eingegebene Interessenbeschreibung anscheinend nicht ausreicht, um Links sicher vorhersagen zu können. Möglichkeiten sind zum Beispiel Formen von Relevance Feedback und die Analyse der auf der Tour schon betrachteten Seiten.

Ausblick und verwandte Projekte

Neben WebWatcher gibt es weitere Projekte, die sich unter dem Fokus des Empfehlens von Webseiten mit Lösungsvarianten beschäftigen.

Zu nennen wäre das Projekt Letizia von Henry Lieberman. In [Li97] schlägt er einen „Autonomous Interface Agent“ vor, der lokal auf dem System des Nutzers arbeitet. Im Unterschied zu WebWatcher findet hier keine Veränderung der ursprünglichen Webseite statt, sondern die Empfehlungen werden in zwei weiteren Browserfenstern eingeblendet. Autonome Agenten arbeiten parallel und unabhängig vom Nutzer. Der Term „Autonomous Interface Agent“ beschreibt ein Agentensystem, in dem der Agent durch einzelne Aktionen des Nutzers selbstständig über sein Verhalten entscheidet, und dieses über ein Interface, z.B. eine Benutzeroberfläche, ein Fenster, etc. kommuniziert. Im Falle von Letizia registriert der Agent die aufgerufenen Webseiten und analysiert diese. Weiterhin durchsucht Letizia die Webseiten in der näheren „Umgebung“[13] mittels einer Breitensuche. Die Ergebnisse präsentiert der Agent dann in einem unabhängigen Fenster.

[13] Seiten, die von der aktuellen Seite über Hyperlinks erreichbar sind.

Das System Alexa der Firma Alexa, Inc.[14] (einem Amazon.com Unternehmen) wird unter anderem in den Webbrowsern Mozilla bzw. Netscape verwendet. Dort findet man in der „Sidebar“ die Schaltfläche „What’s related?“. Beim Klick auf dieses Register werden in der Sidebar Seiten angezeigt, die der gerade betrachteten, ähnlich sind. Da dies ein kommerzielles System ist, sind die Informationen über die Funktionsweise sehr knapp. Nur in der „What’s Related? FAQ“ [Wr03] gibt es Hinweise zur Implementierung. Alexa benutzt Crawler, um das Internet zu durchsuchen, zu archivieren und zu katalogisieren. Aus diesen Daten wird ein Index erstellt. Durch Data Mining Techniken werden Beziehungen zwischen Seiten extrahiert. Dazu werden die besuchten Webseiten und Links an Alexa übermittelt. Will der Nutzer verwandte Seiten betrachten, sendet der Browser die aktuelle URL an Alexa, die mit bis zu 10 Links auf verwandte Seiten, so vorhanden, antwortet.

LIRA [Bs95] schlägt Webseiten vor, die den Nutzer interessieren könnten. Dazu bewertet es Webseiten mithilfe des Vektorraummodells. Dabei wird eine Variation der $TF \cdot IDF$-Formel [SB87] verwendet. Für jeden User wird ein Vektor $\vec{M}$ verwaltet, der sein Interesse formalisiert. Um dem System sein Interesse bekannt zu machen, unterstützt dieses System Relevance Feedback nach Roccio [Ro71]. Der Nutzer kann die ihm empfohlenen Webseiten in eine Skala von [-5; +5] einordnen. LIRA verändert dann die Gewichtung der einzelnen Vektorkomponenten von $\vec{M}$ um das Interesse des Nutzers genauer zu bestimmen. Dadurch können die Empfehlungen von Webseiten immer präziser werden.

Zusammenfassend lässt sich feststellen, dass die Leistung der Agenten fast gleichauf mit der Leistung von Menschen liegt. So ist die angebotene Hilfe beim „Browsing“ durch das World Wide Web eine willkommene Unterstützung, wenn man als Nutzer auf einer zielgerichteten Informationssuche ist. So hat man neben den klassischen Suchmaschinen, die zwar eine mitunter große Ergebnismenge liefern, deren Ergebnisse oft mit viel „Noise“, irrelevanten Dokumenten, verwässert werden und Webkatalogen eine weitere Alternative, die gute Ergebnisse verspricht.

Verbesserungsmöglichkeiten sind durch eine erweiterte Analyse von Webseiten-merkmalen und der genaueren Bestimmung des Nutzerinteresses zu erreichen. Die bisherigen Systeme arbeiten auf der reinen Wortebene, und gehen davon aus, dass ein Wortvorkommen den Relevanzgrad für dieses Wort im Dokument beschreibt. Abhängigkeiten zwischen Wörtern werden ignoriert.

[14] http://www.alexa.com

Durch Initiativen wie das Semantic Web des W3C[15] könnte dieser „Missstand“ vermindert werden. Im Semantic Web sollen Webseiten neben ihrem HTML-Erscheinungsbild ihre Semantik in einer speziellen Ontologiesprache [IX03], z.B. Web Ontology Language (OWL) definieren. Ontologien beschreiben Konzepte einer abgegrenzten Anwendungsdomäne. Dazu werden kontrollierte Vokabulare verwendet. Zusätzlich bieten Ontologien Relationen zwischen den Konzepten an, ähnlich wie bei Thesauri. Schlussendlich können diese Ontologien von Inferenzsystemem genutzt werden, um neues Wissen aus altem zu schließen.

Dadurch soll es Agenten ermöglicht werden, auf den „Kern“ der Dokumente zuzugreifen und dadurch ein genaueres Bild des Inhalts zu erlangen. Im Zuge dessen kann man Agenten erstellen, die dieses Zusatzwissen dazu verwenden, dem Nutzer Webseiten zu empfehlen.

[15] http://www.w3.org/2001/sw/

Literaturverzeichnis

[Be57] Bellman, R.: Dynamic Programming, Princeton University Press, 1957

[Be87] Bertsekas, D.: Dynamic Programming: Deterministic and Stochastic Models, Prentice-Hall, Englewood Cliffs, NJ, 1987

[Bs95] Balabanović, M., Shoham, Y., Learning Information Retrieval Agents: Experiments with Automated Web Browsing, Proceedings of the {AAAI} Spring Symposium on Information Gathering from Heterogenous, Distributed Resources, 1995, 13-18

[Go95] Gordon, G.: Stable Function Approximation in Dynamic Programming, Proceedings of the Twelfth International Conference on Machine Learning, 1995, 261-268

[IX03] Ziegler, C., Surfende Maschinen: Web Ontology Language (OWL): Vokabulare fürs Web, iX 12/2003

[JD96] Joachims, T., Freitag, D., Mitchell, T.: WebWatcher A Tour Guide for the World Wide Web

[Jo96] Joachims, T.: Einsatz eines intelligenten, lernenden Agenten für das World Wide Web, Fachbereich Informatik, Universität Dortmund, Diplomarbeit, 1996.

[Jo97] Joachims, T., Freitag, D., Mitchell, T.: WebWatcher A Tour Guide for the World Wide Web

[Li97] Lieberman, H.: Autonomous Interface Agents, Proceedings of the {ACM} Conference on Computers and Human Interface, {CHI}-97

[Me92] Meadow, C.: Text Information Retrieval Systems, New York Academic Press, 1992.

[Mi97] Mitchell, T.: Machine Learning, McGraw-Hill, 1997

[Ro71] Rocchio, Jr., J., Relevance Feedback in Information Retrieval, The Smart System – Experiments in Automatic Document Processing, Englewood Cliffs, NJ: Prentice Hall Inc 313-323.

[Sa91] Salton, G.: Developments in Automatic Text Retrieval, Science, 1991, 253:974-979

[SB87] Salton, G., Buckley, C.: Term Weighting Approaches in Automatic Text Retrieval, Technical Report 87-881, Cornell University, Department of Computer Science

[Te92] Tesauro, G.: Practical Issues in Temporal Difference Learning. Machine Learning Journal, 1992, 8:257-277.

[Th95] Thrun, S.: An Approach to Learning Mobile Robot Navigation, Robotics and Autonomous Systems, 15:301-319.

[Wr03] What's Related FAQ, http://wp.netscape.com/escapes/related/faq.html

[WW97] Projekt WebWatcher
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/wwdemo.html

Empfehlung für Webseiten - Letizia

Martin Pleßmann
Mjdragan@t-online.de

Abstract: Im Bereich der Software und der Robotik ist der „Agent“ ein populärer Begriff zur Bezeichnung von „intelligenten Programmen“. Unter dem Namen „agenten-orientierte Techniken“ entstehen neue Methoden und Strukturen, bei denen die Künstliche Intelligenz wesentliche Beiträge zu liefern hat. Zunächst wird der Begriff des „Agenten“ untersucht und dessen Eigenschaften herausgearbeitet. Im zweiten Teil wird der am *Media Laboratory* des *Massachusetts Institute of Technology* (MIT) von Henry Lieberman entwickelte autonome Interface Agent Letizia vorgestellt.

1 Einleitung

Das Gebiet der Software-Agenten ist seit einigen Jahren Bestandteil der Forschungsarbeit in den Bereichen der Künstlichen Intelligenz und des Information Retrieval. Da in diesem dynamischen Gebiet noch keine allgemeingültigen Definitionen existieren, werden zunächst die Eigenschaften und besonderen Merkmale von den sogenannten Agenten herausgearbeitet. Der Spielraum der praktischen Anwendungsmöglichkeiten von Software-Agenten ist extrem groß und somit nicht vollständig aufzeigbar. Im Prinzip kann jede Information, die aus der Umwelt auf ein System einwirkt, von einem Agenten aufgenommen ,verarbeitet und in Entscheidungen umgewandelt werden. Am Beispiel des autonomen Interface Agenten Letizia, entwickelt von Henry Lieberman am Media Laboratory des *Massachusetts Institute of Technololgy* (MIT), wird im zweiten Teil der Arbeit eine Möglichkeit vorgestellt, wie Agenten den Menschen in seiner Arbeit oder sonstigen Tätigkeiten unterstützen und entlasten können.

2 Ein Software-Agent: Was ist das?

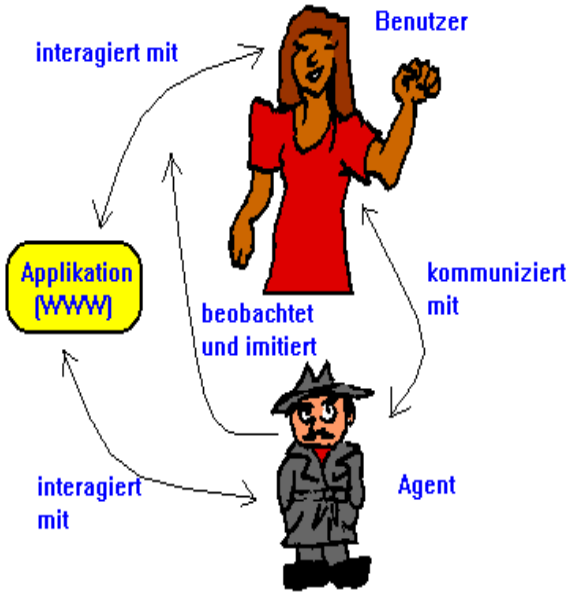


Abbildung 1: Metapher des intelligenten Agenten

Abbildung 1 zeigt in einfacher Weise, was unter einem Agenten im Sinne der Informatik zu verstehen ist. Genau wie ein Agent als reale Person, ist es Aufgabe eines SW-Agenten, für seinen Auftraggeber, hier als Benutzer bezeichnet, Dienste zu verrichten. Durch die Kommunikation zwischen Benutzer und Agent und die Beobachtung des Verhaltens des Benutzers kann der Agent Rückschlüsse auf die Wünsche und Bedürfnisse des Benutzers schließen. Anstelle des Benutzers versucht nun der Agent das Verhalten der auftraggebenden Person zu imitieren und für sie mit den entsprechenden Systemen (im Beispiel das WWW) zu interagieren. Nach dieser sehr einfach gehaltenen Darstellung eines Agenten können nun schon wesentliche Merkmale aufgezeigt werden, die einen Agenten von einem herkömmlichen Programm unterscheiden.

- **Delegation:** Hier liegt die Grundaufgabe eines Agenten, wie sie in jedem handelsüblichen Lexikon nachzulesen ist: „*Ein Agent ist jeder im Auftrag oder Interesse eines anderen Tätige.*“ Anders ausgedrückt, ist also ein Agent eine Software, die etwas für sie erledigt.

- **Kommunikation:** Dies stellt einen wichtigen Punkt dar; um sein Ziel zu erreichen, muss der Agent mit seinem Benutzer, seiner Umgebung und eventuell auch mit anderen Agenten kommunizieren können. Dies wird etwa definiert in [Lingnau95]: „*...ein Agent ist ein Computerprogramm, dessen Zweck es ist, den Anwender bei der Lösung von Aufgaben zu unterstützen. Er besitzt dazu einen eigenen, persistenten Zustand und kommuniziert mit dem Besitzer, anderen Agenten und seinem Umfeld im Allgemeinen.*“

- **Perzeption:** Agenten müssen ihr Umfeld, dessen Teil sie sind, wahrnehmen. Eine Definition nach [GRS 00] beschreibt dies recht deutlich mit: „*Ein Software-Agent ist ein längerfristig arbeitendes Programm, dessen Arbei als eigenständiges Erledigen von Aufträgen oder Verfolgen von Zielen in Interaktion mit einer Umwelt beschrieben werden kann.*“

2.1 Eigenschaften von Agenten

Folgende Begriffe spielen bei der Betrachtung von Agenten allgemein eine Rolle, wobei aber nicht jeder Agent alle diese Eigenschaften besitzen muss. Es handelt sich bei diesen Begriffen eher um “Dimensionen“, mit denen man im konkreten Fall Agenten genauer bestimmen kann. Die Auflistung der Eigenschaften ist aus [GRS 00] übernommen.

Andauernde Verfügbarkeit / Aktivität: Agenten sind über einen längeren Zeitraum hinweg ansprechbar (Verkaufs-Agent) und nehmen Aufträge von Nutzern oder anderen Agenten entgegen. Sie können eigenständig aktiv werden (Aktualisierung beim Verkaufs-Agenten)

Interaktion mit einer Umwelt: Agenten nehmen Informationen aus ihrer auf (Aufträge, Erfassen von Situationen, Kontrolle ihrer Aktionen). Sie agieren in ihrer Umwelt, um sie auftragsgemäß zu beeinflussen.

Situiertheit: Auch hier geht es um die Einbettung von Agenten in eine Umwelt. Es wird der Aspekt betont, dass komplexes Agentenverhalten auch als Resultat von direkten Reaktionen auf Umwelteinflüsse erzeugbar ist („emergentes Verhalten“), was zu einfacheren Architekturen führen kann.

Eigenständigkeit im Handeln (Autonomie): Agenten unterliegen keiner (unmittelbaren) Steuerung und Kontrolle durch einen Nutzer. Sie handeln eigenständig, aber „im Sinne ihrer Auftraggeber“. Die Auswahl/Planung ihrer Handlungen kann (aber muss nicht) nach sehr komplexen Methoden erfolgen.

Reaktivität: Im engeren Sinne wird darunter das unmittelbare Reagieren auf Umweltereignisse verstanden. Im allgemeinen betrifft es die Interaktion mit der Umwelt insgesamt.

Zielgerichtetheit: Agenten verfolgen Ziele bzw. Aufträge, die angepasstes Handeln über lange Zeiträume hinweg erfordern können.

Pro-Aktivität: Der Begriff ist verwandt mit Zielgerichtetheit (und wird teilweise gleichwertig benutzt). Eine spezielle Betonung liegt dabei auf der „Eigeninitiative" des Agenten.

Deliberatives Verhalten: Deliberation bezeichnet die explizit modellierte Auswahl von Zielen bzw. Absichten. Auch dieser Begriff wir als Gegensatz zu reaktivem Verhalten gebraucht.

Intelligenz: Der Begriff kann im Sinne analoger Erklärungen des Begriffs „Künstliche Intelligenz" so interpretiert werden: Agenten vollführen Handlungen, die beim Menschen als intelligent gelten würden.

Rationalität: Agenten treffen sinnvoll Entscheidungen in angemessener Zeit auch unter dem Aspekt beschränkt verfügbarer eigener Ressourcen.

Lernfähigkeit: Agenten können ihr Verhalten an die Umwelt anpassen, indem sie zum Beispiel Fähigkeiten oder Entscheidungsprozessen geeignet variieren.

Mobilität: Agenten können eigenständig auf andere Plattformen (Rechner) migrieren. Dabei wird das Agenten-Programm mit seinen aktuellen Daten übertragen und setzt seine Arbeit dort fort mit der Möglichkeit des Zugriffs auf lokale Ressourcen.

Kooperation: Agenten arbeiten zusammen mit Menschen und anderen Agenten.

Wohlwollen (Benevolenz): Agenten führen Aufträge nach Möglichkeit im Sinne des Auftraggebers aus, wobei auch andere Agenten Auftraggeber sein können.

Soziales Verhalten: Agenten halten sich bei ihrer Arbeit mit anderen Agenten (und Menschen) an vorgegebene Regeln und Normen. Sie bilden Gruppen oder Koalitionen. Vorbild sind soziale Strukturen der Menschen.

Emotionales Verhalten: Emotionen sollen einerseits das Verhalten steuern (sie treten damit neben andere Begriffe wie Ziele, Absichten usw.) und sie sollen die Interaktion mit Menschen verbessern.

Glaubwürdigkeit: Die Erscheinungsform (z.B. Gestik animierter Agenten) soll sich in glaubhafter Übereinstimmung mit Aktionen befinden. Agenten sollen als Individuen mit eigenen Zielen, Bedürfnissen und Emotionen erscheinen.

Wie oben gesagt, müssen Agenten nicht alle diese Eigenschaften erfüllen. Man kann aber anhand dieser Dimensionen die jeweiligen Eigenschaften spezifizieren, zum Beispiel für unterschiedliche Varianten eines Verkaufs-Agenten: in seiner einfachen Form ist er reaktiv, in komplexeren Formen dagegen deliberativ.

2.2 Grundlegende Struktur von Agenten

Aus den oben aufgeführten Punkten kann man erkennen, dass die zugrundeliegende Struktur von Agenten ein andauernd autonomes Agieren/Reagieren auf Einflüsse der Umwelt beschreibt. Hieraus kann man die Arbeit eines Agenten als eine zyklische Folge von Arbeitsschritten auffassen. Görz, Rollinger und Schneeberger gliedern diese Ablaufschritte in einen 3-Phasen-Zyklus, der folgendermaßen beschreibbar ist.

Phase der Informationsaufnahme:

> Der Agent nimmt Informationen aus der Umwelt auf. Er nimmt Nachrichten und Aufträge entgegen. Er beobachtet die Wirkungen seiner Handlungen.

Phase der Wissensverarbeitung und Entscheidung:

> Der Agent aktualisiert sein Wissen mithilfe der eingegangenen Informationen. Er analysiert und bewertet die aktuelle Situation, die neuen Aufträge und den Fortschritt bereits begonnener Handlungen. Er trifft Entscheidungen über seine unmittelbaren und zukünftigen Aktionen.

Phase der Aktionsausführung:

> Die anstehenden Aktionen (z.B. Versenden von Nachrichten, Berechnungen, Präsentation von Ergebnissen) werden ausgeführt.

Prinzipiell ist es möglich, dass diese drei Phasen parallel ablaufen können. Software-Agenten realisieren dagegen meist nur eine quasi-nebenläufige Arbeit oder sogar eine streng zyklische Abfolge der drei Phasen (observation –thought – action – Zyklus).

Zunächst gibt es die von außen eingehenden Daten `S` aus einer Menge `sensoryinputs`. Am Beispiel eines Verkaufs-Agenten wäre `S` zum Beispiel eine Kundenanfrage. Die eingegangen Daten werden durch eine Funktion

```
Sense: sensoryinputs ⟶ perceptions
```

zu einer „Wahrnehmung“ `W Element perceptions` aufbereitet.

Im Verkaufs-Agenten wird hier die Kundenanfrage analysiert (z.B. Abgleich mit der Kundendatenbank, Umwandlung der Anfrage in eine interne Repräsentation). Anschließend folgt ein interner Schlußfolgerungsprozeß, in dem der Wissensstand über die Umwelt aktualisiert werden kann, und in dem Aktionen für die Ausführung von Aufträgen oder zum Erreichen von Zielen geplant werden kann. Allgemein ausgedrückt wird der interne Zustand `Z element internalstates` des Agenten verändert. Ein Teil dieses Zustands betrifft die Annahmen über die Umwelt, der andere Teil betrifft die Verpflichtungen über zukünftige Handlungen. Verpflichtungen bezeichnen dabei allgemein die Festlegungen über das zukünftige Handeln, also auch Verpflichtungen des Agenten gegenüber sich selbst.

Im allgemeinen hängt der neue Zustand Z_{neu} sowohl von der Wahrnehmung `W` als auch vom vorherigen Zustand Z_{alt} ab. Der Schlußfolgerungsprozeß wird also von einer Funktion

```
thought: internalstates  x  perceptions ⟶ internalstates
```

beschrieben. In einem Verkaufs-Agenten kann der Zustand zum Beispiel Informationen über den bisherigen Verlauf des Dialogs, über bereits erfragte Fakten und über bereits geplante kommende Schritte enthalten. Durch den Schlußfolgerungsprozeß werden entsprechende Informationen gemäß der Wahrnehmung `W` aktualisiert, und es werden weitere Schritte geplant. Die im Zustand Z enthaltenen Verpflichtungen führen schließlich zu Aktionen `A element actions` gemäß einer Funktion

```
act : internalstates ⟶ actions
```

In verknappter Form stellt sich das Agieren eines Agenten damit wie folgt dar:

```
repeat
      W := sense(S);
      Zneu := thought(Zalt, P);
      A := act(Zneu);
Forever
```

Die interne Schlußfolgerungsphase `thought` wird bei komplexeren Agenten in weitere Phasen untergliedert. Hauptbestandteile sind dabei die Aktualisierung des Wissens über die Umwelt sowie die Festlegung auf Verpflichtungen zum Handeln. Die Phasen stehen in kausaler Abhängigkeit; bei Überschneidungen kann es folglich Konfusionen geben. Ein entsprechendes Zeitmanagement und die Verwaltung von Zuständen werden dann nötig.

Nach diesem kurzen Ausflug in die Theorie von Software-Agenten wird nun im folgenden Teil Letizia, ein autonomer Interface Agent in Funktion und Nutzen vorgestellt.

3 Letizia – Ein autonomer Interface Agent

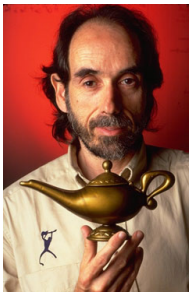


Abbildung 2: Henry Lieberman

Letizia ist ein am *Media Laboratory* des *Massachusetts Institute of Technology* von Henry Lieberman entwickelter intelligenter Agent (geschrieben in Macintosh Common Lisp), der den Benutzer speziell beim Browsing durch das WWW unterstützt. Während der User mit einem herkömmlichen Browser (Letizia arbeitet mit Mosaic oder Netscape) durch das Internet stöbert, beobachtet Letizia sein Verhalten, und versucht vorauszuahnen, an welchen Links der Nutzer das größte Interesse haben wird. Dies erreicht Letizia dadurch, dass sie selbständig die von der momentanen Position des Users ausgehenden Links weiterverfolgt und untersucht. Die so erzielten Ergebnisse werden in einem parallel zum Browser angezeigtem Fenster aufgelistet und dem Benutzer zur Verfügung gestellt. Der User kann nun selbst entscheiden, ob er den Empfehlungen von Letizia folgen will oder seinen eigenen Browsingprozeß fortsetzt.

Letizia wird als ein autonomer Interface Agent bezeichnet; was bedeutet autonom und um was für ein Interface geht es?

Ein autonomer Agent unterstützt den User in seinen Aktionen, ohne dass der User in irgendeiner Weise mit dem Agenten interagieren muss. Der Agent arbeitet selbständig an der Verfolgung und Bestimmung von Zielen und erledigen von Aufgaben.

Die Bezeichnung „Interface Agent“ bedeutet, dass sowohl User als auch Agent zur gleichen Zeit auf dem gleichen Medium operieren. Hierbei ist besonders hervorzuheben, dass der User trotz des gleichen Arbeitsraums in seinen Aktionen nicht vom Agenten gestört oder sogar unterbrochen werden darf. Der Agent hilft aktiv bei der Bedienung des Interfaces; im Fall von Letizia ist das Interface das Internet.

Die Definition von Agent:“ Ein Agent ist eine Person/ein Programm, dass im Auftrag eines Anderen arbeitet.“ kann im Fall von Letizia nicht wörtlich übernommen werden. Die Vorgehensweise von Letizia wird als *indirekte Manipulation* bezeichnet.

Dies bedeutet, dass der Agent zwar Aufgaben/Arbeiten für den User ausführt, dies aber ohne einen direkten Auftrag tut. Mit der Installation von Letizia gibt der User einmalig den Auftrag zur Unterstützung an den Agenten. Im weiteren Zeitablauf benötigt es keiner weiteren Aktivierung seitens des Benutzers; Letizia arbeitet völlig selbständig.

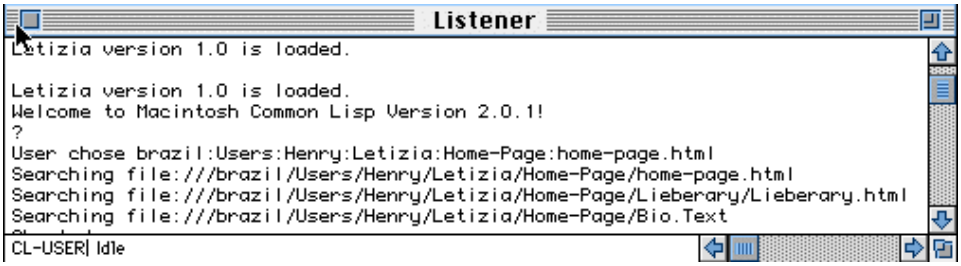


Abbildung 3: Installation von Letizia

3.1 Gleichzeitigkeit von Browsing und automatischer Suche

Letizia arbeitet nach dem Prinzip der Kooperation der zwei gleichberechtigten Partner User und Agent. Die kooperative Zusammenarbeit ermöglicht es, dass sowohl User als auch Agent ihre jeweiligen Stärken ausnutzen können, ohne sich gegenseitig zu behindern. Beide durchstöbern den gleichen, aus WWW-Dokumenten bestehenden, Suchraum.

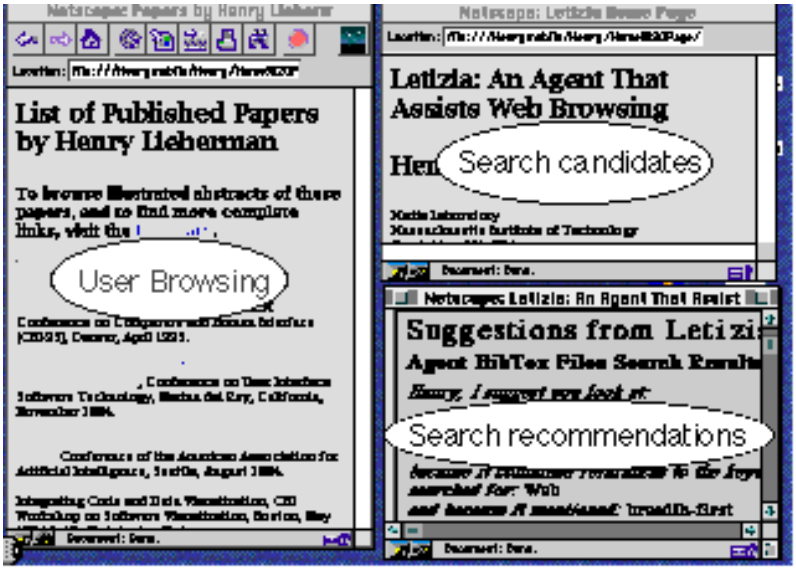


Abbildung 4: Screenshot: Letizia at work

Links: Aktuell betrachtete Seite des Users

Rechts oben: Aktuell von Letizia durchsuchte Links

Rechts unten: Aktuell von Letizia empfohlene Links

Der Unterschied zwischen der Suche des Users und der Suche von Letizia ist, dass der User eine verlässlichere Bewertung der Relevanz eines Dokumentes abgeben kann, als es ein Programm jemals zustandebringen wird. Letizia auf der anderen Seite vermag wesentlich schneller Alternativen aufzuspüren kann als der Benutzer, die eventuell von Interesse und Relevanz sein können. Laut Lieberman leitet Letizia aus dem Verhalten des Users in der Vergangenheit eine grobe Annäherung des gegenwärtigen Interesses ab. Sie muss dazu beurteilen können, welche Aktionen des Users für die Zukunft relevant sind und welche nicht. Dies soll ein Inferenzmechanismus leisten, mit dem Letizia ausgestattet ist. Trotz intensiver Nachforschungen ist es mir allerdings nicht gelungen, diesen Inferenzmechanismus ausfindig zu machen. Wenn später die genaue Funktionsweise von Letizia erläutert wird, wird schnell klar, dass der Anspruch auf künstliche Intelligenz von Letizia über die Verwendung von Inferenzmechanismen wohl eher ein frommer Wunsch des Entwicklers ist. Tatsache ist, dass in der Literatur keinerlei Mechanismen praktisch vorgestellt werden, die Letizia die Fähigkeit geben könnten, eine grobe Annäherung des zukünftigen Interesses des Users aus seinen Aktionen in der Vergangenheit abzuleiten. Parallel zum Browsingprozeß des Benutzers sucht also Letizia nach Links, welche für den Benutzer relevant sein können, und trägt diese in ihre Empfehlungsseite.

Wenn sich die Interessen des Users ändern (z.B. Wechsel von Sporthistorie zu Wirtschaftsprognose), werden die Empfehlungen automatisch an die neuen Interessen angepasst. Anders ausgedrückt vollzieht Letizia eine dynamische Anpassung der Empfehlungen an Veränderungen im Suchprozeß. Zu jedem Zeitpunkt kann sich der User, basierend auf dem jeweiligen Suchzustand, die Empfehlungsseiten anschauen. Dadurch, dass Letizia sucht, während der User ein Dokument liest, entfallen die lästigen Wartezeiten auf Suchergebnisse, die beim Einsatz von Suchmaschinen nicht zu vermeiden sind. Dies funktioniert natürlich nur, wenn der User lange genug bei einem Dokument verweilt.

Die in Abbildung 4 gezeigte Aufteilung des Bildschirms in drei Fenster ist bei Letizia nicht starr festgelegt. Das linke Fenster, wie oben schon dargestellt, ist das Browsing-Fenster des Users (in diesem Beispiel Netscape). Dieses Fenster ist logischerweise immer vorhanden, da es ja der Arbeitsbereich des Benutzers ist. Die anderen, von Letizia geöffneten Fenster auf der rechten Seite können vom User ignoriert werden, da sie keinen direkten Einfluss auf seinen Browsing-Prozeß haben. Allein die beiden Fenster auf der rechten Seite stehen unter der Kontrolle von Letizia. Je nach Vorliebe, kann z.B. das rechte obere Fenster, das die vom Agenten zu durchsuchenden Links anzeigt, auch ausgeblendet werden. Somit würden dem User nur die von Letizia als relevant erachteten Links im rechten unteren Fenster angezeigt. Es ist auch denkbar, dass mehrere Fenster von Letizia angezeigt werden. Dies wäre der Fall, wenn Letizia für mehrere User-Profile gleichzeitig nach relevanten Dokumenten Ausschau hält, und diese dann auch separat in verschiedenen Fenster auflistet.

3.2 Funktionsweise von Letizia

Letizia ist ein sogenannter *behavior based* Agent. Sie trifft ihre Entscheidungen also nicht auf vorher festgelegten, vorprogrammierten Regeln, sondern bezieht ihr Wissen durch Rückschlüsse aus dem aktuellen Verhalten des Benutzers. Auch wenn die dazu benötigten Inferenzmechanismen in meinen Nachforschungen nicht explizit in Letizia eingebunden sind, möchte ich sie der Vollständigkeit halber kurz darstellen.

Induktion: - Schließen vom Speziellen auf das Allgemeine

- Induktion erzeugt Hypothesen, nicht Wissen

Deduktion: - Spezialisierung eines allgemeinen Konzepts

- Nutzung von Vorwissen

Abduktion: - Ableiten einer Prämisse aus Hintergrundwissen und Beispielen

Analogien: - Abstraktion eines Problems

- Finden von Ähnlichkeiten

Die Funktionsweise von Letizia ist ein Mittelweg zwischen *information retrieval* und *information filtering*.

Information retrieval: Aktives Durchsuchen der Wissensbasis durch den User

Information filtering: User ist passives Ziel eines Informationsstroms

Grundlage der Wissensgewinnung und Arbeit für Letizia ist das inormation retrieval durch den User. Ausgehend von den dadurch erhaltenen Daten betreibt Letizia seinerseits eine Suche nach Dokumenten, die eine gewisse Relevanz für den User haben könnten. Die dem User angebotenen Seiten wurden also von Letizia vorher aus der großen Menge der im Netz verfügbaren Dokumente ausgesiebt (information filtering).

3.3 Die Modellierung des Browsingprozesses des Benutzers

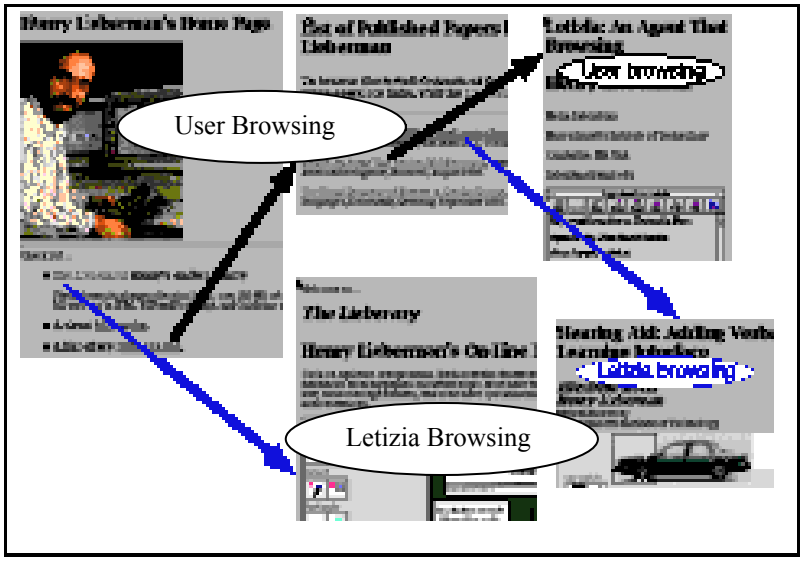
Der Browsingprozeß des Users besteht typischerweise darin, dass er ein Dokument nach für ihn interessanten Themen durchsucht, entscheidet welchen Links er folgen will oder das gegenwärtige Dokument der Hotlist des Browsers hinzufügt. Außerdem kann er zu vorher bereits besuchten Seiten zurückkehren oder eine bestimmte Seite aus der Hotlist auswählen. Das Ziel des Agenten ist es nun, sich in diesen Prozeß hineinzuversetzen und solche Aktionen auszuführen, von denen anzunehmen ist, dass der User sie auch so ausgeführt hätte. Letizia führt Aktionen aus, die mit dem bisherigen Verhaltendes Benutzers am besten in Einklang stehen, bzw. die eine hohe Wahrscheinlichkeit haben, dass sie vom User auch als nächstes ausgeführt würden.

Der Hauptnutzen von Letizia ist die Ausnutzung der Zeit, in der ein Benutzer damit beschäftigt ist, ein Dokument auszuwerten. Letizia agiert sozusagen als Handlanger oder Zuträger von Informationen. Der User kann sich somit ganz auf die Auswertung der Dokumente konzentrieren. Der Agent leistet quasi die „niedere“ Arbeit der Informationssuche. Dies erscheint auch angemessen, da es, wie oben schon kurz erwähnt, niemals einen Agenten geben kann, der die Relevanz eines Dokuments zuverlässiger beurteilen kann als der User selbst. Neben dem Beispiel des Zutragen von wahrscheinlich relevanten Informationen, kann das Aufspüren von „dead-ends“ als ein weiteres Merkmal für „niedere“ Arbeit angesehen werden. Dies kann Letizia perfekt erledigen und so dem User Zeit sparen.

Falls der User versucht, einen solchen Sackgassen-Link zu verfolgen, muss Letizia allerdings von sich aus explizit in den Browsingprozeß eingreifen und dem Benutzer einen entsprechenden Hinweis darauf geben. Allerdings übernimmt Letizia niemals die Kontrolle über die Informationssuche (diese bleibt ständig beim Informationssuchenden), sondern ist nur dafür verantwortlich, Hinweise in Situationen zu geben, in denen der User selbst unsicher ist und daher explizit nach Empfehlungen fragt. In Situationen, in denen man nicht weiß in welche Richtung man weitersuchen soll, sind die Empfehlungen von Letizia immer noch besser, als wenn man gar keine Hinweise hat.

Hierauf kann eine weitere hilfreiche Funktion/Nutzen von Letizia dargestellt werden. Das sogenannte „Lost-in-space-Szenario“ von Usern, die sich über Links so tief in das Internet eingegraben haben, dass sie nicht mehr zum Ausgangspunkt ihrer Suche zurückfinden können. Anhand dieses Beispiels können die verschiedenen Suchstrategien von Benutzer und Agent aufgezeigt werden.

Die übliche Vorgehensweise beim Browsing eines Users ist die sogenannte „depth first search“, auch Tiefensuche genannt. Hierbei folgt man den Links der aktuell betrachteten Seiten immer tiefer in das WWW hinein. Hat man nun bei der jeweils folgenden Seite wiederum einen weiterführenden Link verfolgt, kommt man schneller als man denkt an den oben beschriebenen Punkt, an dem man sich im Netz verloren hat. Um diesem Problem Abhilfe zu schaffen, arbeitet Letizia nach dem Prinzip der „breadth first search“. Bevor die von einem Dokument ausgehenden Links verfolgt werden, wird dieses zuerst auf seine Relevanz für den Benutzer hin evaluiert. Nur wenn das Dokument von Interesse ist, werden auch die Links verfolgt.



Abblidung 5: „depth first“ vs. „breadth first“ Suche

Damit wirkt Letizia der Tendenz vieler User entgegen, in die Tiefensuche zu verfallen. Die Suchtiefe von Letizia ist durch eine Maximalzahl von Zugriffen auf nicht lokale Webknoten pro Minute beschränkt. Eine solche Beschränkung ist notwendig, um den Ressourcenverbrauch im WWW handhabbar zu halten. Letizia hält sich an die von David Eichman in „Ethical Web Agents“ formulierten Prinzipien der „robot ethics“.

3.4 Rückschlüsse aus dem Browsingverhalten des Users

Das von Letizia angestrebte Ziel ist, die Vorgehensweise des Benutzers nachzuahmen. In diesem Abschnitt werden die dafür genutzten Aktionen des Users in einer kurzen Übersicht zusammengefasst.

Die Indikatoren für das Interesse eines Users lassen sich aus den folgende Aktionen ableiten:

- **Eingegeben Begriffe in Suchmaschinen:** Diese Informationen sind die beste Grundlage für die Suche von Letizia. Im nächsten Abschnitt wird darauf noch eingegangen.

- **Verweilzeit auf einer Seite:** Grundsätzlich kann davon ausgegangen werden, dass je länger ein User ein bestimmtes Dokument betrachtet, der Inhalt dieses Dokuments besonders informationshaltig ist. Allerdings tritt an dieser Stelle das Problem der Kaffepause auf. Letizia kann nicht erkennen, ob das Dokument wirklich vom User gelesen wird, oder ob er, ohne die Seite zu schließen, nur nicht mehr vor dem Rechner sitzt. Zur Lösung dieses Problems wäre es angebracht, Letizia einen Timer zur Hand zu geben, der nach einer vorgegebenen maximalen Betrachtungszeit die Gewichtung der Relevanz des aktuellen Dokumentes abbricht. Leider habe ich hierfür keine Angaben gefunden. Es ist also davon auszugehen, dass ein solcher Timer nicht für Letizia zur Verfügung steht.

- **Verfolgung der Links einer Seite:** Je mehr Links einer Seite verfolgt werden, desto relevanter kann der Inhalt dieser Seite gewertet werden.

- **Nichtbeachtete Links einer Seite:** Genauso wie die Verfolgung von Links, gibt auch die Nichtbeachtung von Links Rückschlüsse auf deren Relevanz. Die Informationen hieraus kann Letizia als „No-Goes“ speichern, und somit eine feinere Auswahl ihrer Empfehlungen treffen.

3.5 Angewandte Technologie: TF-IDF

Letizia verwendet bei näherer Betrachtung zur Bestimmung der Relevanz von Dokumenten eigentlich „nur“ die Methode der TF-IDF (Term Frequency – Inverse Document Frequency). Grundidee dieses Vorgehens ist, dass Schlüsselbegriffe, die in einem Dokument häufig, in der Gesamtheit aller Dokumente aber eher selten vorkommen, ein guter Indikator für den Inhalt von Dokumenten und damit von Relevanz für den User sind. Zunächst wird das vom User betrachtete Dokument rein auf die Vorkommenshäufigkeit der einzelnen Wörter hin untersucht.

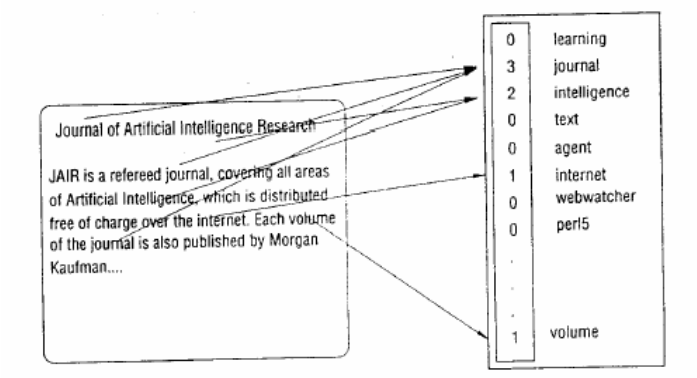


Abbildung 6: Term Frequency

Als Ergebnis erhält man, wie in Abbildung 6 gezeigt, eine Liste mit allen vorkommenden Worten mit der jeweiligen Vorkommenshäufigkeit.

Der nächste Schritt ist nun die Gewichtung der Aussagekraft der einzelnen Worte mit folgender Formel.

$$d_{ij} = tf_{ij} \times \log\left(\frac{N}{d_{fj}}\right)$$

Wobei:
- d_{ij}: Kombiniertes Gewicht des Wortes j in Dokument i
- tf_{ij}: Absolute Häufigkeit des Wortes j in Dokument i
- d_{fj}: Absolute Häufigkeit der Dokumente innerhalb der Gesamtheit von N Dokumenten, in denen Wort j mindestens einmal auftaucht

Nach einer solchermaßen erfolgten Berechnung verschiedener Dokumente werden die Erbegnisvektoren verglichen und nach Ähnlichkeit untereinander oder bezüglich eines Referenzvektors kategorisiert, wozu verschiedene Algorithmen dienen können. Im Verlauf der Suche des Users kann sich Letizia so eine recht aussagekräftige Liste an Begriffen erstellen, mit denen sie die Relevanz der von ihr untersuchten Links recht gut prüfen, und im Gut-Fall dem User zur Betrachtung empfehlen kann.

4 Schlußwort

Wenn man sich mit dem Bereich der Software-Agenten im Allgemeinen, und mit Letizia im Speziellen näher befasst, muss ich sagen, dass dies ein sehr interessantes und auch ein bisschen geheimnisvolles Gebiet ist. Allerdings ist die Bezeichnung Intelligenz für Software-Agenten meiner Meinung nach doch etwas zu gewagt. Lieberman schreibt zwar von Inferenzmechanismen und stellt damit Letizia in die Sphären der Künstlichen Intelligenz, versucht man dies allerdings zu belegen oder nachzuvollziehen, kommt man schließlich zu dem Ergebnis, dass es sich doch „nur“ um Information Retrieval mit Hilfe der TF-IDF Methode handelt. Wie Lieberman aber selbst eingesteht, ist Letizia nur der Anfang seiner Forschungsarbeit hin zu wirklich intelligenten Agenten, und es gibt zahlreiche Möglichkeiten, auch Letizia noch weit effektiver und nützlicher zu machen. Wie aber oben schon einmal erwähnt ist es immer noch besser eine, wenn auch vielleicht wenig hilfreiche, Empfehlung zu bekommen, als überhaupt keine Hilfe.

Literaturverzeichnis

[GRS00] Görz, G; Rollinger, C.-R.; Schneeberger, J: Handbuch der Künstlichen Intelligenz. Oldenburgverlag, München, Wien, 2000.

[BZW98] Brenner, W; Zarnekow, R; Wittig, H: Intelligente Softwareagenten: Springerverlag, Berlin, Heidelberg, 1998.

http://lieber.www.media.mit.edu/people/lieber/lieberary/letizia/letizia-intro.html

http://lieber.www.media.mit.edu/people/lieber/lieberary/letizia/letizia-AAAI/Letizia.html

http://www.ecommerce.wiwi.uni-frankfurt.de/lehre/00ss/seminarss00

Kontextbasiertes Information Retrieval

Softwareentwicklung

Stefanie Sieber
Norbert Wächtler

mail@steffi-sieber.de
norbert.waechtler@freenet.de

Abstract: Ein wichtiges Einsatzgebiet für kontextbasiertes Information Retrieval stellt die Unterstützung der Softwareentwicklung dar. In diesem Dokument soll die Unterstützung in Form der Suche nach passenden bereits entwickelten Softwarekomponenten beschrieben werden. Es werden verschiedene Systeme und Ansätze erläutert, die diese Funktion erfüllen.

1 Einführung

Wenn es darum geht ein Repository zur Vereinfachung, Erleichterung und auch Dokumentation von Softwareentwicklung zu schaffen, sehen sich potentielle Entwickler zwei generellen Problemen gegenübergestellt. Zum einen muss genügend Wissen gewonnen werden, um das Repository initial zu erstellen, zum anderen muss das so entstandene Repository permanent weiterentwickelt werden, um den dynamischen Ansprüchen der nutzenden Entwickler gerecht zu werden.

Mit steigender Anzahl und steigendem Umfang von Software-Repositories steigt auch das Interesse an effektiven Strukturen und Algorithmen, um die gespeicherten Komponenten wieder zu finden und erneut zu verwenden. Dabei kommt es zu einem Dilemma: Um auch nützlich zu sein, muss ein Repository eine gewisse Anzahl von Komponenten enthalten. Je größer die Zahl der enthaltenen Komponenten, desto schwieriger wird es die passenden Komponenten zu einer Anfrage, egal ob explizit oder implizit gestellt, zu finden.

Generell existieren für die Suchmethoden zu diesen Software-Repositories drei Kategorien[Hen99]:
Aufzählende Klassifikation (Enumerated Classification): Diese Retrieval-Methode teilt Komponenten bzw. Information in verschiedene Kategorien ein, die üblicherweise auch eine Hierarchie von Subkategorien enthalten. Ziel dieser Struktur ist, den gesamten Informationsraum schrittweise in immer kleinere relevante Teilbereiche aufzuteilen. Aus diesem Grund, muss je nach Gebiet, natürlich zwischen Tiefe der Struktur und maximal möglicher Zahl der Mitglieder einer Kategorie abgewogen werden. Dieser Ansatz erfordert vom Nutzer insbesondere das Verständnis der Struktur des Repositories. Das Hauptproblem liegt hier in der Findung einer adäquaten Struktur für die betroffene Domäne, da die einmal entwickelte Struktur bindend ist. Die Entwicklung einer solchen Struktur erweist sich vor allen Dingen deswegen als komplexer Prozess, da es keine Struktur gibt, die ein Gebiet für alle Sichtweisen treffend unterteilt.

Facettierende Klassifikation (Faceted Indexing): Diese Art der Klassifikation versucht die Probleme der aufzählenden Klassifikation zu vermeiden, indem Attributklassen verwendet werden, die mit bestimmten Ausprägungen instanziert werden. Durch diese Art der Klassifikation entwickelt sich ein flexibleres Schema als bei der aufzählenden Klassifikation, da einzelne Facetten eines Design erhalten werden können, ohne dass alle anderen Facetten ebenfalls geändert werden müssen. Der Nutzer startet seine Suche durch Spezifikation der gewünschten und benötigten Ausprägungen. Dabei wird er von diversen Klassifikationstechniken bei der Wahl der richtigen Begriffe unterstützt. Trotz dieser Hilfe liegt das Problem in der Wahl der zutreffenden Terme für die einzelnen Komponenten. Oft ist ein intensives Training für die Nutzer nötig um eine effektive Nutzung überhaupt möglich zu machen.

Freie Text-Indexierung (Free-Text Indexing): Die Grundlage für diese Indexierung bildet der Text des Dokumentes. Dabei wird das Dokument zu Beginn mit einer Stoppwortliste abgeglichen, die hochfrequentierte Terme ohne Aussage über den Inhalt des Dokumentes entfernt. Die verbleibenden Begriffe werden als Index für das betreffende Dokument verwendet. Der Nutzer definiert seine Suche ebenfalls durch Spezifikation dieser Begriffe, so dass die Ergebnisse durch ein einfaches Index-Matching ermittelt werden können. Die Vorteile liegen hier klar in der Einfachheit des Verfahrens. Ein Nachteil dieser Art der Indexierung ist das Aufstellen der Stoppwortliste.
Allen in dieser Arbeit vorgestellten Systemen liegt die freie Text-Indexierung zugrunde.

In dieser Arbeit sollen drei mögliche Ansätze zum Aufbau eines Software-Repositories vorgestellt werden. Ein strukturbasiertes System nach Scott Henninger [Hen99], ein textbasiertes System von Yoelle S. Mareek, Daniel M. Berry und Gail E. Kaiser [Ma91] und ein kombiniertes System von Yunwen Ye und Gerhard Fischer [YF02].

2 Strukturbasierte Ansätze [Hen99]

Die Qualität eines Systems ist hauptsächlich von der zugrunde liegenden Struktur abhängig. Das Hauptproblem von strukturbasierten Entwicklungsansätzen für Software-Repositories liegt somit in einem Trade Off zwischen der Qualität der Struktur und den anfallenden Kosten um diese Struktur zu erstellen. Zur Erstellung einer qualitativ hochwertigen Struktur sind zum Großteil aufwändige und kostenintensive Analysen der betroffenen Domäne notwendig. Gerade diese Notwendigkeit und die so entstehenden immensen FrontUp-Kosten lassen viele Unternehmen und Entwickler bereits vor Beginn der eigentlichen Entwicklung scheitern oder führen aufgrund zu geringer Etats zu schlechten Produkten, mit denen nicht effektiv gearbeitet werden kann.

Bei dem nun vorgestellten Ansatz von Scott Henninger handelt es sich um einen strukturbasierten Ansatz, der versucht die anfängliche Konstruktion des Repositories mit einer minimalen Struktur zu bewältigen, die später durch diverse Techniken kompensiert wird.

Als Tool zur initialen Erzeugung wird PEEL verwendet. Zur späteren kontinuierlichen dynamischen Verbesserung kommt CodeFinder zum Einsatz.

Das vorliegende System wurde anhand von Individualisierungen des GNU Emacs Text Editors getestet. Als Komponenten werden somit Funktionen, Variablen und Konstanten verstanden, die zwei eMail-Anwendungen und drei News-Anwendungen definieren, die aus der Emacs-Umgebung aufgerufen werden können.

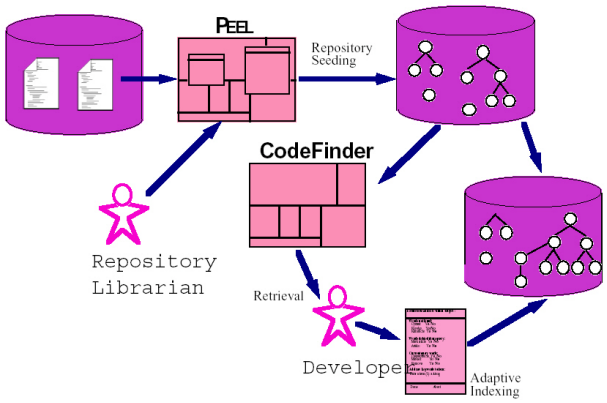


Abbildung 1: Erzeugung des Repositories [Hen99]

2.1. Aufbau des Repositories

Die Konstruktion des Systems erfolgt mit Hilfe des Prototypen PEEL.

Bevor mit der eigentlichen Erstellung des Repositories begonnen werden kann, müssen die einzelnen Komponenten-Repräsentationen erstellt werden. Dies erfolgt anhand der bereits beschriebenen freien Text-Indexierung.

PEEL extrahiert von jeder einzelnen Komponente Informationen, wie Funktionen, Variablen und Konstanten, die nötig sind um eine Repräsentation der Komponente zu erstellen aus dem Quellcode und übersetzt diese in Kandor. Kandor wird von CodeFinder benutzt um Komponenten zu indexieren. Die Repräsentation der Komponenten erfolgt als Kandor-Objekt. Für unsere Belange ist es ausreichend unter einem Kandor-Objekt eine Menge von Attributen und Werten zu verstehen, die die Komponente repräsentieren.

Da die hier stattfindende Indexierung ausschlaggebend für die Qualität des Systems ist, bietet PEEL dem Nutzer die Möglichkeit Teile der Repräsentation zu editieren und zu erweitern. Dem Nutzer werden dazu der zugrunde liegende Quelltext der Komponente und die bisher erstellte Repräsentation, die direkt editiert werden kann, angezeigt.

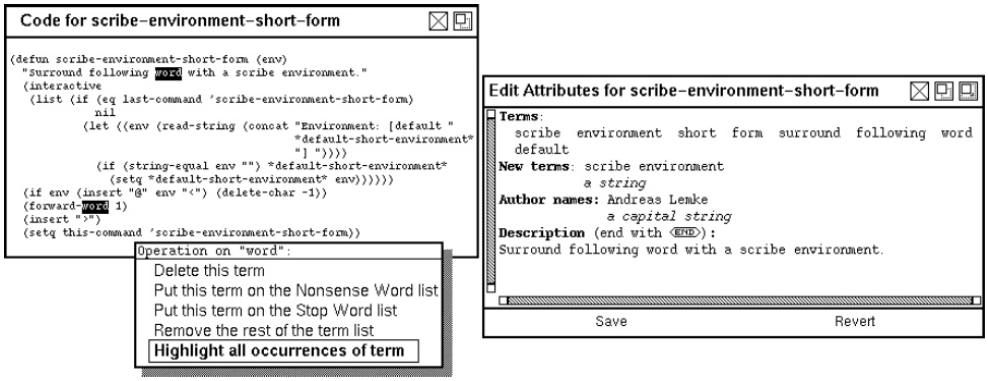


Abbildung 2: Editierungsmöglichkeiten zum Indexierungszeitpunkt [Hen99]

Um alle wichtigen Begriffe und Benennungen zu extrahieren, nutzt PEEL eine dreiteilige Methode.

Der erste Schritt findet automatisch, ohne Eingriff des Nutzers statt. Zunächst werden alle Begriffe extrahiert, die in Funktionsnamen enthalten sind oder direkt davor stehen. Eine gemeinsame Stoppwortliste wird auf die so gewonnenen Terme angewendet, um semantisch unwichtige Begriffe wieder aus der Repräsentation zu entfernen. Natürlich kann auch diese Stoppwortliste während der Erstellung der Repräsentationen erweitert werden (Abbildung 2). In einem zweiten Schritt wird das Ergebnis des ersten Schrittes dem Benutzer angezeigt. Der Nutzer kann die Repräsentation bearbeiten, Begriffe überprüfen, kürzen (um auf den Wortstamm zurückzuführen) oder auch entfernen. Im finalen Schritt kann der Nutzer seine eigene Begriffe und Phrasen hinzufügen.

Abschließend entsteht so eine vollständige Kandor-Repräsentation der indexierten Komponenten. Erweitert wird diese durch die Möglichkeit in CodeFinder Hyperlinks auf alle aufgerufenen Funktionen zu setzen, so dass der Nutzer ohne Umstände einen Überblick über die komplette Funktion der Komponenten erhält.

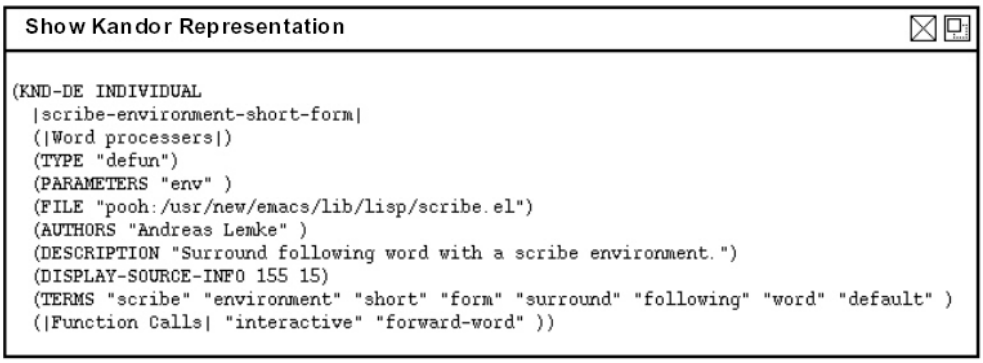


Abbildung 3: Kandor-Repräsentation [Hen99]

Bis zu diesem Zeitpunkt geht es einzig und allein um die Erstellung der Repräsentationen und nicht um die Wiederverwendung der Komponenten.

2.2. Retrieval

Das Problem der Ermittlung von Resultaten liegt in der inkonsistenten Indexierung der Komponenten sowie Anfragen. Eine Anfrage darf nicht als exakter Ausdruck der Wünsche des Nutzers verstanden werden, sondern ist vielmehr als Näherung des eigentlichen Wunsches zu verstehen. Daher ist die implizite Anfrageerweiterung als primäre Aufgabe des Retrievals zu sehen. Ein einfaches Matching ist für zufrieden stellende Ergebnisse nicht ausreichend. An dieser Stelle kommt CodeFinder zum Einsatz.

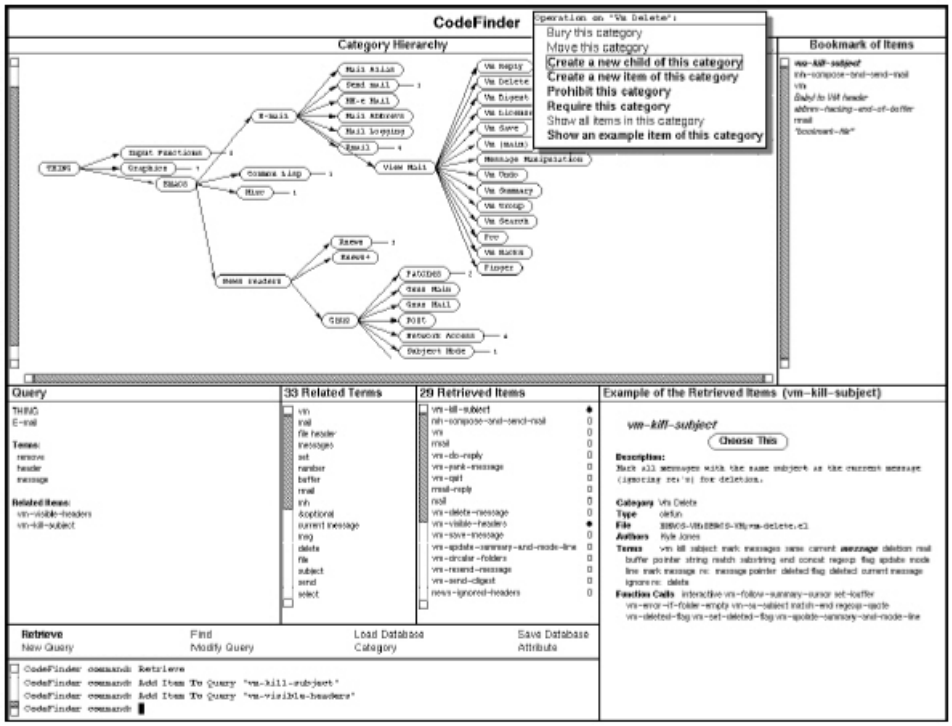


Abbildung 4: CodeFinder Interface [Hen99]

Im obersten Teil der Benutzeroberfläche wird der graphische Aufbau des Repositories dargestellt. Im unteren Teil kann der Nutzer seine Anfrage spezifizieren. Daneben werden die zur Anfrage ermittelten Begriffe, sowie die Ergebnisliste der Komponenten und Details zu ausgewählten Ergebnissen angezeigt. Zusätzlich wird der Nutzer durch eine Bookmark-Liste, die die kürzlich angesehenen Komponenten enthält, unterstützt.

2.2.1. „Spreading Activation“ - Algorithmus

Ein Matching erfolgt generell durch den Abgleich der Terme aus der Anfrage mit den Begriffen, die sich in Term-Feldern der Kandor-Repräsentationen der Komponenten befinden. CodeFinder benutzt diese Repräsentation um ein Assoziationsnetzwerk aufzubauen.
Die Anfrage des Nutzers kann sowohl Komponenten als auch Terme enthalten, verallgemeinert sprechen wir hier von Knoten. Den Knoten wird ein Startaktivierungswert von 1.0 zugewiesen. Alle positiven Werte werden nun jeweils an die verknüpften Terme und Komponenten weitergegeben. Zur Errechnung des neuen Aktivierungswertes werden die einzelnen Aktivierungswerte mit einem zugehörigen Link-Gewicht multipliziert und anschließend aufsummiert. Der erhaltene Wert wird noch durch einige Parameter, auf die in den folgenden Unterkapiteln noch eingegangen wird, modifiziert und ergibt so den neuen Aktivierungswert. Eine permanente Erhöhung durch Kreisbeziehungen wird durch eine Restriktion, die die maximal mögliche Anzahl an Kreisdurchläufen angibt unterbunden. Normalerweise stabilisiert sich der Kreislauf nach vier bis fünf Durchgängen.

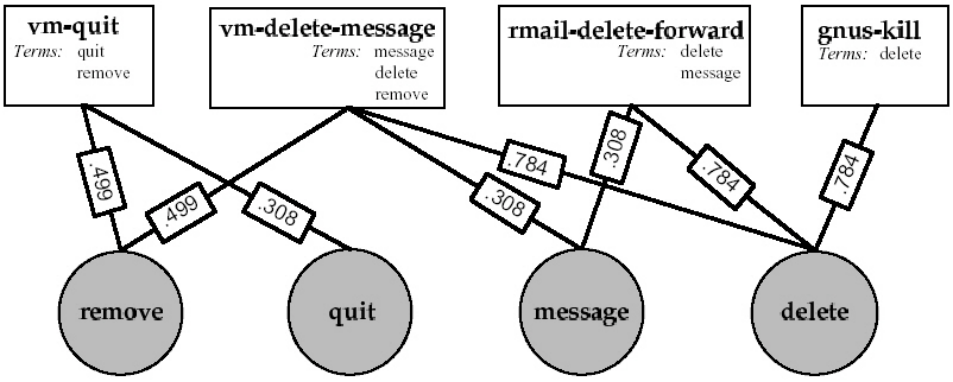


Abbildung 5: Assoziationsnetzwerk [Hen99]

Abbildung 5 zeigt ein solches Beispiel-Assoziationsnetzwerk. Die Rechtecke stellen dabei die Komponenten dar, die Kreise die Terme. Die Linen zeigen die Verknüpfungen zwischen den einzelnen Objekten und sind mit den Link-Gewichten beschriftet.

2.2.1.1. Ermittlung der Aktivierungswerte

Der Aktivierungswert der einzelnen Komponenten zu einem bestimmten Zeitpunkt wird durch folgende Formel ermittelt:

$$D_i(t+1) = \begin{cases} \delta_{Di}(t) + \Psi_{Di}(t)(M - D_i(t)) & \text{wenn } \Psi_{Di}(t) > 0 \\ \delta_{Di}(t) + \Psi_{Di}(t)(D_i(t) - m) & \text{wenn } \Psi_{Di}(t) \le 0 \end{cases}$$

mit

$\delta_{Di}(t) = (1 - \Theta_D)D_i(t)$

= vorheriger Aktivierungswert, minimiert durch den Dokument-Alterungsfaktor

$$\Psi_{Di}(t) = \left(\frac{\overline{Dt}}{Dt_i} \right)^{\alpha_D} \sum_{j=1}^{nT} w_{ij}\, U(T_j(t))$$

= Netzinput zu einem Dokument i zum Zeitpunkt t

$$U(x) = \begin{cases} 0 & \text{wenn } x \le 0 \\ x & \text{wenn } x > 0 \end{cases}$$

= Null-Schwellen-Funktion, die negative Werte unterbindet

Die Ermittlung der Aktivierungswerte für Terme erfolgt nach dem gleichen Prinzip

$$T_j(t+1) = \begin{cases} \delta_{Tj}(t) + \Psi_{Tj}(t)(M - T_j(t)) & \text{wenn } \Psi_{Tj}(t) > 0 \\ \delta_{Tj}(t) + \Psi_{Tj}(t)(T_j(t) - m) & \text{wenn } \Psi_{Tj}(t) \le 0 \end{cases}$$

mit

$\delta_{Tj}(t) = (1 - \Theta_T)T_j(t)$

= vorheriger Aktivierungswert, minimiert durch den Dokument-Alterungsfaktor

$$\Psi_{Tj}(t) = \left(\frac{\overline{df}}{dj_j} \right)^{\alpha_T} \sum_{i=1}^{nD} w_{ij}\, U(D_i(t))$$

= Netzinput zu einem Term j zum Zeitpunkt t

Symboldefinitionen:

n_D = Anzahl der Komponenten im Repository

n_T = Anzahl unterschiedlicher Terme im Repository

w_{ij} = Verbindungsstärke zwischen den Knoten i und j

idf = inverse Dokumentenhäufigkeit (hier Komponentenhäufigkeit)

Θ_D = Dokument-Alterungsfaktor

Θ_T = Term-Alterungsfaktor

M = maximaler Aktivierungswert (1.0)

m = minimaler Aktivierungswert (-0.2)

Dt_i = Anzahl der Terme in Komponente i

df_j = Anzahl der Komponenten die mit Term j indiziert wurden

$\overline{Dt}$ = durchschnittliche Anzahl der Terme pro Komponente im Repository

$\overline{df}$ = durchschnittliche Anzahl der Komponenten pro Term im Repository

α_D = Dokument Fan-In-Faktor

α_T = Term Fan-In-Faktor

2.2.1.2. Parameter des Algorithmus

Die Formeln zur Ermittlung der Aktivierungswerte ergeben, dass Knoten die einen hohen Startaktivierungswert haben, anschließend aber kein Feedback von verknüpften Knoten bekommen letztendlich den Wert 0 annehmen. Dies soll vermeiden, dass einmal aufgenommene Knoten starr im jeweiligen Assoziationsnetzwerk bleiben.

Aufgrund der erhöhten Vorkommenshäufigkeit von Termen gegenüber Komponenten, muss der „Alterungsfaktor“ bei Termen auch wesentlich höher sein, als bei Komponenten. In Experimenten wurde ermittelt, dass bei Termen ein Alterungsfaktor von 0,25 und bei Dokumenten ein Faktor von 0,1 empfehlenswert ist.

Der Netzinput wird, wie bereits erwähnt, von einem Fan-In-Faktor abgeschwächt, um zu vermeiden, dass Knoten mit vielen Verbindungen zu viele Aktivierungswerte erhalten. Experimente haben ergeben, dass Werte von 0.3 für den Komponenten-Fan-In und 0.2 für den Term-Fan-In den angesprochenen Überhang effektiv entfernen.

2.2.1.3. Link-Gewichte

Mit dem Link-Gewicht wird den übergebenen Informationen und somit Aktivierungswerten eine Wertung zugeordnet. Wichtig ist dabei stärker qualifizierenden Termen sowie Komponenten einen entsprechenden Wert zuzuweisen, um sie gegenüber unwichtigeren abzugrenzen. Da es insbesondere in einem umfangreichen Repository enormen Aufwand bedeuten würde, diesen Informationsgehalt zu ermitteln, nimmt man vereinfacht an, dass Terme mit einer geringeren Vorkommenshäufigkeit stärker differenzieren und qualifizieren als Terme mit hoher Vorkommenshäufigkeit. Dies entspricht der inversen Dokumentenhäufigkeit nach Salton und Buckley, hier wird allerdings eine leicht modifizierte Form der ursprünglichen Formel verwendet.

Die Link-Gewichte ermitteln sich somit wie folgt:

$$w_{ij} = \begin{cases} 0 \text{ (falls keine Verknüpfung zwischen i und j besteht)} \\ \text{idf} = \dfrac{\log(n_D / df_i)}{\log(n_D)} \text{ (falls eine Verknüpfung zwischen i und j besteht)} \end{cases}$$

Die so ermittelten Werte geben natürlich nur eine sehr allgemeine Aussage über den Informationsgehalt. Daher beinhaltet CodeFinder eine zusätzliche Funktion, die anhand von Relevance Feedback die Gewichte individuell anpasst (siehe Kapitel 2.3.2.).

2.2.1.4. Ermittlung der Verknüpfungen

Wichtig ist zu erwähnen, dass die Ermittlung der Verknüpfungen für das Assoziationsnetzwerk alleine aufgrund der Struktur des Repositories erfolgt. Es existieren keine vorherigen Informationen über Term- oder Komponenten-Beziehungen. Durch die Rückführung der Verknüpfungen auf die Struktur des Repositories, werden nicht nur Synonyme ermittelt, sondern insbesondere Terme, die durch häufiges gemeinsames Auftreten miteinander in Verbindung gebracht wurden. So können auch Zusammenhänge ermittelt werden, die speziell in diesem Repository existieren.

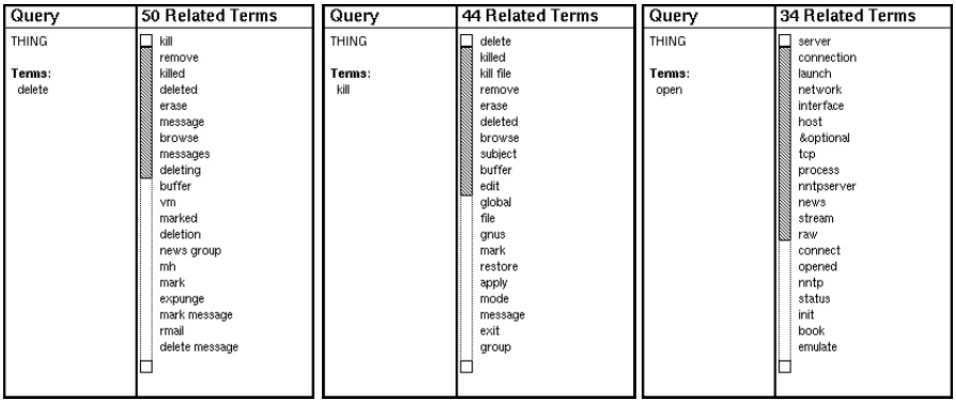


Abbildung 6: Beispiel der ermittelten Verknüpfungen zu delete, kill und open [Hen99]

2.2.2. Retrieval by Reformulation

Neben dem vorgestellten „weichen“ Matching-Algorithmus wird das Ergebnis weiterhin durch das so genannte „Retrieval by Reformulation“ verbessert. Generell kann darunter die Schulung des Nutzers während des Anfrageprozesses verstanden werden, so dass zukünftige Anfragen passender formuliert werden können. CodeFinder erreicht dies durch die vollständige Anzeige der Komponenten und ihrer Repräsentationen im Ergebnis (Abbildung 4, „Example of Retrieved Items“). Einzelne Terme können bequem zur Anfrage hinzugefügt werden und per Klick wird die Anfrage neu gestellt.

CodeFinder verbessert dieses Prinzip durch einige neue Details: Hinzugefügte Terme werden automatisch im korrekten Teil der Anfrage platziert, so dass der Nutzer nicht entscheiden muss in welchem Teil der Anfrage der Term erscheinen sollte. Weiterhin erscheinen Ergebnisse automatisch gerankt in der Ergebnisliste. Die besten Ergebnisse stehen am Beginn der Liste und das beste Ergebnis wird automatisch mit seiner vollständigen Repräsentation im Beispielfenster angezeigt. So sieht der Nutzer sofort, wie qualifiziert seine gestellte Anfrage war. Ebenso werden automatisch verwandte Begriffe angezeigt, die zur Anfrageerweiterung genutzt werden. Auch hier wird die Liste nach einem Ranking sortiert.

2.3. Inkrementelle Verbesserung des Repositories

Allein aufgrund der minimalen Struktur mit der das Repository erstellt wurde, ist es nötig eine ständige Verbesserung durchzuführen, um den Ansprüchen der Nutzer gerecht zu werden. Zusätzlich muss die enthaltene Information permanent auf dem aktuellen Stand gehalten werden, um weiterhin effektives Arbeiten zu garantieren.

Das Problem liegt darin, den Index so zu verwalten, dass er den aktuellen Problemstellungen und Denkweisen im Unternehmen angepasst ist. Dies ist besonders schwierig, da das Repository generell allein auf Basis der Syntax arbeitet und keine Informationen über die zugrunde liegende Semantik vorliegen hat.

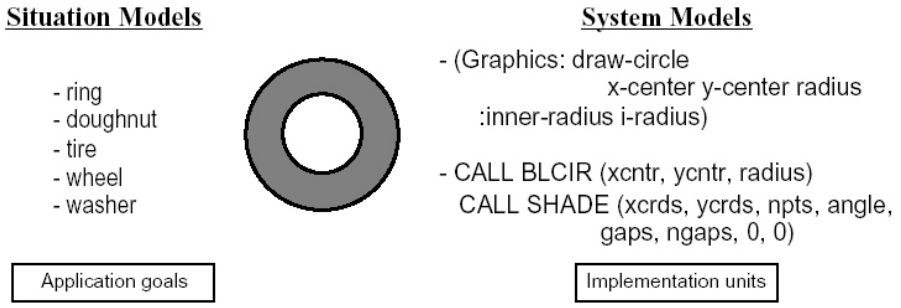


Abbildung 7: Konflikt Semantik und Syntax [Hen99]

Abbildung 7 zeigt ein Beispiel für die unterschiedlichen Interpretationen der gleichen Syntax.

2.3.1. Techniken zur Anpassung

Die Anpassung erfolgt durch ein Monitoring der Nutzer-Aktivitäten. Es soll ein System und somit ein Indexierungsschema geschaffen werden, dass mit dem Wissen der Nutzer konform ist.

Studien haben gezeigt, dass die Wahrscheinlichkeit für die Wahl der gleichen Schlüsselwörter zu einem Objekt bei zwei Nutzern nur zwischen 10 – 20% liegt. Allein um eine 60-80%ige Übereinstimmung zu erreichen muss man bereits mit 15 Synonymen arbeiten. Dies führte zur Idee der „unbegrenzten Synonyme“, d.h. jedes von einem Nutzer gewählte Wort zur Beschreibung des Objektes soll in die zugehörige Repräsentation aufgenommen werden.

Ein Nachteil der Idee liegt in den offensichtlich auftretenden Precision-verschlechterungen. Die Lösung des Problems stellt die „anpassungsfähige Indexierung“. CodeFinder unterstützt diese Art der Indexierung, indem während einer Retrieval-Session alle Terme die hinzugefügt oder gelöscht werden, in einen temporären Speicher geschrieben werden.

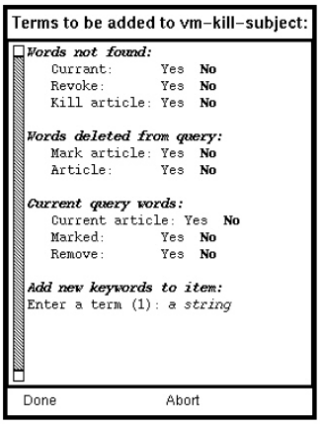


Abbildung 8: temporäre Sammlung einer Retrieval-Session [Hen99]

Um inkorrekte Anpassungen des Index, z.B. durch Tippfehler zu vermeiden, wird diese Sammlung dem Nutzer abschließend präsentiert. Per Klick kann er sich die zusammengehörigen Komponenten und Terme im Beispielfenster (Abbildung 4) anzeigen lassen und entweder dem Index hinzufügen oder diese Änderungen verwerfen.

Weitere Anpassungen des Index können auch direkt in der Kandor-Repräsentation vorgenommen werden, die durch das CodeFinder-Interface jederzeit editierbar ist. Diese Art von Änderungen erfordert allerdings einen gewissen Kenntnisstand um auch effektiv zu sein.

2.3.2. Anpassung der Link-Gewichte durch Relevance Feedback

Normalerweise wird Relevance Feedback alleine durch die Neuformulierung der Anfrage erfolgen. CodeFinder nutzt diese Art des Relevance Feedback, das der Nutzer im Interface per Klick durchführen kann, für ein erweitertes Relevance Feedback.

Wird ein Term oder eine Komponente hinzugefügt, so geht Codefinder davon aus, dass diese ebenfalls relevant zur Anfrage sind. Unter diesen Umständen wird das Link-Gewicht zwischen den bereits in der Anfrage enthaltenen und neu hinzugefügten Termen und Komponenten erhöht.

Folgende Regel wird dabei angewandt:

$w'_{ij} = w_{ij} + \eta f_i a_j$

mit

a_j = Aktivierungswert des Knoten

f_i = Feedback des User (lediglich binär, ja = 1, nein = 0)

η = Lernrate (üblicherweise zwischen 0.0 und 1.0)

Auch hier können Probleme auftreten, da oft nicht klar zugeordnet werden kann, welcher Term wie stark für die Relevanz verantwortlich ist. Studien haben aber gezeigt, dass sich die Performance durch diese Art von Relevance Feedback verbessern lässt.

2.4. Implementierung

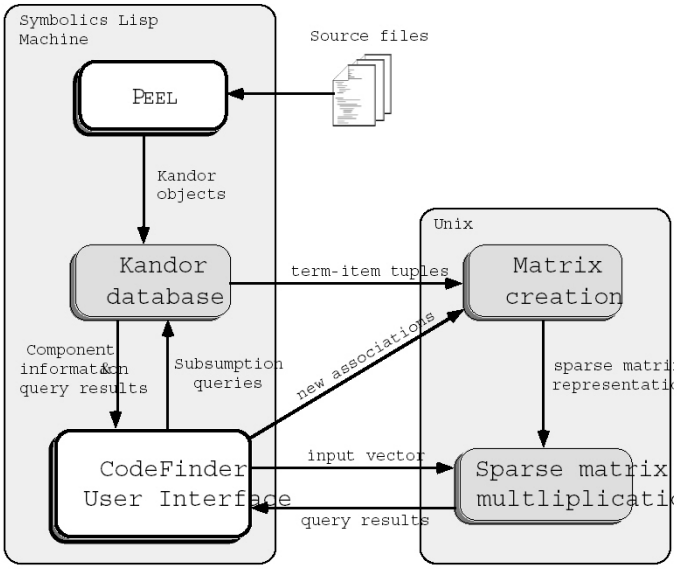


Abbildung 9: Architektur von CodeFinder [Hen99]

Abbildung 9 spezifiziert den genauen Aufbau der Architektur und Implementierung von CodeFinder.

2.5. Evaluation

In einem Experiment wurde CodeFinder mit zwei anderen Systemen, die „harte“ Matching-Algorithmen verwendet haben und bedingte Möglichkeiten zur Neuformulierung von Anfragen boten, verglichen. Getestet wurden in den Dimensionen von drei verschiedenen Kategorien der Problemdefinition (well-defined, directed, ill-defined) und Vokabular-Matching (match, mismatch).

In den Kategorien „ill-defined“ und „mismatch“ lieferte CodeFinder bessere Ergebnisse in kürzerer Zeit als die anderen beiden Systeme.

Wichtig ist an dieser Stelle zu erwähnen, dass das Experiment dazu dienen sollte das Interface zu testen und nicht den zugrunde liegenden Algorithmus. Daher konnte auch nicht mit den üblichen Evaluierungskritieren wie Precision und Recall gearbeitet werden.

3 Textbasierte Systeme [Ma91]

Der textbasierte Ansatz von Yoelle S. Mareek wurde speziell entwickelt um automatisch große Software-Repositories aufzubauen. Der Aufbau des Repositories erfolgt in zwei Schritten: Indexierung und Klassifizierung. Die Implementierung des hier vorgestellten Ansatzes GURU wurde anhand der AIX-Dokumentation getestet. Des Weiteren wird auf die Nutzung und Evaluierung des Systems eingegangen.

3.1. Indexierungsphase

Die erste Phase zur Erstellung des Systems ist die Indexierungsphase. Es handelt sich hier um eine automatische Indexierung, da dies sowohl die kostengünstigste Methode darstellt, als auch eine konsequent einheitliche Indexierung und spätere Kompatibilität der einzelnen Indices gewährleistet.

Die Indexierung verwendet kein kontrolliertes Vokabular, es handelt sich hierbei also um eine freie Text-Indexierung. Eine Besonderheit liegt in der Art des Indexes. Als Einzelteile des Index werden nicht, wie normalerweise übliche einzelne Terme verwendet, sondern Units, die zusätzlich lexikalische Ähnlichkeiten enthalten. Lexikalische Ähnlichkeiten werden dabei nur auf das Dokument und nicht auf das gesamte Vokabular bezogen, zudem werden nur open-class-words[16] verwendet. Als lexikalisch ähnlich werden alle Terme definiert, die sich innerhalb eines Satzes nicht mehr als fünf Wörter von einander entfernt befinden.

Die Extraktion erfolgt mit Hilfe der sliding-window-Technik. Termweise wird durch den Text gegangen und ein Term wird jeweils als Vergleichsterm definiert. Dann wird ein imaginäres Fenster über den Text geschoben, so dass alle Begriffe die in diesem Fenster auftauchen und den Kriterien entsprechen, zu dem Vergleichsterm als lexikalische Ähnlichkeit gespeichert werden. Die Terme werden in der kanonischen Form gespeichert, zusätzlich wird auch die Vorkommenshäufigkeit mit abgelegt.

Da nicht alle lexikalischen Ähnlichkeiten auf konzeptuelle Informationen schließen lassen, wird die Vorkommenshäufigkeit als Indikator herangezogen um den Informationsgehalt der lexikalischen Ähnlichkeit zu ermitteln. Dabei muss allerdings beachtet werden, dass eine zu hohe Vorkommenshäufigkeit den Informationsgehalt verringert.

---

[16] open-class-words sind Substantive, Verben und Adjektive. Im Kontrast dazu versteht man unter close-class-words z.B. Konjunktionen und Subjunktionen

Die Ermittlung des Informationsgehalts erfolgt nach folgenden Formeln:

Informationsgehalt eines Terms w:
$Info(w) = -\log_2(P\{w\})$
mit P = Wahrscheinlichkeit des Auftretens von w

Informationsgehalt einer lexikalischen Ähnlichkeit zwischen w_1 und w_2:
$Info(w_1, w_2) = -\log_2(P\{w_1\} \times P\{w_2\})$

Um eine weitere Information über die Aussagekraft der lexikalischen Ähnlichkeit im Bezug auf das Dokument zu erhalten, wird die „resolving power“ ρ berechnet und standardisiert:

$\rho(w_1, w_2, f) = f \times Info(w_1, w_2)$
f = Anzahl der lexikalischen Verwandtschaft (w_1, w_2) in d

Standardisierung der „resolving power“
$\rho_z = (\rho - \rho_s)/\sigma$
mit ρ_s = durchschnittliche Abweichung der ρ-Werte
und σ = Standardabweichung der ρ-Werte

Als nützliche Schwelle zur Bewertung der „resolving power“ hat sich $\rho \geq \rho_s + \sigma$ erwiesen. Alle über dieser Schwelle liegenden lexikalischen Ähnlichkeiten werden in den Index aufgenommen.

3.2. Klassifikation

Die Indexierung ist eine herkömmliche Methode, die auch für Internetsuchdienste und lokale Suchmaschinen verwendet wird. Das Problem in der Indexierung liegt oft darin, dass allgemein Dokumente bzw. speziell Komponenten, anders indexiert sind, als der Nutzer dies vermuten würde. Die Notwendigkeit der exakten Formulierung der Anfrage um die richtigen Komponenten zu finden, macht die Suche unter diesen Bedingungen sehr schwierig oder unmöglich. Um diesem Problem entgegenzuwirken, wird zusätzlich das Prinzip der Klassifikation angewandt. Dem Nutzer soll ermöglicht werden durch das Repository zu browsen. Um das Browsing effektiv gestalten zu können, ist eine korrekt aufgebaute Struktur nötig. Die Realisierung dieser Struktur erfolgt über einen invertierten Index und Clustering.

Der invertierte Index, der als erster Schritt in der Klassifikationsphase erstellt wird, wird aus dem während der Indexierung erstellten Profil abgeleitet. In diesem invertierten Index werden die bereits bekannten lexikalischen Ähnlichkeiten einzeln und als Tupel abgebildet, um doppelte lexikalische Ähnlichkeiten zu vermeiden.

Ist die Bildung des invertierten Index abgeschlossen, erfolgt das Clustering, durch das eine binäre Hierarchie entstehen soll. Es wird somit hierarchisches Clustering (Hierarchical Agglomerative Clustering, HAC) verwendet. Zu Beginn werden alle Elemente in einen einzelnen Cluster verschoben. Iterativ werden nun jeweils die ähnlichsten Cluster miteinander zu einem neuen größeren Cluster verschmolzen. Verfahren zur hierarchischen Clusterverschmelzung können single link oder complete link ebenso wie average link und Zentroidmethode sein. Wichtig ist an dieser Stelle die signifikanteste Ebene des Clusterings zu wählen, da die komplette Durchführung natürlich letztendlich zu einem großen Cluster führen würde. Die Definition der nötigen Ähnlichkeit kann durch unterschiedliche Ähnlichkeitsmaße erfolgen. Die gängigsten Maße sind Dice-Koeffizient und Jaccard-Koeffizient. Das hier verwendete Maß definiert sich wie folgt:

$$\delta(X,Y) = \sum_{i\rho\in(X)\cap\rho(Y)} (\rho_X(i) \times \rho_Y(i))$$

Auf die Verwendung der Klassifikation während der Suche wird im nächsten Unterkapitel eingegangen.

3.3. Retrieval

Die Arbeit mit dem System erfolgt primär wie bei einer normalen Suchmaschine. Da es sich hier um eine freie Text-Indexierung handelt, wird der Nutzer nicht gezwungen sich in ein kontrolliertes Vokabular einzuarbeiten, sondern kann seine Anfrage in natürlicher Sprache stellen. Das System bietet somit eine sehr nutzerfreundliche und komfortable Schnittstelle. Die gestellte Anfrage wird durch die Indexierung in für das System verständliche Attribute übersetzt, dies beinhaltet die Ermittlung der lexikalischen Ähnlichkeiten. Allerdings darf bei der Stellung der Anfrage nicht vergessen werden, dass die Anfrage nur übersetzt und nicht interpretiert wird.

Die Ermittlung der Kandidatenmenge erfolgt nach Indexierung der Komponenten und der Anfrage anhand eines simplen linearen Retrievals. Das Ergebnis der Ähnlichkeitsabfrage wird dem Nutzer als absteigendes Ranking angezeigt. Liefert das System auf diesem Level kein zufrieden stellendes Ergebnis, hat der Nutzer die Möglichkeit die lexikalischen Ähnlichkeiten als eine erste Art von Relevance Feedback in die Anfrage mit einzubeziehen.

Das eigentliche Relevance Feedback erfolgt im Anschluss an die lineare Suche und findet durch Einsatz der Klassifikation statt. Ziel ist es mit Hilfe der Ergebnisse aus der ursprünglichen Suche bessere und relevantere Komponenten zur Anfrage zu finden. Der Nutzer wählt einen Kandidaten aus der Ergebnismenge und kann im Cluster des Kandidaten browsen um auf weitere relevante Komponenten zu stoßen. Vom Startcluster aus können übergeordnete Cluster sowie Cluster der gleichen Ebene durchsucht werden. Ebenso hat der Nutzer die Möglichkeit jederzeit das Profil einer Komponente anzusehen. Dadurch wird nicht nur eine Ergebnisverbesserung, sondern vor allen Dingen auch ein Lerneffekt beim Nutzer für zukünftige Anfragen erwartet.

3.4. Systemevaluierung

Bevor das System nach den üblichen Kriterien Recall und Precision bewertet wird, sollen an dieser Stelle noch einige weitere wichtige Kriterien erwähnt werden. Das System ist sehr nutzerfreundlich. Anfragen können aufgrund des unbegrenzten Vokabulars ohne Probleme gestellt werden und auch der Wartungsaufwand nach erstmaliger Erstellung des Systems ist minimal, da das Hinzufügen neuer Komponenten problemlos über die automatische Indexierung erfolgt.

Im Zusammenhang mit den Evaluierungskriterien Recall und Precision sollen zuerst die Verbesserungen erwähnt werden, die durch die Erweiterungen des Systems erreicht werden können. Die Erweiterung der Anfrage bringt eine erwähnenswerte Verbesserung des Recalls, allerdings leidet die Precision aufgrund der erhöhten Anzahl der Terme in der Anfrage.

Zu Evaluierungszwecken wurde GURU mit dem System InfoExplorer verglichen. InfoExplorer ist ein vergleichbares System, dass auf Hypertext-Basis arbeitet. Zusätzlich gibt es in diesem System die Möglichkeit Boolesche Anfrage zu stellen, eine Alternative die in GURU nicht besteht. Die Ergebniskurven (siehe Abb. X) zeigen ein deutliches Bild.

Recall	GURU precision	INFO precision	Improvement
0.1	0.85	0.7	0.15
0.3	0.84	0.68	0.15
0.5	0.76	0.56	0.20
0.7	0.58	0.4	0.18
0.9	0.52	0.39	0.13

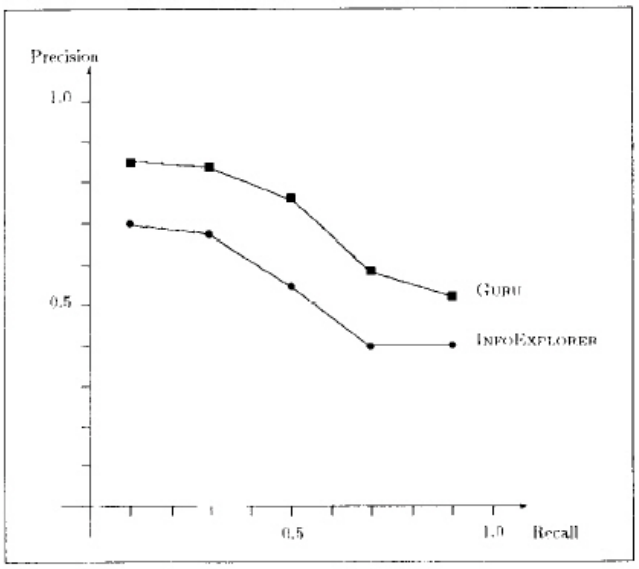


Abb. X: Ergebniskurven der Testreihe [Ma91]

Während der Recall bei beiden Systemen im Durchschnitt bei etwa 88% liegt, ist die Precision bei GURU deutlich höher, im Schnitt ca. 15% als die des InfoExplorers.

4 Kombinierte Systeme [YF02]

Die Wiederverwendung von Softwarekomponenten kann helfen sowohl Qualität als auch Produktivität von Softwareentwicklungen zu steigern. Allerdings lässt sich dabei immer wieder ein Problem beobachten: Entwickler von Software sind entweder nicht gewillt oder außerstande vorhandene Komponenten wieder zu verwenden, wenn sie nichts von ihrer Existenz wissen oder unfähig sind sie zu finden, zu verstehen und zu benutzen.

Yunwen Ye und Gerhard Fischer versuchen mit ihrem System *CodeBroker* dieses Problem anzugehen. Dabei realisieren sie in ihrem Ansatz eine Kombination aus Text- und Strukturbasierten Systemen, wobei sowohl die Dokumentation also auch die Methodensignatur des Quellcodes, die ein Entwickler gerade verfasst, extrahiert wird.

CodeBroker an sich stellt eine Erweiterung des frei zugänglichen Linux-Texteditors *Emacs* dar, welcher u.a. auch als Entwicklungsumgebung dient. Anhand des verfassten Quellcodes werden in zusätzlich eingeblendeten Fenstern Softwarekomponenten aus einem Repository vorgeschlagen, die möglicherweise zur Wiederverwendung dienen können. Darüber hinaus kann der Benutzer Ergebnisse verfeinern und das System mit Hilfe spezieller Modelle an seine Bedürfnisse weiter anpassen. Das System ist zurzeit nur für die Programmiersprache *Java* ausgelegt.

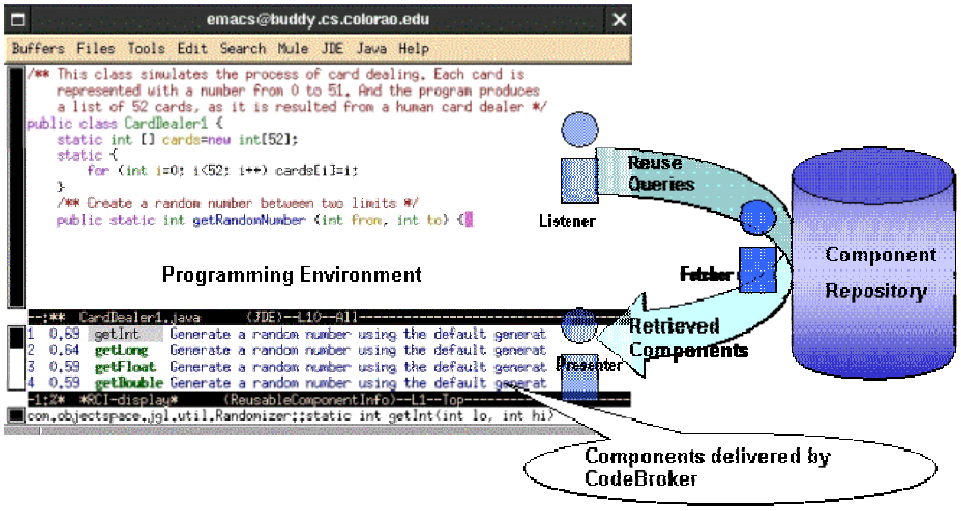


Abbildung 71: Arbeitsumgebung und Funktionsablauf von CodeBroker [Ye01]

4.1. Überblick

Bevor näher auf die Struktur und die Funktionsweise der einzelnen Komponenten von CodeBroker eingegangen wird, ist es sinnvoll sich zunächst einen groben Überblick über das Gesamtsystem zu verschaffen. Hier werden vor allem alle Hauptbestandteile von CodeBroker und deren Zusammenspiel untereinander erläutert. Dies ist wichtig, um später den Sinn und Zweck der einzelnen Bestandteile besser zu verstehen.

Der Kern von CodeBroker besteht im Wesentlichen aus zwei Systemen: Der *Interface Agent* und eine *Backend Search Engine*.
Der Interface Agent, der kontinuierlich als Hintergrundprozess von Emacs arbeitet, extrahiert und folgert automatisch Anfragen aus den Entwicklungsaktivitäten des Programmierers. Die Anfragen werden von dort an die Suchmaschine (Backend Search Engine) weitergeleitet, die das Repository nach passenden Komponenten durchstöbert.

Die gefundenen Komponenten werden mit Hilfe des Interface Agents in einem speziellen Anzeigefenster, dem so genannten *RCI-Display* (Reusable Component Information-Display) dargestellt. Bevor dies geschieht, werden die gefundenen Treffer jedoch noch anhand zweier spezieller Modelle, dem *Diskursmodell* und dem *Nutzermodell*, gefiltert und nur Komponenten angezeigt, die diese Filter passieren.

Das Diskursmodell wird vom Softwareentwickler während vorangegangener Interaktionen mit dem System erstellt. Es beinhaltet alle Komponenten, die dieser als uninteressant für die laufende Programmier- und Entwicklungssitzung empfindet.
Im Gegensatz dazu wird das Nutzermodell vom System selbst **und** vom Entwickler erstellt. Es beinhaltet alle Komponenten, die der einzelne Nutzer bereits kennt.
Komponenten, die in einem dieser Modelle bereits aufgeführt sind, haben keinen Nutzen für den Entwickler und werden folgerichtig nicht mehr in der Ergebnisliste angezeigt.

Sollte der Nutzer der Meinung sein, dass zu viele irrelevante Komponenten in der Ergebnisliste aufgeführt werden, besitzt er die Möglichkeit das so genannte *Skip Components Menu* aufzurufen. Mit dieser Funktionalität kann der Entwickler gewisse, für ihn uninteressante oder falsch interpretierte Komponenten ausblenden. Weiterhin werden die, durch diesen Vorgang gewonnen Daten, in den einzelnen Modellen zur Filterung zukünftiger Suchanfragen gespeichert.

4.2. Gewinnung kontext-basierter Informationen

Ein System, welches dem Benutzer effektiv bei der Erstellung von Software unterstützen soll, muss neben den eigentlichen Suchbegriffen auch kontext-basierte Informationen über den Anfragesteller bei der Suche berücksichtigen, um ein qualitativ hochwertiges Ergebnis zu garantieren und somit gleichzeitig den Benutzer zu motivieren das System weiterhin zu verwenden.

CodeBroker setzt hierbei auf verschiedene Techniken, die nur relevante Informationen im Bezug auf die zu bearbeitende Aufgabe und auf das benutzerspezifische Hintergrundwissen eines Entwicklers liefern. Dazu werden zwei verschiedene Modelle herangezogen, die zur Filterung der von der Suchmaschine gelieferten Komponenten dienen: Das *Diskursmodell* zur Erlangung aufgabenbezogener Komponenten und das *Nutzermodell* zur Gewinnung benutzerspezifischer Komponenten.

4.2.1. Diskursmodell

Kommentare und Signaturen im Quellcode beschreiben gewöhnlich die gerade zu bearbeitende Programmieraufgabe, und zwar die Softwaremodule, die der Entwickler erstellen wird. Ein Modul ist jedoch nur ein Teil eines gesamten Entwicklungsprojektes und deren Funktionalität ist meist tief mit anderen Modulen verwurzelt, die bis zu diesem Zeitpunkt entwickelt wurden. Deswegen ist es wichtig vorhergehende Interaktionen des Nutzers mit dem System heranzuziehen, um die gegenwärtige Entwicklungsaktivität besser interpretieren zu können und die Eignung der gewonnenen Informationen auf die gegenwärtige Aufgabe eventuell zu beschränken.

Diese früheren Interaktionen zwischen Entwickler und CodeBroker in einer bestimmten Programmiersitzung werden im Diskursmodell festgehalten. Es dient praktisch als Filter, um die Relevanz gefundener Komponenten in Bezug auf die zu bearbeitende Aufgabe zu verbessern. Dabei kann der Entwickler selbst, durch Aktivierung und Deaktivierung von CodeBroker, entscheiden, wann eine neue Sitzung beginnt und wann sie endet.

```
;; Discourse model for the session started at Thu Nov 2 08:53:12
(("java.util.zip") ;; Package added at Thu Nov 2 08:55:53.
 ("java.awt" ("CardLayout"))) ;; Class added at Thu Nov 2 09:20:3
```

Abb. 12: Struktur des Diskursmodells [YF02]

Jede neue Entwicklungssitzung beginnt zunächst mit einem leeren Diskursmodell, welches im Laufe der Zeit inkrementell vom Nutzer erweitert wird. Im Diskursmodell werden aber nur Komponenten mit aufgenommen, die **nicht** im Interesse des Nutzers sind. Die Aufnahme erfolgt mit Hilfe des Skip Components Menu, auf welches später noch näher eingegangen wird.

4.2.2. Nutzermodell

Eine Komponente zu präsentieren, mit der der Entwickler bereits aus vorangegangen Projekten gut vertraut ist, ist überflüssig. Da jeder Programmierer unterschiedliches Wissen über das Repository verfügt, sollte das System das Ergebnis einer Suchanfrage dem Bedürfnissen eines jeden einzelnen Nutzers anpassen.

Zu diesem Zweck benutzt CodeBroker das Nutzermodell, welches im Endeffekt das Wissen eines Entwicklers über das Komponenten-Repository abbildet.

Ein wesentlicher Unterschied zum Diskursmodell besteht darin, dass das Nutzermodell nicht nur vom Nutzer sondern auch vom System selbst „gewartet“ und aktualisiert wird.

Zu Beginn erstellt CodeBroker ein initiales Nutzermodell, in dem es frühere Java-Programme des Entwicklers analysiert. Dieses kann vom Nutzer im Laufe der Zeit noch explizit auf seine Bedürfnisse angepasst werden. Eine bereits bekannte Komponente, die in der Ergebnismenge auftritt, kann mit Hilfe des Skip Components Menu dem Nutzermodell hinzugefügt werden, wobei der Nutzer wählen kann, ob er nur die einzelne Komponente, die gesamte Klasse oder das gesamte Paket, in der sich die Komponente befindet, hinzufügt. Je nach dem werden Komponenten gleicher Klasse bzw. gleichen Pakets nicht mehr in der Ergebnismenge erscheinen.

Des Weiteren wird das Nutzermodell implizit von CodeBroker erweitert bzw. aktualisiert, sobald das System beobachtet, dass der Entwickler eine neue Methode während seiner Programmiertätigkeiten einfügt. Dabei macht sich CodeBroker die Tatsache zu Nutze, dass eine Methode in Java von einer linken runden Klammer gefolgt wird.

```
;; user model for Jeff
(
("java.applet"
    ("Applet"
        ("getParameterInfo")) ;; Added by Jeff at Thu 2 08:20:10 2000
)
("java.io"
    ("File"
        ("exists" "Thu Nov 2 08:35:49 2000" "Nov 2 08:15:10 2000" "Nov 2 08:
        ("isAbsolute" "Thu Nov 2 08:36:31 2000" "Nov 2 08:19:15 2000" "Nov 2
    ("CharArrayWriter"
        ("toCharArray")) ;; Added by Jeff at Thu 2 09:00:11 2000
)
("java.net") ;; Added by Jeff at Thu 2 09:15:11 2000
)
```

Abbildung 13: Beispiel eines Nutzermodells [YF02]

Die Abbildung 13 zeigt die Struktur eines Nutzermodells in CodeBroker. Das Modell wird als eine Lisp[17]-Liste mit folgendem Format gespeichert:

```
(package
 (class
  (method use-time use-time use-time …)))
```

Das *Use-Time-Field* zeigt an, wann der Entwickler eine Komponente wieder verwendet hat. Kein Use-Time-Field bedeutet, dass die Komponente vom Nutzer selbst hinzugefügt wurde.

4.2.3. Skip Components Menu

Das Skip Components Menu ermöglicht dem Nutzer irrelevante gefundene Komponenten vom RCI-Display zu filtern, in dem er sie manuell dem Diskurs- bzw. Nutzermodell hinzufügt.

[17] List Processing Language

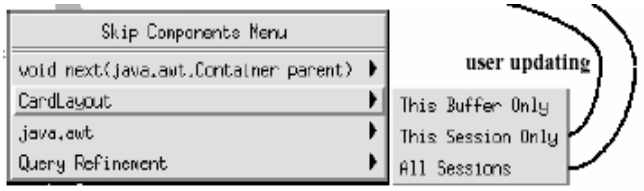


Abbildung 14: Skip Components Menu [YF02]

Es können drei Ebenen unterschieden werden, die den Umfang der Filterung beschreiben: (1) In der ersten Ebene wird nur die Komponente selbst aus der Ergebnismenge entfernt, (2) während in Ebene 2 alle Komponenten der gleichen Klasse nicht mehr erscheinen. (3) Letztendlich kann auch noch das gesamte Paket, in der sich die Komponente befindet, dem Diskursmodell hinzugefügt werden.

Weiterhin existieren für jede Ebene noch drei Befehle: Der erste Befehl, *This Buffer Only*, entfernt lediglich die Komponente aus dem Speicher des RCI-Displays. Die entsprechende Komponente wird nicht mehr angezeigt; um die Komponente zusätzlich noch dem Diskursmodell hinzuzufügen, wird der Befehl *This Session Only* ausgeführt; der dritte Befehl, *All Sessions*, veranlasst die Aufnahme im Nutzermodell.

4.3. Indexierungs- und Retrievaltechnologien [Ye01]

4.3.1. Erstellung eines Komponenten-Repositories

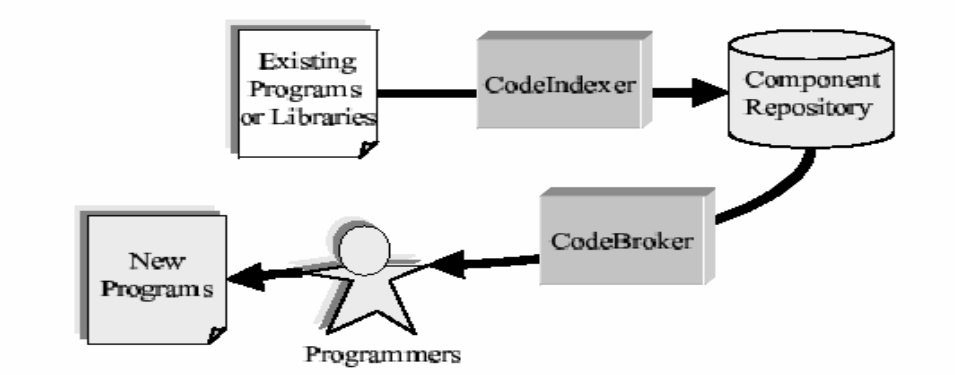


Abbildung 15: Die Teilsysteme *CodeIndexer* und *CodeBroker* [Ye01]

Damit CodeBroker überhaupt erst nach passenden Komponenten suchen kann, muss es auf ein speziell dafür erstelltes Komponenten-Repository zugreifen können. Dies ist die Aufgabe des *CodeIndexers*, der ein Repository aus existierenden *Javadoc*-Dokumenten erstellt.

Der CodeIndexer extrahiert und indexiert die Beschreibungen, die so genannten *Concepts*, und die Signaturen, die so genannten *Constraints*, von Funktionen bzw. Methoden aus der HTML-Basierten Online-Dokumentation. Diese wird mit Hilfe des Programms *Javadoc* aus den Quelltexten von Java-Programmen erzeugt. Die unten stehende Abbildung veranschaulicht diesen Vorgang.

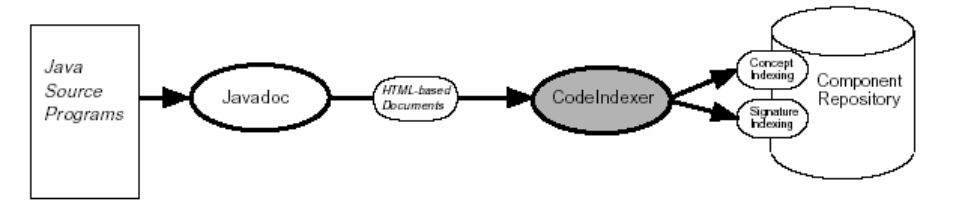


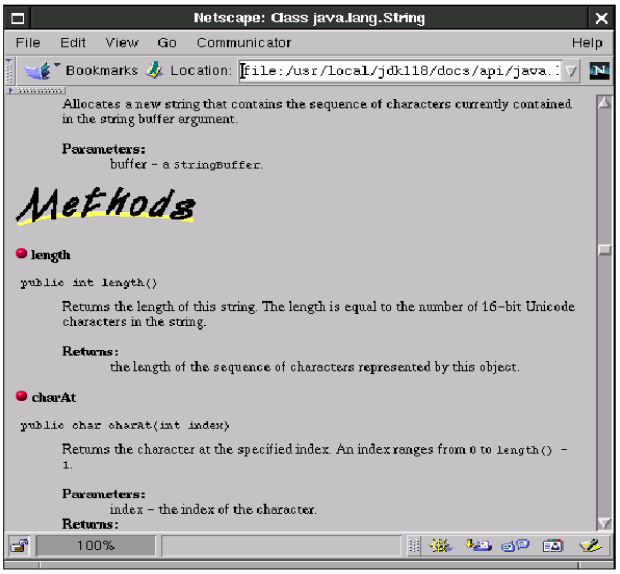
Abbildung 16: Erzeugung eines Repositories aus Java-Programmen [Ye01]

Die Erzeugung von Indizes für Java-Komponenten erfolgt im CodeIndexer in zwei Schritten:

Zunächst werden die notwendigen Informationen für den Index aus den Javadoc-Dokumenten extrahiert und in ein spezielles Format für CodeBroker überführt. Dabei werden fünf verschiedene Informationstypen gespeichert: Den gesamten Klassennamen; der HTML-Tag-Name, der den genauen Ort der Methode im Javadoc-Dokument spezifiziert; der Methodenname; die Signatur; und die Methodenbeschreibung.

Als zweiten Schritt erzeugt CodeIndexer aus den gewonnen Daten Index-Dateien für die verschiedenen Retrievaltechnologien, auf die später noch näher eingegangen wird: Eine Datei für das *Latent Semantic Analysis* (LSA), welches eine textbasierte Suche ermöglicht und eine Signatur-Index-Datei für das *Signature Matching* für eine strukturbasierte Suche.

Die LSA-Index-Datei beinhalten dabei die konzeptuellen Indizes der Komponenten, währenddessen die Signatur-Index-Datei die Indizes der Signaturen enthält.




```
NEW METHOD::
CLS: java.lang.String
TAG: length
MET: length
SIG: int length()
DEF: Returns the length of this string. The length is equal
     to the number of 16-bit Unicode characters
     in the string.

NEW METHOD::
CLS: java.lang.String
TAG: charAt
MET: charAt
SIG: char charAt(int index)
DEF: Returns the character at the specified index. An index
     ranges from 0 to length() - 1.
```

Abbildung 17: Beispiel eines Javadoc-Dokuments (oben) und das dazugehörige Indexformat in Code Broker (unten) [Ye01]

4.3.2. Retrievaltechnologien

Die Relevanz einer bestimmten Komponente wird in CodeBroker durch die Kombination aus Konzept- und Signaturgleichheit bestimmt. Aufgrund der unterschiedlichen Art der Daten unterstützt das System zwei unterschiedliche Retrievaltechnologien um eine effiziente Suche nach Komponenten zu gewährleisten. Diese Technologien sollen hier kurz vorgestellt werden:

4.3.2.1. Latent Semantic Analysis

LSA stellt eine Erweiterung des Vektorraummodells dar. Das Vektorraummodell geht davon aus, dass Terme unabhängig voneinander vorkommen und betrachtet daher keinerlei Semantiken.

LSA dagegen ist eine Freitext-Indexierungs- und Retrievaltechnik, die Semantiken in Betracht zieht. Sie kann genutzt werden, um die konzeptuelle Ähnlichkeit zwischen der gegenwärtig zu bearbeitenden Aufgabe und den Komponenten im Repository zu bestimmen. Zunächst wird aus einer großen Menge von Trainingsdokumenten eines bestimmten Bereichs ein domänen-spezifischer Wortraum erzeugt, um dann aus den Beziehungen untereinander ein allgemeines Muster zu erstellen.

Ziel ist es also, Wörter, die häufig gemeinsam auftreten, in Beziehung zu bringen. Beispielsweise tritt das Wort „Datenbank“ häufig mit dem Wort „SQL“ zusammen in Dokumenten auf. So kann man auf eine semantische Beziehung zwischen den Wörtern schließen, obwohl die beiden Ausdrücke nicht das gleiche bedeuten.

In CodeBroker wird dieses System eingesetzt, um Funktionsbeschreibungen der Javadoc-Dokumente mit den Kommentaren zur gegenwärtigen Aufgabe zu vergleichen und somit eine konzeptuelle Ähnlichkeit zwischen Anfrage und Komponente zu finden.

4.3.2.3. Signature Matching

Als Signatur einer Komponente wird die Art und Anzahl der Eingabeparameter sowie die Art des Rückgabewertes einer Methode verstanden. Das *Signature Matching* oder der Signaturvergleich ist ein Prozess zur Bestimmung der Kompatibilität zweier Komponenten bezüglich ihrer Signaturen. Hier wird also, im Gegensatz zur textbasierten Suche, ein Ansatz verfolgt, der sich die Struktur von Komponenten zu Nutze macht.

Zwei Signaturen

```
Sig1 : OutTypeExp1 <- InTypeExp1

Sig2 : OutTypeExp2 <- InTypeExp2
```

gelten nur dann als gleich, wenn der `InTypeExp1` der Signatur 1 strukturell mit dem `InTypeExp2` der Signatur 2 übereinstimmt und der `OutTypeExp1` in einer strukturellen Übereinstimmung mit dem `OutTypeExp2` steht. Zwei Typausdrücke stimmen strukturell überein, wenn sie durch Anwendung des gleichen Typ-Konstruktors zu strukturell übereinstimmenden Typen geformt wurden.

Da diese Art der Definition des Signature Matching als sehr restriktiv gilt, würden bei einer Suche Komponenten nicht auftauchen, deren Signaturen nicht hundertprozentig übereinstimmen, aber praktisch ähnlich genug sind, um sie nach kleinen Modifikationen doch wieder zu verwenden.

Das *Partial Signature Matching* „lockert“ deswegen diese Definition der strukturellen Übereinstimmung von Typen: Ein Typ wird auch dann als konform zu sich selbst angesehen, wenn dieser in einer allgemeineren oder spezialisierteren Form auftritt. Als Beispiel: In vielen Programmiersprachen wird der Typ *Integer* als eine spezielle Form von *Float* und *Float* als eine allgemeine Form von *Integer* angesehen. Allerdings wird diese Art dann mit einem geringeren Wert gewichtet als eine hundertprozentige Übereinstimmung der Signaturen.

4.4. Durchführung der Suche

In diesem Abschnitt wird eine komplette Suchdurchführung mit dem CodeBroker-System betrachtet. Dabei wird nicht nur ausschließlich auf den eigentlichen Suchprozess nach den Komponenten im Repository eingegangen, sondern vielmehr die ganzheitliche Abfolge einzelner Schritte, die für die Suche und Wiederverwendung von Softwarekomponenten notwendig sind, betrachtet. Dieser Prozess zieht sich von der Anfragestellung, über die eigentliche Suche, bis hin zur letztendlichen Präsentation der Ergebnisse. CodeBroker bietet des Weiteren auch noch die Möglichkeit der Ergebnisverfeinerung (Retrieval-by-Reformulation) um die Qualität der Ergebnisse nachträglich zu verbessern.

Die einzelnen Schritte werden anhand eines Fallbeispiels verdeutlicht. In diesem Beispiel hat ein Programmierer die Aufgabe den Austeilvorgang in einem Kartenspiel zu simulieren. Das Programm soll dabei eine Liste von 52 Karten mit dem gleichen Ergebnis erzeugen, als hätte ein menschlicher Kartengeber diese ausgeteilt.

4.4.1. Anfragestellung und Suche

In Forschungsarbeiten über Softwarewiederverwendung wurden bisher zahlreiche Mechanismen zur Lokalisierung von Komponenten getestet und vorgestellt. Zum größten Teil zielen diese auf Mechanismen wie *Browsing*, also das „Durchstöbern“ eines Verzeichnisses, und *Searching*, das Eingeben von Suchbegriffen in eine spezielle Suchmaske, ab.

Der große Nachteil an diesen Techniken ist, dass der Nutzer selbst den Suchvorgang starten und durchführen muss. Der Erfolg einer solchen Suche, und damit auch der Wiederverwendung, hängt daher stark vom Wissen des Entwicklers über das Repository ab. Browsing und Searching haben für weniger erfahrene Programmierer einen geringen Nutzen, da sie die Existenz einer Komponente zur Wiederverwendung nicht einmal erwarten oder sie möglicherweise nichts über die Nutzung des Repositories wissen.

Ye und Fischer schlagen daher mit ihren System einen neuen Mechanismus zur Suche von Komponenten vor: *Information Delivery* (Informationszustellung) soll Softwareentwicklern automatisch aufgabenbezogene und benutzerspezifische Komponenten lokalisieren und anzeigen.

Für den Vorgang des Information Delivery sind in CodeBroker zwei Teilsysteme zuständig: Der *Interface Agent* und eine *Backend Search Engine*. Beide Systeme arbeiten ständig im Hintergrund, wobei der Interfache Agent die Benutzeraktivitäten beobachtet und daraus Anfragen zur Wiederverwendung folgert und extrahiert. Diese Anfragen werden an die Suchmaschine übergeben, welche das Repository nach übereinstimmenden Komponenten durchforstet. Bevor aber nun eine Ergebnisliste im RCI-Display erscheint, werden die Treffer noch anhand des Diskurs- und Nutzermodells gefiltert.

Die große Herausforderung ist es dabei den Kontext und die eindeutigen Informationsbedürfnisse jedes Entwicklers auszuwerten, um den Nutzer nur Komponenten zu präsentieren, die einen tatsächlichen Bezug zu seiner Arbeit zeigen und die er bis jetzt noch nicht kannte. Ein schlechtes Beispiel für ein Informationslieferungs-System ist Microsoft Office's *Tipp des Tages*, da hier meist wahllos, mit der gegenwärtigen Aufgabe in keinem Bezug stehende Informationen gezeigt werden.

Beispiel:

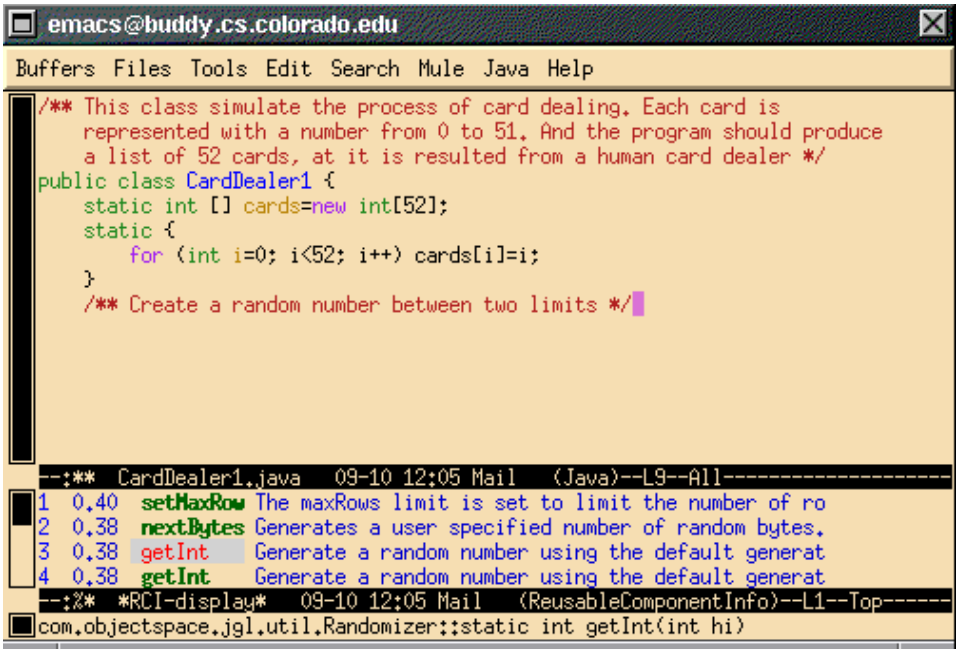


Abbildung 18: Beispiel einer automatischen Anfragegenerierung [Ye01]

In unserem Beispiel entscheidet sich der Entwickler zunächst für die Erstellung einer Methode die Zufallszahlen zwischen zwei Integerwerten generiert. Sobald er den Kommentar der Methode abschließt (zu erkennen an dem Rechtsstehenden `/`), wird der Inhalt des Kommentars in eine Anfrage extrahiert und übereinstimmende Komponenten sofort im RCI-Display präsentiert. Diese Art der Suche erfolgt auf Basis des Latent Semantic Analysis, dabei wird nach einer konzeptuellen Ähnlichkeit zwischen der Aufgabe und den Komponenten im Repository gesucht.

Es wurden nun verschiedene Komponenten mit ähnlichen Beschreibungen im unteren Teil des Editors angezeigt. Die dritte Komponente (Markierung in Abbildung 18) bietet eine ähnliche Funktionalität, jedoch liegt eine unterschiedliche Signatur vor (zu sehen in der letzten Zeile im Editor).

Der Entwickler setzt nun mit der Implementierung fort und definiert die Signatur der Methode. Sobald die Signaturbestimmung beendet ist (zu erkennen an der linken Klammer `{` vor den Cursor), extrahiert und formuliert CodeBroker eine *Constraint Query* (Signaturanfrage) aus dem Editor. Nun werden Komponenten angezeigt, die sowohl ähnliche Funktionsbeschreibungen aufweisen und auch kompatible Signaturen besitzen.

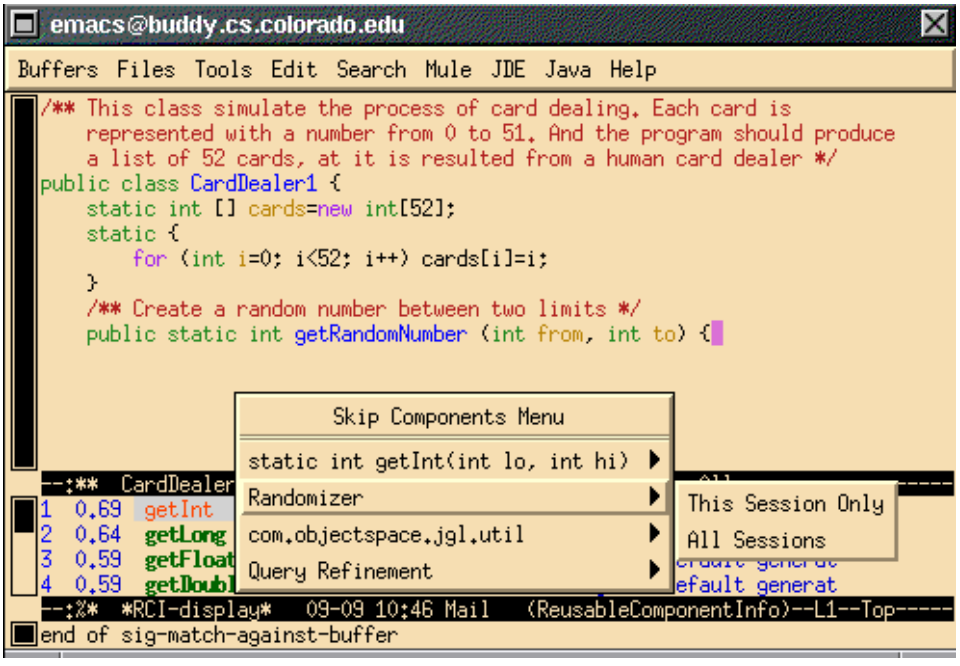


Abbildung 19: Ergebnis nach einem Signaturvergleich [Ye01]

Die erste Komponente der Ergebnisliste kann vom Nutzer direkt wieder verwendet werden.

4.4.3. Ergebnispräsentation

CodeBroker präsentiert Informationen auf drei unterschiedlichen Abstraktionsebenen.

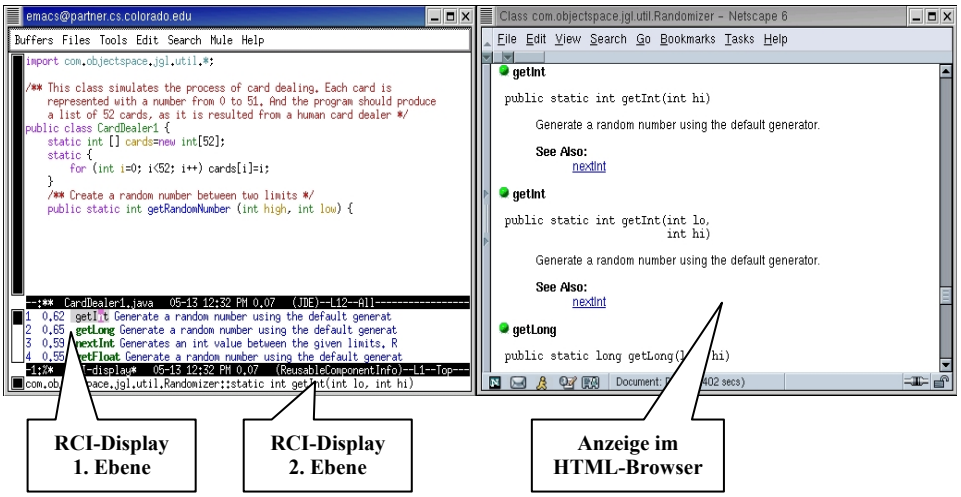


Abbildung 20: Präsentationsebenen in CodeBroker [Ye01]

Die erste Ebene bildet das RCI-Display (Reusable Component Information-Display), in der 20 (die Anzahl kann angepasst werden) gefundene Komponenten bezüglich ihrer Aufgabenrelevanz angezeigt werden können. Jede Komponente wird zusammen mit ihrem Rang, dem dazugehörigen Relevanzwert, dem Namen und einer kleinen Übersicht präsentiert, wobei Komponenten mit einem hohen Relevanzwert an vorderster Stelle angezeigt werden. Um nicht zu aufdringlich zu erscheinen, ist der Nutzer nicht gezwungen mit dem System zu interagieren, falls er kein Interesse an den gelieferten Komponenten hat.

Möchte der Entwickler jedoch weitere Informationen über eine aufgeführte Komponente im RCI-Display, so hat er die Möglichkeit eine zweite Präsentationsschicht aufzurufen. Diese erreicht er, in dem er mit dem Maus-Cursor über den Komponentennamen fährt. Nun wird die Signatur der Komponente im so genannten *Mini-Buffer* angezeigt (letzte Zeile im Editor in der Abbildung 20). Wird der Cursor dagegen über die Komponentenübersicht gezogen, werden stattdessen Wörter angezeigt, die zur Relevanz zwischen der Komponente und der gegenwärtigen Aufgabe beitragen. Dies soll den Nutzer deutlich machen, warum diese Komponente als relevant eingestuft wird und ihm bei einer eventuell notwendigen Ergebnisverfeinerung unterstützen.

Die dritte Präsentationsebene bietet letztendlich eine komplette Ansicht der Komponentenbeschreibung. Mit einem Links-Klick auf den Komponentennamen wird ein externer HTML-Browser aufgerufen, der die vollständige Javadoc-Dokumentation der entsprechenden Komponente zeigt.

4.4.4. Retrieval-by-Reformulation

Da Kommentare und Signaturen nicht immer eine Programmieraufgabe vollständig und präzise beschreiben, ist es unvermeidbar, dass CodeBroker auch einige irrelevante Komponenten liefert und genauso einige relevante Komponenten in der Ergebnisliste nicht mit aufführt. Des Weiteren sind heutige Retrievalalgorithmen, so auch das Latent Semantic Analysis, noch nicht in der Lage alle aufgabenbezogenen Informationen vollständig abzufragen.

Die *Retrieval-by-Reformulation*-Schnittstelle ermöglicht dem Benutzer von CodeBroker, nachdem er die gelieferten Komponenten einer Suche durchforstet hat, Anfragen inkrementell neu zu formulieren, bis er zu einem zufrieden stellenden Ergebnis kommt (Ergebnisverfeinerung).

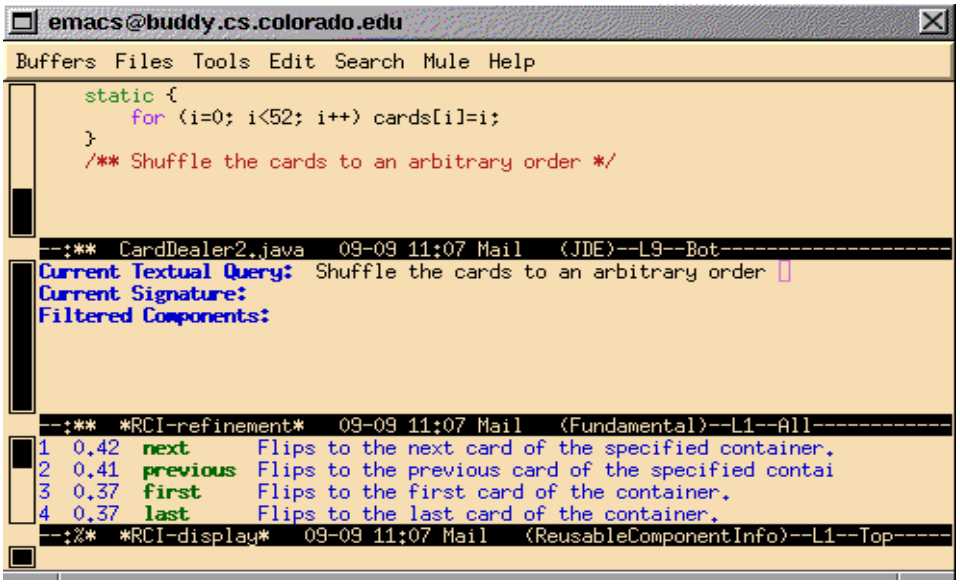


Abbildung 21: Die Retrieval-by-Reformulation-Schnittstelle [Ye01]

Die Retrieval-by-Reformulation-Schnittstelle bietet den Nutzern somit einen expliziten Kommunikationskanal, mit dem eine Anfrage verfeinert und ebenso die Suche auf einen bestimmten Bereich eingeschränkt werden kann.

Viele Repositories sind hierarchisch strukturiert. So sind in Java beispielsweise Komponenten, je nach Anwendungsbereich, in entsprechenden Klassen und Paketen organisiert. Da viele Entwicklungsaufgaben nur bestimmte Bereiche eines Repositories involvieren, kann es für Entwickler sinnvoll sein, nicht benötigte und uninteressante Klassen oder Pakete mit Hilfe von Retrieval-by-Reformulation in der Anfrage auszuschließen, und somit eine differenziertere Ergebnismenge zu erhalten.

4.5. Evaluierung

Um zu verstehen, inwiefern *Information Delivery* Entwickler bei der Wiederverwendung von Softwarekomponenten unterstützen kann, führten Ye und Fischer einen Test mit Softwareentwicklern durch, die CodeBroker bewerten sollten.

Als Test-Repository wurden 673 Klassen und 7.338 Methoden aus der Java 1.1.8 Klassenbibliothek und der JGL[18] 1.3 Bibliothek herangezogen. Mit LSA wurde zunächst ein semantischer Wortraum gebildet. Dazu wurden verschiedene Trainingsdokumente benutzt: Linux Online-Manuals, Programmier-Lehrbücher, die Java Sprachspezifikation sowie die Spezifikation der Java Virtual Machine und die Java Klassenbibliothek.

4.5.1. Testablauf

Für den Testablauf wurden insgesamt fünf Personen mit erweiterten Programmierkenntnissen ausgewählt. Die Erfahrungen in der Softwareentwicklung unter Java variierten bei den Probanden von mittelmäßig bis sehr gut.

Insgesamt wurden 12 Testreihen durchgeführt. In jedem Experiment sollten die Testpersonen ein kleines Java-Programm implementieren, welches vorher ausgewählt wurde. Jede Aufgabe konnte dabei auf unterschiedliche Art und mit verschiedenen Kombinationen der Komponenten aus dem Repository gelöst werden.

Vor Beginn der Experimente wurde jedoch zuerst ein initiales Nutzermodell der Probanden erstellt, in dem kürzlich entwickelte Java-Programme der Teilnehmer analysiert worden sind. Die Probanden wurden angehalten ihre normale Arbeitsweise bei der Implementierung beizubehalten. Jedoch wurden sie ebenso ermutigt die Vorteile von CodeBroker zu nutzen.

4.5.2. Ergebnisse

Die Abbildung 22 zeigt die Gesamtergebnisse der Studie.

Zu erkennen ist, dass die Probanden bei 10 von 12 Experimenten von CodeBroker gefundene Komponente auch wieder verwendet haben. In den 12 Programmen, die insgesamt erstellt worden sind, wurden 57 eindeutige Komponenten wieder verwendet. 20 davon wurden von CodeBroker gefunden.

[18] Java Generic Library von der Fa. Objectspace

Subject	Experiment no.	Total no. of distinct components reused	No. of distinct components reused from deliveries	Breakdown of reused components from deliveries			No. of reused components triggered by deliveries
				Unanticipated (L4-L3)	Anticipated but unknown (L3)	Vaguely known (L2)	
S1	1	10	4	2	2	0	0
	2	3	1	1	0	0	1
S2	3	7	1	1	0	0	0
	4	4	1	1	0	0	0
	5	5	3	0	2	1	1
S3	6	5	2	1	1	0	1
	7	4	3	1	2	0	1
	8	3	0	0	0	0	0
S4	9	4	3	0	3	0	0
	10	3	1	1	0	0	2
S5	11	4	1	1	0	0	2
	12	5	0	0	0	0	0
Sum		**57**	**20**	**9**	**10**	**1**	**8**

Abbildung 22: Gesamtergebnisse der Studie [YF02]

Wiederverwendung von unvorhergesehenen Komponenten. Von den 20 wieder verwendeten Komponenten, die gefunden wurden, wussten die Probanden bei 9 (siehe 5. Spalte der Tabelle) Komponenten nichts von ihrer Existenz. D.h., ohne CodeBroker wären diese Komponenten nicht wieder verwendet worden.

Verringerung der Suchzeit. In einigen Fällen wussten die Probanden zwar, dass wieder verwendbare Komponenten existierten, hätten sie aber ohne die Hilfe von CodeBroker nicht so leicht und so schnell gefunden (siehe 6. und 7. Spalte der Tabelle).

Schneeballeffekt von Ergebnissen. Die letzte Spalte der Tabelle zeigt die Anzahl der wieder verwendeten Komponenten, die zwar nicht direkt von CodeBroker gefunden wurden, jedoch die Probanden anhielt, aufgrund anderer gelieferten Komponenten, nach ihnen zu suchen.

Da Yen und Fischer eine große Gewichtung in ihrem System auf die Einbeziehung von kontext-basierten Daten mit Hilfe des Diskurs- und Nutzermodells legen, ist es interessant zu erfahren, welchen Nutzen sie in der Studie erreichen konnten:

Rolle des Diskursmodells. Wenn Diskursmodelle von den Testpersonen erzeugt wurden, verbesserten sie die Relevanz von den gelieferten Komponenten. In fünf Experimenten wurden von den Probanden Diskursmodelle erstellt. Diese filterten ca. 10% der gefundenen Komponenten. Alle entfernten Komponenten waren für die jeweilige Aufgabe irrelevant. Das Diskursmodell erfüllte also seine Aufgabe sehr zufrieden stellend.

Rolle des Nutzermodells. Im Gegensatz dazu konnte das Nutzermodell seine Aufgabe voll erfüllen. Nur 2% der gefundenen Komponenten wurden durch das Modell gefiltert. Yen und Fischer erklärten sich dies hauptsächlich mit der Unvollständigkeit der anfänglichen Nutzermodelle.

Gesamtergebnis. Insgesamt hat CodeBroker gezeigt, dass es förderlich in der Unterstützung von Software-Wiederverwendung sein kann. Seine Stärken spielt es vor allem in der Lokalisierung und Wiederverwendung von Komponenten aus, die dem Nutzer vorher nicht bekannt waren. Des Weiteren kann CodeBroker dazu beitragen Zeit und Kosten bei der Suche nach Komponenten zu reduzieren.

Die untenstehende Abbildung zeigt, dass nahezu alle Probanden das System begrüßten und gaben auch dementsprechend hohe Bewertungen auf einer Skala von 1 (für absolut nutzlos) bis 10 (für extrem nützlich). Es wurde ein Durchschnitt von 6,9 Punkten erreicht, was auf einer Schulnotenskala ca. die Note 2 bedeuten würde.

Subject	S1	S2	S3	S4	S5
Rating	7	4	8.5	7	8

Abbildung 23: Subjektive Einschätzung über die Nützlichkeit von CodeBroker [YF02]

4.6. Fazit

Ein großes Manko von CodeBroker kristallisiert sich in der Abhängigkeit des Systems mit den bereitgestellten „Lerndaten“ heraus. Bekommt CodeBroker nur wenig Material zur Analyse und Indexierung, so ist ein vernünftiges Arbeiten mit CodeBroker nur bedingt möglich. Je mehr Quellen Codebroker zur Verfügung hat und je qualitativ hochwertiger diese sind, desto besser kann CodeBroker die Daten auf semantische Eigenschaften und Räume untersuchen und so eine bessere Indexierung ermöglichen.

Eine Verbesserung und Erweiterung des Systems könnte unter Umständen mit der Adaption auf einer abstrakteren Ebene erreicht werden. Statt einzelner Komponenten könnten beispielsweise komplette Softwaredesigns oder -pattern vorgeschlagen werden. Diese Funktionalität würde allerdings völlig neue Strukturen und Algorithmen erfordern und wird in naher Zukunft wohl nicht implementiert werden. Ye und Fischer haben diesen Punkt jedoch bereits erkannt und in ihren zukünftlichen Arbeiten mit aufgenommen.

Meiner Meinung nach hat CodeBroker den Namen als unterstützendes Wiederverwendungswerkzeug verdient, mehr aber auch nicht: Von einem intelligenten System kann nämlich nicht die Rede sein. Hier werden „nur“ durch – zugegebenermaßen - geschickte Filter- und Retrieval-Techniken aus einem bereits gut strukturiertem Repository (Java-API) lediglich Komponenten vorgeschlagen. Ob diese dann genau der Intention der Entwickler entsprechen und ihnen helfen können komplexe Programmieraufgaben zu lösen, bleibt mehr oder weniger doch Glücksache, und hängt stark von der Formulierung der Anfragen ab, also im Endeffekt von den Fähigkeiten des Entwicklers selbst. Fortgeschrittene und erfahrene Programmierer können von CodeBroker sicherlich in der einen oder anderen Situation profitieren, aber ohne hinreichendes Basiswissen in Sachen Programmierung und Grundkenntnisse im Aufbau und Struktur des Repositories ist CodeBroker mehr oder weniger nutzlos.

5 Schlusswort

Die Lokalisierung von Softwarekomponenten aus großen Repositories ist der erste Schritt für den Erfolg von Software-Wiederverwendung. In dieser Arbeit wurden drei Systeme vorgestellt, die dies mit unterschiedlichen Ansätzen zu realisieren versuchen.

Alle gezeigten Ansätze sind sich im Prinzip sehr ähnlich und unterscheiden hauptsächlich von den verwendeten Features, die aus den Softwarekomponenten extrahiert werden. Während das strukturbasierte System von Henninger eine Wissensrepräsentation aus Komponenten des Repositories erzeugt, versucht das rein textbasierte System von Mareek, Berry und Kaiser die Beschreibung einer Komponente zu nutzen. Ye und Fischer haben mit ihren Ansatz gezeigt, dass sich beide Systeme auch vereinigen lassen.

Einige allgemeine Probleme, die alle Systeme betreffen, sollen bereits an dieser Stelle erwähnt werden. Wichtig ist, dass viele Algorithmen davon ausgehen, dass bereits eine Struktur für die zu speichernden Komponenten oder zumindest eine einfache Klassifikation vorhanden ist.

Einmal entwickelte Strukturen sind oft statisch und können nur durch komplettes Re-Design des Software-Repositories verändert werden. Dies erweist sich als besonders problematisch, da es ein nicht triviales Problem ist, die Struktur gleich zu Beginn richtig zu definieren. Selbst ein sehr effektiv arbeitender Algorithmus kann eine schlechte Struktur nicht kompensieren.

Neben Struktur und Aufbau eines Repositories ist auch die Qualität der Komponentenbeschreibungen von entscheidender Wichtigkeit. Textretrieval-Mechanismen lassen sich nur dann sinnvoll einsetzen wenn die vorhandenen Komponenten ausreichend und vor allem richtig dokumentiert wurden. In den meisten Ansätzen wurde auf ein qualitativ hochwertiges Repository zurückgegriffen. Es bleibt abzuwarten wie sich ein weniger gut gepflegtes Repository auf die Ergebnisqualität auswirkt.

Insgesamt kann festgehalten werden, dass einige gute Ansätze zur Unterstützung von Software-Wiederverwendung zu finden sind. Allerdings sind die meisten Systeme noch weit von einem endgültigen, in der Praxis einsetzbaren Produkt entfernt. Einige der Forschungsarbeiten wurden leider auch schon wieder eingestellt.

So bleibt letztendlich eigentlich nur zu hoffen, dass die Arbeiten auf diesem Gebiet auch weiterhin vorangetrieben werden, da das Ausmaß an wieder verwertbaren Software-Komponenten in Zukunft mit Sicherheit noch stark anwachsen wird und ein sinnvoller Einsatz sich dadurch als immer schwieriger gestaltet.

Literaturverzeichnis

[FF95] Frakes, W.B.; Fox, C.J.: Sixteen questions about software reuse. Communication of the ACM, 38(6):75-87.

[Ga03] Gall, H.: Software Wiederverwendung Einführung, http://www.infosys.tuwien.ac.at/Teaching/Courses/SWV/, Technische Universität Wien, 2003

[Gh03] Gall, H.: Domain Analysis, http://www.infosys.tuwien.ac.at/Teaching/Courses/SWV/, Technische Universität, Wien, 2003

[Hez99] Henninger, S.: An Evolutionary Approach to Constructing Effective Software Reuse Repositories, November 23, 1999

[Ma91] Maarek, Y.; Berry, D.; Kaiser, G.: An Information Retrieval Approach For Automatically Constructing Software Libraries, IEEE Transactions on Software Engineering, Vol. 17, No. 8, August 1991

[Wi04] Wieser, R.: Reuse von Code und Auswirkungen auf die Aufwandschätzung, Wirtschaftsuniversität Wien, 2004

[Ye01] Ye, Y.: Code Broker: An Active Component Repository System, http://www.cs.colorado.edu/~yunwen/codebroker/, University of Colorado, 2001

[Ye03] Ye, Y.: Programmeing with an intelligent agent, IEEE Intelligent Systems 18(3):43.47

[YF02] Y. Ye and G. Fischer. Supporting reuse by delivering task-relevant and personalized information. In Proceedings of 2002 International Conference on Software Engineering (ICSE'02), Orlando, Florida, USA, Mai 19-25 2002.

„News Dude“ – Ein persönlicher Nachrichten-Agent, der spricht, lernt und erklärt

Sebastian Gaßner
sebastian.gassner@stud.uni-bamberg.de

Abstract: War bisher von intelligenten Agenten zur Informationssuche die Rede, bezog sich dies meist auf Systeme, die einen Anwender bei der unmittelbaren Tätigkeit am Rechner unterstützen. Viele dieser Agenten stehen zudem ausschließlich über das World Wide Web zur Verfügung. Die zunehmende Mobilität der Gesellschaft verlangt jedoch immer mehr auch nach Systemen, die ohne unmittelbaren Zugang zu einem Rechnersystem bzw. einer graphischen Nutzeroberfläche funktionieren und Informationen ohne Zeitverzug zur Verfügung stellen können. Hier setzen Billsus und Pazzani mit der Vorstellung ihres persönlichen Nachrichten-Agenten „News Dude“ an: Als Teil eines IP-fähigen Autoradios liest ein Sprachsynthesizer dem Nutzer Nachrichtentexte vor. Basierend auf dessen verbalen Reaktionen paßt sich das System nach und nach an die Präferenzen und Interessen seines Benutzers an. Neben zeit-kodiertem Feedback kommt ein Multi-Strategie-Lern-Ansatz zur Anwendung, der auf einem hybriden Nutzermodell zur Modellierung sowohl der kurzfristigen als auch der langfristigen Interessen des Nutzers basiert. Besonderes Augenmerk wird der bislang eher vernachlässigten Tatsache gewidmet, daß sich das Informationsbedürfnis eines Nutzers durch die Interaktion mit Informationen verändert. Unter diesem Aspekt wird auch das explizite Modellieren dem Nutzer bereits präsentierter Informationen gesehen. Des weiteren ist es dem Nutzer möglich, mittels Konzept-Feedback über die Erklärungskomponente des Agenten direkt in das Nutzermodell einzugreifen. Die Evaluation erfolgt mit Trainingsdaten, die aus praktischen Gründen von mehreren Testpersonen über ein Web-Interface gewonnen wurden. Hierbei wird insbesondere untersucht, welchen Beitrag die einzelnen Systemkomponenten zur Gesamtperformance leisten können.

1 Motivation und Einführung

War bis zum Jahr 1999 die Rede von „Intelligenten Informations-Agenten“ (engl. intelligent information agents), also Software-Agenten, die für einen Anwender nützliche Informationen lokalisieren und abrufen, so bezog sich dies zumeist auf Agenten, die im World Wide Web zur Verfügung standen und dem Nutzer bei der unmittelbaren Arbeit am Rechner assistierten. Ihre Daseinsberechtigung finden diese Agenten im explosionsartigen Wachstum der im Internet angebotenen Informationsmenge. Das Internet als per se „anarchisches“ Medium ohne zentralen Index wird dadurch immer unübersichtlicher und die Anwender fühlen sich auf der Suche nach für sie relevanten Daten schnell völlig überfordert. Abhilfe können hier automatisierte Methoden – zum Beispiel die genannten Agentensysteme – schaffen, die unter Einbeziehung der persönlichen Präferenzen eines Nutzers die für ihn benötigten Information finden und zur Verfügung stellen.

Ein großer Nachteil der meisten bisher existierenden Agentensysteme besteht jedoch in der Notwendigkeit, zu ihrer Bedienung direkten Zugang zu einem Rechnersystem sowie dem WWW haben zu müssen. Die stetig mobiler werdende Gesellschaft mit ihren immer volleren Terminkalendern verlangt jedoch auch unterwegs Zugriff auf personalisierte Informationen – also ohne Computer mit graphischer Benutzeroberfläche.

Ein Beispiel: Zwei Vertreter A und B verbringen einen Großteil ihres Arbeitstages im Auto auf der Fahrt von einem Kunden zum nächsten. Die einzigen Informationsquellen, die ihnen unterwegs zur Verfügung stehen, sind das Autoradio sowie ihre Mobiltelefone. Zwar stehen in der Regel eine Vielzahl von Radioprogrammen verschiedener Kategorien zur Auswahl, jedoch existiert mit Sicherheit keines, das individuell auf die jeweiligen Interessen der beiden zugeschnitten ist. Angenommen, Nutzer A interessiere sich primär für Wirtschaftsnachrichten und Börseninformationen. Er bekommt diese am ehesten auf einem Nachrichtenkanal geliefert, muß sich jedoch zwischenzeitlich auch viele andere Nachrichten aus aller Welt anhören. Um trotzdem seinen Informationsbedarf zu decken, bleibt ihm nichts anderes übrig, als ständig den Sender zu wechseln und sich die Anfangszeiten der dortigen Börsennachrichten zu merken. Ähnliches gilt für Vertreter B, der ein großer Fan eines lokalen Fußballvereins ist.
Das Mobiltelefon scheidet im Auto als permanente Informationsquelle in der Regel aus, da Datendienste erstens über das kleine Display während der Fahrt nicht nutzbar und zweitens (heute noch) mit vergleichsweise hohen Kosten verbunden sind.

Billsus und Pazzani [BP99a und BP99b] haben sich nun zum Ziel gesetzt, einen Nachrichten-Agenten als Teil eines portablen Systems zu entwickeln, der – basierend auf dem IP-Protokoll – Nachrichten aus dem Internet abruft und sie seinem Träger mittels eines Sprachsynthesizers vorliest. Angeleitet vom Feedback des Nutzers paßt sich das System nach und nach an dessen individuelle Informationswünsche an und hilft ihm so, sich auch in großen Informationsangeboten zurechtzufinden. Eine Besonderheit des vorgestellten Systems stellt das sogenannte „zeit-kodierte“ (engl. time-coded) Feedback dar, das in dieser Form für textuelle Dokumente nicht anwendbar ist (siehe Kapitel 5). Langfristig streben Billsus und Pazzani die Entwicklung eine IP-fähigen Autoradios an,

das Nachrichten aus dem Internet lädt, die Interessenschwerpunkte des Fahrers lernt und ihm sein persönliches Nachrichtenprogramm zusammenstellt. Als Alternative zum herkömmlichen Rundfunkprogramm hätte die Ausstrahlung von Textinformationen an IP-fähige Empfänger den großen Vorteil, daß Text- im Gegensatz zu Audiodaten mit wesentlich geringeren Bandbreitenanforderungen übertragen werden können. Es ließe sich also ein wahre Fülle von Informationen übermitteln, aus der das „intelligente Autoradio“ dann nur diejenigen herausfiltert, die dem Nutzerprofil des Fahrers entsprechen. Dieser Auswahlprozeß kann unter anderem mit Techniken des klassischen, text-basierten Information Retrieval realisiert werden, die eine Verarbeitung, Evaluierung sowie letztlich die Empfehlung von Artikeln erst möglich machen.

Es folgen die Beschreibung der Systemarchitektur und die Erläuterung der Funktionsweise eines Prototypen. Dieser diente Billsus und Pazzani zum Sammeln von Trainingsdaten und damit zur Evaluation der Nutzermodelle bzw. der für die Modellierung verwendeten Algorithmen. Besonderes Augenmerk legen Billsus und Pazzani dabei auf den Vergleich der Kombination von zeit-kodiertem Feedback und synthetischer Sprache mit dem klassischen „Relevanz-Feedback“ (engl. relevance feedback) für geschriebene Texte. Sie beschreiben den Aufbau eines detaillierten Nutzermodells (engl. user model) und die Möglichkeiten des Agenten, dieses induzierte Nutzermodell auf der Basis von Konzept-Feedback (engl. concept feedback) zu überarbeiten. Die Evaluation erfolgt auf der Basis von realen Nutzerdaten, die aus pragmatischen Gründen über ein graphisches Trainingsinterface gewonnen wurden. Insbesondere fließt hier der Beitrag der einzelnen Systemkomponenten zur Gesamtleistung in die Bewertung ein.

2 Systemarchitektur

Beim klassischen Information Retrieval – oft ist hiermit in erster Linie die Anfragebearbeitung gemeint – geht man davon aus, daß der Nutzer einen konkreten, klar abgrenzbaren Informationsbedarf hat. Dieser entspräche bei einer Suchmaschine à la Google der Anfrage. Das vorgestellte System ist von seiner Grundkonzeption her jedoch eher dem „Information Filtering“ zuzuordnen. Der Nutzer stellt hier – grob gesprochen – die pauschale Anfrage „Was gibt es Neues in der Welt, über das ich noch nicht Bescheid weiß, das ich aber wissen sollte?“. Das Auffinden eines sinnvollen Ergebnisses zu dieser Anfrage ist für ein Rechnersystem in keinster Weise trivial: Es gilt nicht nur die stets schwankenden Interessengebiete des Nutzers in Betracht zu ziehen, vor allem muß der Tatsache Rechnung getragen werden, daß es in vielen Fällen die Aktualität einer Nachricht ist, die sie interessant macht. Gemäß dem Motto „Nichts ist älter als die Zeitung von gestern“ muß das System speichern, welche Nachrichten der Nutzer bereits gehört hat. Ziel ist also der Bau eines Systems, das die multiplen Interessen eines Nutzers in ein entsprechendes Nutzermodell abbildet, gleichzeitig flexibel genug bleibt, um rasche Änderungen der Interessenschwerpunkte zu integrieren, und parallel dazu stets weiß, welche Informationen der Anwender bereits gehört hat.

Abbildung 8: Der „News Dude“, ein Charakter aus Microsofts Agenten-Bibliothek

Der Prototyp des Systems wurde in Java implementiert und greift auf die Agenten-Bibliothek von Microsoft[19] zurück, um einen animierten Charakter anzuzeigen, der dem Nutzer Nachrichten vorliest. Dieser „News Dude“ stand für den Namen des Gesamtsystems Pate. Abbildung 1 zeigt den „News Dude“ in einigen der vielen möglichen graphischen Variationen. Darüber hinausgehend realisiert die „Microsoft-Agent“-Komponente nicht nur die Grafikausgabe, sondern übernimmt sowohl die Sprachsynthese[20] als auch die Spracherkennung[21] für die Feedback-Optionen.

Leider ist das Applet heute im Web nicht mehr erreichbar[22], da es im Home-Verzeichnis von Daniel Billsus gehostet war. Dieser hat seine Promotion mittlerweile abgeschlossen und arbeitet in der freien Wirtschaft, daher wurde sein Webspace an der Universität von Kalifornien gelöscht.

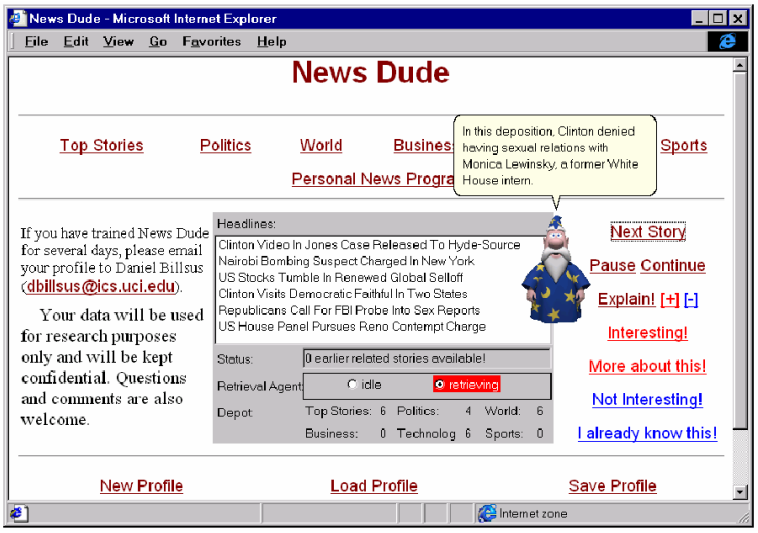


Abbildung 9: Das web-basierte User-Interface des „News Dude“

[19] http://www.microsoft.com/msagent/default.asp
[20] http://www.microsoft.com/msagent/downloads/user.asp#tts
[21] http://www.microsoft.com/msagent/downloads/user.asp#sr
[22] Das Applet war erreichbar unter http://www.ics.uci.edu/~dbillsus/NewsDude/

2.1 User Interface

Alle Funktionen des „News Dude“ können sowohl mittels Spracheingabe als auch über ein Web-Interface (siehe Abbildung 2) erreicht werden. Fragt der Nutzer eine neue Nachricht an, liest das Web-Interface die oberste Nachricht der Empfehlungsliste (engl. recommendation queue) aus und übergibt sie an den Sprachsynthesizer. Der Nutzer kann den Synthesizer jederzeit im Lesefluß unterbrechen, um ein Feedback abzugeben. Dieses Feedback wird dann vom System zur laufenden Aktualisierung des Nutzermodells verwendet.

Der Rückgriff auf die klassische web-basierte Interaktionsmöglichkeit erfolgte, wie bereits eingangs erwähnt, nur aus pragmatischen Gründen, um das System zu Testzwecken einem großen Nutzerkreis zugänglich zu machen und möglichst viele Nutzerdaten für die Evaluation sammeln zu können. Unabhängig von dem Ziel, einen rein sprachgesteuerten Agenten zu bauen, ließe sich auch im Web eine Vielzahl von Anwendungsmöglichkeiten für den „News Dude“ finden, zum Beispiel um Sehbehinderten den Zugriff auf das Medium Internet zu ermöglichen bzw. zu vereinfachen.

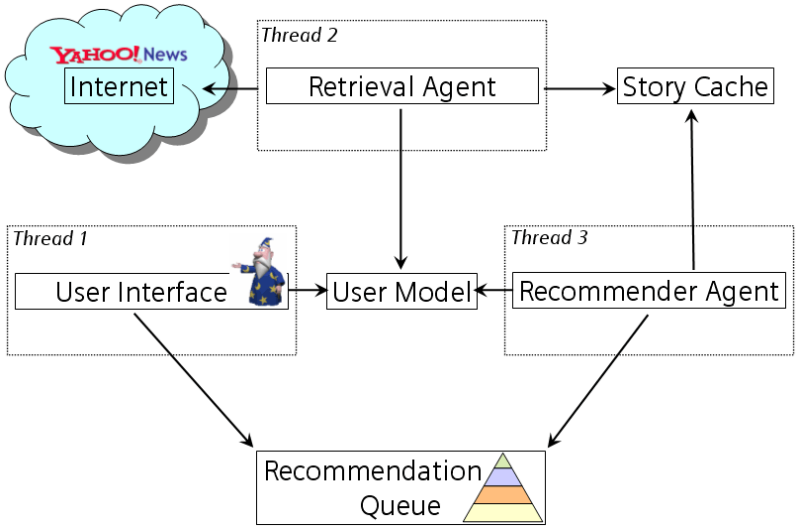


Abbildung 10: Die Systemarchitektur des „News Dude“

2.2 Retrieval Agent

Als Datenquelle dient die Site „Yahoo! News“[23] des Portalanbieters „Yahoo!“[24], die aktuelle Nachrichten zu den gängigsten Themenfeldern anbietet. Der Agent ermöglicht

[23] http://news.yahoo.com/
[24] http://www.yahoo.com/

den Zugriff auf sechs dieser Themenbereiche: Top Stories, Politics, World, Business, Technology und Sports. Der „Retrieval Agent“ läuft als einer von drei Threads (siehe Abbildung 3) permanent im Hintergrund und lädt neue Nachrichten herunter. Die neu hinzugekommenen Texte werden in einem lokalen Nachrichtenpuffer (engl. story cache) vorgehalten, bis sie vom Nutzermodell einen Relevanzwert zugewiesen bekommen und in die Recommendation Queue einsortiert werden.

2.3 Recommender Agent

Aufgabe des „Recommender Agent“ ist die Berechnung der Relevanzwahrscheinlichkeiten für neu eingegangene Nachrichten im Nachrichtenpuffer. Dies geschieht durch Anwendung des aktuellen Nutzermodells auf die einzelnen Nachrichten. Nach erfolgreicher Beurteilung (siehe Kapitel 4) wird die Nachricht an der richtigen Position in die Recommendation Queue einsortiert und steht zum Abruf durch das User Interface bereit.

2.4 Zusammenwirken der Komponenten

Ziel des Systems ist die Zusammenstellung eines persönlichen Nachrichtenprogramms für den Nutzer. Die Reihenfolge der Nachrichten soll dabei die unterschiedlich stark ausgeprägten Interessen des Nutzers widerspiegeln, d.h. interessante Themen werden vor weniger interessanten vorgelesen. Der Nutzer bewertet mittels verschiedener Feedback-Techniken (siehe Kapitel 5) die „Treffsicherheit“ (engl. accuracy) des Systems und stößt damit eine Aktualisierung seines Interessenprofils an.

Durch die Aufteilung der drei Komponenten auf eigene Threads können der Recommender Agent sowie der Retrieval Agent im Hintergrund weiter arbeiten, während das User Interface gerade mit dem Vorlesen von Nachrichten beschäftigt ist. Dies verhindert ein Leerlaufen des Nachrichtenspeichers sowie der Recommendation Queue und garantiert ein stets aktuelles Nachrichtenangebot.

Das Zusammenspiel der drei Threads resultiert in einem System, das eine nach Priorität geordnete Liste von Nachrichten vorhält, neue Nachrichten nach ihrer initialen Bewertung durch das Nutzermodell in die Liste einsortiert und dem Nutzer auf Anfrage immer die am höchsten bewertete Nachricht liefert.

3 Akquisition der kontext-bezogenen Nutzerdaten

Betrachtet man den Agenten „News Dude“ unter dem Aspekt seiner Zugehörigkeit zur Gattung der Information Retrieval-Systeme (kurz: IR-Systeme), so fällt auf, daß er im Gegensatz zum Großteil der verwandten Systeme nicht den „klassischen“ IR-Systemen zuzurechnen ist. Die Aufgabengebiete für IR-Systeme lassen sich in vier Teilbereiche einteilen [He03]: Die Klassifikation von Dokumenten, die Anfragebearbeitung, das Browsing sowie das Information Filtering. Im allgemeinen Sprachgebrauch bezieht sich der Begriff „IR-System“ jedoch oft ausschließlich auf Systeme zur Anfragebearbeitung.

Vor allem die dominante Stellung von Internetsuchmaschinen wie Google dürfte hierfür mitverantwortlich zu sein. Diese Systeme arbeiten auf dem Index einer vergleichsweise statischen Dokumentenkollektion, haben jedoch sehr verschiedenartige Benutzeranfragen möglichst akkurat zu beantworten.

Im Gegensatz zu diesen ist der „News Dude“ dem Information Filtering zuzurechnen. Aufgabe dieser Filter ist es, aus einem konstanten Strom von eingehenden Dokumenten großer thematischer Bandbreite (z. B. Usenet-News) nur diejenigen herauszusuchen, die für einen Nutzer gemäß seines Benutzerprofils relevant sind. Es versteht sich von selbst, daß in diesem Fall kein Index zur Unterstützung der Suche genutzt werden kann, da die Kollektion nie einen abgeschlossenen Zustand erreicht. Im Gegensatz zur Anfragebearbeitung sind jedoch die Präferenzen des Nutzers relativ statisch.

Genau aus diesem Grund findet beim „News Dude“ keine explizite, nutzerinitiierte Suche statt, sondern die Suchkomponente (genauer der Retrieval Agent) läuft permanent im Hintergrund (ist quasi proaktiv tätig)[25]. Die so heruntergeladenen Nachrichten werden – ebenfalls im Hintergrund – an den Recommender Agent übergeben und via Puffer fortlaufend in die Recommendation Queue einsortiert. Das „Suchergebnis“ ist also stets präsent und wartet sozusagen nur noch darauf, daß der Nutzer es abholt.

Erweitert man nun den Fokus von den klassischen IR-Systemen auch auf Systeme, die Kontextdaten des Nutzers in ihre Suchen einbeziehen, so fällt der „News Dude“ auch hier wieder aus der Reihe. Im Gegensatz zu Systemen, die vor dem Start mit Daten über ihre Nutzer gefüttert werden müssen, lernt dieser Agent selbständig. Der Kontext besteht ausschließlich aus dem durch permanentes Feedback erlernten Nutzermodell (genauer gesagt: den Nutzermodellen, mehr hierzu im folgenden Kapitel 4).

4 Das hybride Nutzermodell

Pazzani und Billsus setzen beim „News Dude“ auf einen Multi-Strategie-Lern-Ansatz (engl. multi-strategy machine learning approach), also die Verwendung mehrerer, unterschiedlicher Nutzermodelle zur Abbildung und Speicherung der Nutzerinteressen. Ausgangsbasis und Motivation für dieses Vorgehen sind drei Grundanforderungen an das System. Erstens: Der Agent muß in der Lage sein, das Interesse des Nutzers für unterschiedliche Themengebiete abzubilden. Zweitens: Das Nutzermodell muß dabei flexibel genug bleiben, um sich auch auf kurzfristige Änderungen bzw. Verschiebungen der Anwenderinteressen einstellen zu können. Dies gilt besonders dann, wenn eine längere Trainingsphase vorausging. Drittens: Das Modell muß auch in Betracht ziehen, daß sich die Interessenlage eines Nutzers durch Interaktion mit vorangegangenen Informationen verändert bzw. verändern kann. Belkin hierzu [Be97]: „Users' understanding of information problem and of useful information changes over time, and as a result of interaction with information.” Konkret bedeutet dies vor allem eines: Eine Nachricht, die der Nutzer bereits kennt, möchte er in der Regel kein zweites Mal hören.

[25] Dies wird in der Literatur auch als „queryless“ Information Retrieval bezeichnet [Be97].

Der Agent muß daher eine Historie darüber führen, was er dem Nutzer in der Vergangenheit bereits präsentiert hat. Weitere Konsequenz aus Belkins Beobachtungen: Bedingt durch die sich über die Zeit verändernden Nutzerinteressen kann eine statische Repräsentation den oben aufgestellten Anforderungen nie genügen. Interessanterweise (O-Ton Pazzani: „surprisingly“) wurde diesem Aspekt auf dem Gebiet des Information Retrieval bislang kaum Beachtung geschenkt.

Zur Erreichung eines hohen Zielerfüllungsgrades für alle drei geforderten Kriterien stehen zwei Alternativen zur Verfügung: Entweder eine getrennte Modellierung kurz- und langfristiger Interessen („hybrides Modell“) oder eine einzige, dafür in hohem Maße flexible (und komplexe) Repräsentationsform. Billsus und Pazzani wählen für den „News Dude“ die erste Variante. Sie folgen damit Belkins Philosophie, in der er feststellt [Be97]: „In general, relatively straightforward user modeling is effective in IR. The effectiveness of more complex models has been at best only partially demonstrated. Constructing complex models is quite difficult, and prone to error.“ Dies deckt sich mit den Erkenntnissen aus [He03]: Auch hier wurde regelmäßig gezeigt, daß auf dem Gebiet des Information Retrieval ein erhöhter Aufwand nicht automatisch zu besseren Ergebnissen führt. Belkin führt an gleicher Stelle weiter aus: „There is a trend in IR research to consider IR not as a system for representation and comparison of information needs and information objects, with the goal of retrieval of information objects, but as a system for supporting people’s interactions with information.“ Das Konzept des „News Dude“ fügt sich sehr gut in diesen Ideenkomplex ein.

Die zum Einsatz kommende getrennte Modellierung der kurz- und langfristigen Nutzer-Präferenzen bietet diverse Vorzüge in Bereichen, in denen zeitliche Abhängigkeiten eine große Rolle spielen. Der „News Dude“ als Agent zur Nachrichtenpräsentation fällt sicherlich in diese Kategorie. Chiu und Webb hierzu in [CW98]: „One important issue is managing temporal knowledge in agent modeling […] These approaches are based on an assumption that the agent’s knowledge, beliefs, and skills may alter over time.“ Chiu und Webb implementierten ein System, das die Kenntnisse von Studenten auf Basis von Prüfungsergebnissen abbilden und daraus Voraussagen über ihre zukünftige Entwicklung abgeben sollte. Sie setzen hierbei auf die Verwendung eines „fresh model“ für die kurzfristigen (rezenten) sowie eines „extended model“ für die Modellierung der langfristigen (weiter zurückliegenden) Resultate eines Studenten. „If the fresh model makes no prediction, due to insufficient training data or inconsistency among the training data, the system then consults the extended model.“ Chiu und Webb verglichen verschiedenste Ansätze, das Problem der temporalen Implikationen zu lösen, kommen aber schlußendlich zu dem Ergebnis, daß das eben beschriebene duale (hybride) Modell am besten abschneidet: „The results of the Dual-model approach show that the simple combination of a *fresh model* and an *extended model* is effective for achieving our objective. […] The Dual-model approach achieved significant improvement in prediction rate without significantly affecting prediction accuracy.“ Den einzigen Nachteil sehen Chiu und Webb im erhöhten Aufwand für die Erstellung und Aktualisierung zweier statt eines einzigen Nutzermodells.

Die Verwendung eines kurzfristigen Nutzermodells, das jeweils nur die letzten n aktuellen Beobachtungen speichert, führt idealerweise zu einem sehr flexiblen System,

das sich schnell an sich ändernde Interessen des Nutzers anpassen kann. Dies war eine der drei Hauptforderungen an den „News Dude“. Im konkreten Fall limitieren Billsus und Pazzani die Zuständigkeit des kurzfristigen Modells auf die letzten 100 bewerteten Nachrichten. Näheres hierzu im folgenden Kapitel 4.1. Ein weiteres Argument für den Einsatz des hybriden Modells liefert die Art der zu verarbeitenden Informationen, hier sind es Nachrichten („News“). Oft folgen auf eine initiale Nachricht mehrere Follow-Ups, die über die aktuellen Entwicklungen zu diesem Thema informieren. Man kann in der Regel davon ausgehen, daß ein Nutzer, der die erste Nachricht mit „interessant“ bewertet hat, auch an den folgenden Nachrichten interessiert ist, dem Thread also folgen möchte. Ein Beispiel: Ein Nutzer, der einer Nachricht über den erfolgreichen Start des Space Shuttle mittels Feedback eine hohe Relevanz zuordnet, wird mit großer Wahrscheinlichkeit auch an weiteren Nachrichten über den Ablauf der Mission interessiert sein. Dieser Informationswunsch läßt sich mit dem kurzfristigen Modell relativ leicht befriedigen, da hier bereits eine einzige Nachricht ausreicht, um zuverlässig Follow-Ups zu identifizieren.

Auf der anderen Seite hat ein Nutzer auch eher allgemeine Interessenschwerpunkte, die sich mit wenigen Stichworten beschreiben lassen. Wichtig wird dies dann, wenn es um die Bewertung einer neuen Nachricht geht, die vom vorgeschalteten kurzfristigen Modell aufgrund fehlender Daten oder Widersprüche nicht bewertet werden kann. Kenntnisse des Systems über die angesprochenen allgemeinen Themeninteressen helfen dann, der Nachricht doch noch eine realistische erste Relevanzwahrscheinlichkeit zukommen zu lassen. Wendet man diese Idee wieder auf das Beispiel mit dem Space Shuttle an, so würde ein Interesse für dessen Missionen wohl unter die Rubriken „Raumfahrt“, „Wissenschaft“ und „Technologie“ fallen.

Die beiden folgenden Unterkapitel erläutern nun detailliert die Funktionsweise der beiden Teil-Nutzermodelle. Mit dem Zusammenwirken der beiden Modelle beschäftigt sich dann Kapitel 4.3.

4.1 Modellierung kurzfristiger Interessen

Das kurzfristige Nutzermodelle erfüllt als Teilkomponente zwei Hauptaufgaben: Es speichert Informationen über kürzlich bewertete Nachrichten und kann dadurch inhaltliche Threads identifizieren. Billsus und Pazzani limitieren das „Gedächtnis“ des kurzfristigen Modells auf n = 100, d.h. nur die letzten 100 bewerteten Nachrichten werden zur Thread-Identifikation herangezogen. Dies garantiert die oben geforderte Flexibilität des Modells.

Außerdem ist es Aufgabe des Modells, bereits bekannte Nachrichten zu erkennen und durch geeignetes Setzen der Relevanzwahrscheinlichkeit an das untere Ende der Recommendation Queue zu verbannen.

Eine gute Wahl zur Umsetzung der Forderungen schien Billsus und Pazzani der Nearest Neighbor-Algorithmus (NN) zu sein. Dieser speichert alle ihm bekannten Trainingsdaten ab – in unserem Falle bewertete Nachrichten. Wird ihm eine neue – bislang unbewertete – Nachricht übergeben, so vergleicht er sie mit allen im Speicher

vorhandenen und ermittelt den „nearest neighbor“ bzw. die „k nearest neighbors“. Die Bewertung der neuen Nachricht kann dann aus den Bewertungen der Nachbarn abgeleitet werden.

Die prinzipielle Eignung des NN-Algorithmus zur Textklassifikation ist in der Literatur bereits mehrfach belegt. So vergleichen Cohen und Hirsh [CH98] in ihrem Paper zum Thema „WHIRL“, einem DBMS, das sogenannte „soft joins“ bzw. „similarity joins“ zur Ähnlichkeitssuche unterstützt, diverse Verfahren zur Textklassifikation (engl. baseline learning algorithms). Unter anderem kommen ein vektorbasiertes Entscheidungsbaum-Verfahren (C4.5), Nearest Neighbor mit k = 1 (1-NN), zwei Nearest Neighbor mit k > 1 ($K\text{-}NN_M$ und $K\text{-}NN_S$) und WHIRL selbst zum Einsatz. Experimentell ließ sich zeigen, daß sich k-NN WHIRL in Bezug auf die Qualität der Klassifikation nur knapp geschlagen geben mußte, dafür jedoch wesentlich zügiger arbeitete. „With respect to accuracy, WHIRL uniformly outperforms $K\text{-}NN_M$, outperforms both 1-NN and C4.5 eight of nine times, and outperforms RIPPER seven of nine times. […] The algorithm that comes closest to WHIRL is $K\text{-}NN_S$. WHIRL outperforms $K\text{-}NN_S$ eight of nine times, but all of these eight differences are small.“

Um den NN-Algorithmus auf natürlichsprachigen Text anwenden zu können, wird ein Ähnlichkeitsmaß (engl. similarity measure) benötigt. Diese Problematik ist im Information Retrieval bereits hinlänglich bekannt und gelöst (vgl. auch [He03]). Billsus und Pazzani greifen auf TF-IDF-Vektoren (term-frequency / inverse document frequency) als Dokumentenrepräsentation und die Kosinusähnlichkeit (engl. cosine similarity) zur Berechnung der Ähnlichkeit zweier Vektoren zurück [Sa89 und SB88]. Bewertete Nachrichten werden in TF-IDF-Vektoren konvertiert und im Nutzermodell abgespeichert. Die Voraussage für die Relevanzwahrscheinlichkeit einer neuen Nachricht (engl. score prediction) berechnet sich wie folgt:

Alle Nachrichten im Speicher des Modells, die näher als ein Grenzwert *t_min* am Vektor der neuen Nachricht liegen, werden „voting stories“ oder kurz „voters“. Die Relevanzkennzahl (engl. score) der neuen Nachricht berechnet sich dann aus dem gewichteten Mittelwert der Relevanzkennzahlen aller „voters“, wobei als Gewicht die Ähnlichkeit des „voters“ zur neuen Nachricht zum Einsatz kommt. Liegt jedoch einer der „voters“ näher als ein Grenzwert *t_max* am TF-IDF-Vektor der neuen Nachricht, so wird die Nachricht als bereits bekannt betrachtet und durch Multiplikation ihrer Relevanzkennzahl mit einem Faktor $k << 1$ an das untere Ende der Recommendation Queue geschoben. Lassen sich zu einer neuen Nachricht keine „voting stories“ bestimmen, so kann die Nachricht vom kurzfristigen Modell nicht bewertet werden und wird an das langfristige Modell weitergereicht.

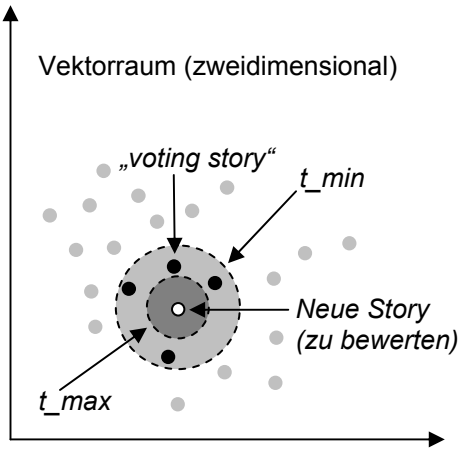


Abbildung 11: Prinzipskizze der „score prediction“, hier im zweidimensionalen Vektorraum

Das beschriebene Modell auf Basis des NN-Algorithmus erfüllt somit zwei der Hauptforderungen an das Gesamtsystem: Abbildung der unterschiedlichen Interessen eines Nutzers, sowie schnelle Anpassung an dessen veränderte Interessenlage. Als Hauptvorteil des NN-Algorithmus in diesem Kontext kann gelten, daß bereits eine einzige bewertete Nachricht im Speicher ausreicht, um zuverlässig Threads bzw. Follow-Ups zu identifizieren. Diese sogenannten „tracking abilities“ wurden auch von Allen et al. [AC+98] in einem ganz ähnlichen Projekt untersucht. Andere Verfahren benötigen in der Regel eine große Menge an Trainingsdaten, um ein starkes Pattern zuverlässig erkennen zu können.

4.2 Modellierung langfristiger Interessen

Dem langfristigen Nutzermodell kommt die Aufgabe zu, die grundlegenden Interessen des Nutzers für bestimmte Sachgebiete abzubilden und daraus Relevanzwahrscheinlichkeiten für Nachrichten zu bilden, die vom kurzfristigen Modell aus den genannten Gründen nicht bearbeitet werden konnten. Hierfür wählten Billsus und Pazzani einen probabilistischen Lernalgorithmus (engl. probabilistic learning algorithm), den naiven Bayes'schen Klassifikator (NBK). Ganz ähnlich wie beim NN-Algorithmus handelt es sich auch hier um einen relativ „simplen“ Algorithmus, der sich jedoch in der Praxis als nicht schlechter als andere – wesentlich komplexere – Verfahren erwiesen hat. Daher erfreut sich der NBK auf dem Gebiet der Textklassifikation steigender Beliebtheit.

Billsus und Pazzani beschreiben im Jahr 1997 in ihrem Paper [BP97] zum Agenten „Syskill & Webert“ detailliert, warum sie Bayes zur Nutzermodellierung einsetzen und vergleichen dessen Ergebnisse mit denen aufwendigerer Verfahren. Sie weisen nach, daß die guten Ergebnisse des NBK bereits bei sehr kleinen Trainingssets erreicht werden können. Auch wird gezeigt, warum der vorgestellte NN-Algorithmus nicht für die Modellierung langfristiger Interessen genutzt werden sollte. „In an experimental evaluation we compare the Bayesian classifier to computationally more intensive

alternatives, and show that it performs at least as well as these approaches throughout a range of different domains." Die vereinfachende Annahme der unabhängigen Vektorkomponenten wirkt sich nicht nachhaltig negativ auf das Ergebnis aus. „Furthermore it is very fast both for learning and predicting. Its learning time is linear in the number of examples and its prediction time is independent of the number of examples. It is trivial to create an incremental version of the naïve Bayesian classifier. As we shall see in the next section, it is also simple to add some form of prior knowledge to the Bayesian classifier, increasing its accuracy." Diese Erkenntnisse decken sich mit denen aus der Evaluation des „News Dude" (siehe Kapitel 6), auch hier steigt die Ergebnisqualität mit den ersten Trainingseinheiten ungewöhnlich stark an.

Auch das langfristige Modell muß die Nachrichten erst geeignet kodieren, bevor es sie abspeichern kann. Als Dokumentenrepräsentation kommen hierfür Bool'sche Merkmalsvektor zum Einsatz. Jede Merkmalsausprägung repräsentiert das Vorhandensein (1) oder Fehlen (0) eines Wortes. Da es offensichtlich in der Praxis nicht praktikabel ist, alle in den Nachrichten vorkommenden Wörter für die Bildung der Vektoren zu verwenden, einigt man sich auf ein beschränktes Vokabular. Da das Ziel die Abbildung der grundlegenden Nutzerinteressen ist, verwenden Billsus und Pazzani ein Set von 200 (handselektierten) Wörtern, die für ihre Domänen typisch sind, also Wörter, anhand deren Vorkommen man auf die Zugehörigkeit einer Nachricht zu einem der sechs vom „News Dude" bedienten Themenbereiche schließen kann. Der Algorithmus erhält dadurch die in [BP97] geforderten Hintergrundinformationen. Beispiele für die verwendeten Merkmale im Vektor sind Ländernamen sowie Terme, die zu den Gebieten Verbrechen, Katastrophen, Politik, Technik, Wirtschaft und Sport gehören. Ausgehend von der „naiven" Annahme, daß diese Merkmale voneinander unabhängig die Klassenbezeichnung (engl. class label) „interessant" bzw. „uninteressant" erhalten, erhält man folgende Wahrscheinlichkeit dafür, daß eine Nachricht unter Beachtung der in ihr enthaltenen Merkmale zur Klasse j gehört:

$$p(class_j \mid f_1, f_2, ... f_n) \text{ ist proportional zu } p(class_j) \prod_i^n p(f_i \mid class_j).$$

Die Wahrscheinlichkeiten $p(class_j)$ und $p(f_i|class_j)$ lassen sich dabei leicht aus Trainingsdaten gewinnen. Konkret verwenden Billsus und Pazzani die multivariate Bernoulli Ereignismodell-Formulierung[26] des NBK und berechnen Bayes-optimale Schätzungen von $p(class_j)$ und $p(f_i|class_j)$ durch schlichtes Zählen der Wort- und Klassenhäufigkeiten in den Trainingsdaten. Die Anwendung der Laplace-Glättung[27] sorgt schließlich dafür, daß bei selten vorkommenden Wörtern keine Null-

[26] "The multivariate Bernoulli event model represents each document by a vector of binary components, each component indicates whether or not a certain term occurs in that document. The statistical event is the document." [KG00]

[27] Die Laplace-Glättung – auch Add-One-Glättung – verhindert zuverlässig Null-Wahrscheinlichkeiten, indem zu den absoluten Häufigkeiten der Merkmale „1" addiert wird. Dieses Vorgehen weist jedoch diverse Nachteile auf, weshalb in der Praxis bessere Verfahren wie die Witten-Bell-Glättung oder die Good-Turing-Glättung eingesetzt werden.

Wahrscheinlichkeiten auftreten können. Im Endergebnis kann eine Nachricht dann mit ihrer Wahrscheinlichkeit ausgezeichnet werden, zu einer bestimmten Klasse zu gehören.

Die meisten Anwendungen der multivariaten Bernoulli-Formulierung des NBK berücksichtigen sowohl das Vorhandensein als auch das Fehlen eines Merkmals (hier: Wortes) bei der Berechnung der Wahrscheinlichkeiten. Beim langfristigen Modell des „News Dude“ fließt hingegen bloß das Vorhandensein in die Berechnung ein, was in einem konservativer agierenden Klassifikator resultiert.

Zu guter Letzt will man das langfristige Modell noch daran hindern, Nachrichten zu klassifizieren, die zu wenige Merkmale aufweisen, die ein Indikator für die Klassenzugehörigkeit sind. Beim „News Dude“ liegt die Grenze bei n = 3, das heißt, eine Nachricht muß mindestens drei Merkmale aufweisen, die alle auf die Zugehörigkeit zur gleichen Klasse deuten.

4.3 Kombination der beiden Modelle

Das gewünschte hybride Modell erhält man durch Nacheinanderschaltung des kurzfristigen und des langfristigen Modells. Eine neue, bislang nicht klassifizierte Nachricht wird dann wie folgt klassifiziert (Pseudo-Code):

```
If story can be classified by short-term model
{
    score = weighted average over nearest neighbors
    If story is too close to known story
      score = score * SMALL_CONSTANT
}
Else
    If story can be classified by long-term model
      score = probability estimated by naïve Bayes
    Else
      score = DEFAULT_SCORE
```

Das System versucht also immer zuerst das kurzfristige Modell heranzuziehen, da dieses Threads identifizieren und bereits bekannte Nachrichten herausfiltern kann. Schlägt dessen Anwendung fehl, wird die zu klassifizierende Nachricht an das langfristige Modell übergeben. Kann auch dieses die Nachricht aufgrund mangelnder Trennschärfe (zu wenige Merkmale für eindeutige Klassenzugehörigkeit, vgl. oben) nicht bewerten, wird ein Standardwert (engl. default score) vergeben. Dieser muß empirisch derart gewählt werden, daß davon betroffene Nachrichten so in die Recommendation Queue eingefügt werden, daß sie einerseits keine zu hohe Relevanzwahrscheinlichkeit erhalten, andererseits aber „interessanter“ bleiben als Nachrichten, die erfolgreich als „uninteressant“ gekennzeichnet wurden. Beim „News Dude“ wurde der Wert „0,3“

gewählt, es erfolgt also eine Einsortierung im unteren Drittel der Recommendation Queue.

5 Feedbackmöglichkeiten

Eines der Ziele von Billsus und Pazzani war es, beim „News Dude“ einen hohen Grad an Interaktion zwischen System und Nutzer zu unterstützen. Die Rückmeldungen (engl. feedback) des Nutzers versetzen das System in die Lage, ein präzises Nutzermodell zu erlernen und dieses sukzessive zu verbessern. Dem Anwender stehen verschiedene Möglichkeiten offen, die Relevanzeinschätzungen des Systems zu evaluieren. Die folgenden Unterkapitel beschäftigen sich näher mit der Funktionsweise dieser Features.

Eine Besonderheit auf dem Gebiet der intelligenten Agenten stellt die Fähigkeit des „News Dude“ dar, seine Entscheidungen betreffend die Relevanz einer Nachricht zu begründen und diese Begründung dem Nutzer akustisch mitzuteilen. Diese Funktionalität war bislang wissensbasierten Systemen, spezieller Expertensystemen, vorbehalten.

5.1 Funktionsweise des „time-coded feedback“

Diese Feedbackmöglichkeit wurde ursprünglich von Henry Lieberman [Li95] für seinen Web-Agenten „Letizia“ erdacht. Sie basiert darauf, die Verweilzeit (engl. dwell time), die ein Nutzer auf einer Webseite verbringt für die Einstufung ihrer Relevanz zu nutzen. „Repeatedly returning to a document also connotes interest, as would spending a lot of time browsing it [relative to its length], if we tracked dwell time.“ Da es sich beim „News Dude“ um einen Agenten ohne GUI handelt, muß das Feedback-Prinzip für die Sprachkommunikation zugänglich gemacht werden. Dies wurde von Billsus und Pazzani folgendermaßen umgesetzt: Der Nutzer kann die Sprachausgabe einer Nachricht jederzeit unterbrechen und ein Feedback geben. Das System nutzt diese Eingaben, um die gespeicherten Nutzermodelle zu ergänzen bzw. zu überarbeiten. Die Audiosynthese der Nachrichten läßt sich also implizit zur Gewinnung von Relevanzinformationen nutzen, die ein Nutzer beim reinen Lesen eines Textes nicht zur Verfügung stellt.

Man geht hierbei von der Grundannahme aus, daß der Zeitpunkt, zu dem der Nutzer den Sprachsynthesizer zum Feedback unterbricht, hoch aussagekräftig ist und daher bei der Nutzermodellierung berücksichtigt werden sollte. Ein Anwender wird einer Nachricht, die er interessant findet, sicherlich länger Gehör schenken als einer, die ihn weniger interessiert. In der vorliegenden Implementierung des „News Dude“ wird das binäre Feedback des Nutzers („interessant“ contra „uninteressant“) wie folgt auf eine feinere Skala konvertiert. Dabei sei *pl* der prozentuale Anteil der Nachricht, den der Anwender vor der Eingabe des Feedbacks gehört hat:

Nachricht wurde mit ‚uninteressant’ bewertet:	$score = 0{,}3 \times pl$
Nachricht wurde mit ‚interessant’ bewertet:	$score = 0{,}7 + 0{,}3 \times pl$
Nutzer wünscht zusätzliche Informationen:	$score = 1{,}0$

Dabei gilt zu beachten, daß die Konstanten „0,3“ und „0,7“ willkürlich gewählt wurden. Sie sollen sicherstellen, daß „uninteressante“ Nachrichten stets geringere „score“-Werte erhalten, als „interessante“ Nachrichten. Bessere Konstanten ließen sich zwar experimentell ermitteln, dies haben Billsus und Pazzani jedoch erst für zukünftige Projekte vorgesehen. Wünscht der Nutzer eventuell sogar weitere Informationen, hat das System sicher einen „Volltreffer“ gelandet, daher wird *pl* in diesem Fall nicht weiter berücksichtigt.

5.2 Erklärungskomponente für die Empfehlungen

Sowohl im klassischen Information Retrieval als auch bei bisherigen Agenten wurde auf die Fähigkeit, Relevanzbewertungen des Systems zu erklären, kaum Wert gelegt. Das System modelliert intern die Informationsbedürfnisse des Nutzers, sucht mit diesen Angaben nach relevanten Daten und präsentiert sie dem Nutzer. Der Anwender hat jedoch keine Möglichkeit, Einblick in das Nutzermodell zu bekommen um z. B. einer ihm dubios vorkommenden Relevanzbeurteilung auf den Grund zu gehen. Ganz zu schweigen von der Möglichkeit, das Nutzermodell an dieser Stelle gezielt zu manipulieren. Diese von Billsus und Pazzani als Konzeptfeedback eingeführte Technik wird im folgenden Kapitel 5.3 näher beleuchtet.

Mehrere Gründe sprechen dafür, dem Nutzer zu erläutern, welche Argumente zur Klassifikation einer Nachricht genutzt wurden: So kann eine kurze Erklärung dem Nutzer helfen zu beurteilen, ob er eine Nachricht näher „ansehen“ möchte. Besonders im hier gegebenen Audiokontext gilt dies, da der Anwender keine Möglichkeit hat (wie z. B. im Web) alle angebotenen Items kurz zu überfliegen und danach die interessantesten auszuwählen. Die in Kapitel 5.1 eingeführte zeitliche Komponente macht sich in diesem Falle negativ bemerkbar: Da eine parallele Sprachausgabe kaum sinnvoll wäre, bleibt es beim streng sequentiellen Ablauf. Ein weiterer Vorteil des Erklärungsansatzes: Der Nutzer hat damit direkten Einblick in den aktuellen Zustand des Nutzermodells und kann beurteilen, ob die gerade zum Einsatz gekommene Entscheidungsregel zukünftig zum Auffinden relevanter Informationen nützlich sein kann.

Die Erklärungen erfolgen in Form vorgeformter Sätze (engl. explanation templates) deren Variablen mit entsprechenden Werten gefüllt werden. Je nachdem, ob die Nachricht mit dem kurzfristigen oder dem langfristigen Modell bewertet wurde, kommen unterschiedliche Konstrukte zum Einsatz:

Wurde die gerade abgespielte Nachricht mit dem kurzfristigen Modell bewertet, so lautet die Erklärung: *„This story received a **[high | low]** relevance score, because you told me earlier that you were **[not]** interested in **[closest_headline]**.“* Die Variable „closest_headline“ bezeichnet in diesem Falle die Überschrift der Nachricht, die im kurzfristigen Nutzermodell die höchste Ähnlichkeit zur aktuellen Nachricht aufweist. Geht das System davon aus, daß die Nachricht bereits bekannt ist, so lautet die entsprechende Aussage: *„I think you already know about this, because I told you earlier that **[closest_headline]**.“*

Wurde die Nachricht hingegen vom langfristigen Modell bewertet, so muß zur Begründung erst einmal bestimmt werden, welche Merkmale (= Wörter) die Klassenzuordnung am nachhaltigsten beeinflußten. Dies geschieht durch Berechnung von $influence_f = log\ [p(f \mid c) / p(f \mid not\ c)]$ und Ausgabe von: *„This story received a **[high | low]** relevance score, because it contains the words f1, f2 and f3."* Der „News Dude" berücksichtigt für die Erklärung also die drei aussagekräftigsten Wörter.

Kann keines der beiden Modelle die Nachricht bewerten und wurde daher der Standardwert zugewiesen, so wird dies mit *„The story received a default score, because it did not seem to be related to any previously rated story, and did not contain enough informative words that would allow classification."* begründet.

5.3 Funktionsweise des „concept feedback"

Der „News Dude" erlaubt es dem Nutzer nun, die gegebenen Erklärungen zu kritisieren und damit ein Feedback zu geben. Diese Kritik geht umgehend ins Nutzermodell ein und modifiziert es. Billsus und Pazzani nennen diese Form der Rückmeldung an das System Konzeptfeedback (engl. concept feedback) und grenzen es damit vom im Information Retrieval weit verbreiteten Relevanzfeedback (engl. relevance feedback) ab: Nicht die Klassifikation selbst (also das Ergebnis) ist hier die Grundlage der Bewertung, sondern das Modell („Konzept"), das zur Klassifikation führte! Drei denkbare Vorteile sprechen für dieses Vorgehen: 1. Die Vorlieben des Nutzers werden präziser erfaßt. 2. Die Menge der benötigten Trainingsdaten läßt sich reduzieren. 3. Das hybride Nutzermodell gewinnt an Flexibilität. Gerade letzterer Aspekt ist mehr als erwünscht, da im Falle des „News Dude" dem Anwender stets schwankende Interessenslagen unterstellt werden.

Folgende Terme stehen dem Nutzer für das Feedback zur Verfügung und werden von der Spracherkennung des „News Dude" erkannt: „interesting", „not interesting", „I already know this", sowie „tell me more about this". Abhängig von der Erklärungsvorlage beeinflußt das Feedback die Nutzermodelle auf unterschiedliche Weise:

Wurde eine Nachricht durch das kurzfristige Modell bewertet und der Nutzer gibt ein negatives Feedback ab, so wird die Nachricht aus dem Modell gelöscht. Dies ist vor allem dann sinnvoll, wenn ein Nutzer einem Thread, den er bisher interessant fand, nicht mehr länger folgen möchte. Umgekehrt wird eine korrekt klassifizierte Nachricht erneut ins Modell eingefügt, dies verleiht ihr künftig doppeltes Gewicht. Ähnlich verhält es sich im Fall bereits bekannter Nachrichten. Hat das System richtig „getippt", so bleibt das Modell unverändert. Gibt der Nutzer jedoch ein negatives Feedback, so wird der Wert von *t_max* (vgl. Kapitel 4.1) minimal verändert, um künftig eine präzisere Erkennung bekannter Nachrichten zu ermöglichen. Im Falle eines „false positive" wird die Grenze *t_max* verengt, so daß neue Nachrichten in Zukunft eine noch höhere Ähnlichkeit zu einer bereits bekannten Nachricht aufweisen müssen, um als bekannt eingestuft zu werden. Wurde eine Nachricht jedoch fälschlicherweise nicht als Dublette erkannt („false negative"), so wird *t_max* weiter nach außen verschoben.

Nachrichten, die durch das langfristige Modell erfaßt wurden, werden anders behandelt. Hier wird im Falle eines positiven Feedbacks ein künstliches Trainingsbeispiel kreiert, daß lediglich aus den Wörtern (*f1*, *f2*, *f3*) besteht, die in der Erklärung verwendet wurden. Dieses Set wird dann in beide Nutzermodelle eingefügt! Die Klassenzugehörigkeit ergibt sich dabei aus dem Feedback des Nutzers. Dies hat folgende Konsequenzen: Im langfristigen Modell werden lediglich die Frequenzzähler aktualisiert, die für die Berechnung der Wahrscheinlichkeiten benutzt werden. Daher werden Fehleinschätzungen, die aus den gleichen Termen resultieren, seltener. Im kurzfristigen Modell ist der Effekt problematischer. Da es künftig für die Erkennung von Nachrichten zuständig ist, die die genannten Terme enthalten, kann es passieren, daß eine Nachricht nur aufgrund des Vorhandenseins der drei Terme als hochrelevant eingestuft wird, unabhängig vom restlichen Inhalt der Nachricht. Vice versa kann es passieren, daß eine Nachricht aus diesem Grund ausgefiltert wird, obwohl sie prinzipiell doch interessant wäre. Leider bieten Billsus und Pazzani beim „News Dude“ (noch) keinen Lösungsvorschlag für dieses Problem an.

6 Evaluation

Um eine Evaluation des Systems durchführen zu können, mußten Nutzerdaten gesammelt werden, um die anfangs leeren Nutzermodelle zu füllen. Dies geschah wie eingangs erwähnt über ein Web-Frontend, da sich nur so eine größere Anzahl von Nutzern ansprechen und zur Teilnahme motivieren ließ. Der Prototyp wurde von zehn Personen über einen Zeitraum von vier bis acht Tagen trainiert. Insgesamt wurden somit mehr als 3000 Nachrichten bewertet, pro Nutzer ungefähr 300 Stück. Da es sich um eine relativ kleinen Testumfang handelt, lassen sich keine allgemeingültigen Schlüsse ziehen, jedoch kann man das relative Abschneiden des Systems bzw. seiner Einzelkomponenten beurteilen.

Die Leistungsfähigkeit des „News Dude“ zu messen stellt eine gewisse Schwierigkeit dar. Durch die zeitlichen Abhängigkeiten zwischen den Nachrichten muß die Reihenfolge der Trainingseinheiten immer exakt eingehalten werden. Mißt man die Performance des Systems auf einer täglichen Basis, gilt weiter zu beachten, daß man damit auch die stark schwankende Verteilung der Nachrichten(themen) mitmißt. Eine weitere Schwierigkeit stellt der „wankelmütige“ Benutzer dar. Würde er die gleiche Nachrichtenliste zu einem späteren Zeitpunkt noch einmal zur Beurteilung erhalten, könnte es ein, daß er anders urteilt und andere Relevanzurteile vergibt.

Billsus und Pazzani entschieden sich zu folgendem Vorgehen. Die Trainingsdaten jedes Nutzers wurden in einzelne Sets unterteilt (engl. training sessions), entsprechend dem Nutzungsverhaltens des Anwenders. Die Regel war eine Trainingseinheit pro Tag. Das System wurde tageweise mit den Eingaben des Nutzer gefüttert und die Performance gemessen. Zuletzt wurde noch über alle Nutzer gemittelt. Dieses Vorgehen kommt einer realen Nutzung des Systems am nächsten, da alle vorhandenen Trainingsdaten genutzt wurden, um Nachrichten zu klassifizieren.

Zur Beschreibung der Systemleistung kamen die im Information Retrieval gebräuchlichen Maße Precision und Recall zum Einsatz. Außerdem ein sogenanntes „F-Maß“ (engl. F-measure), das die beiden erstgenannten in einer Maßzahl kombiniert. Dies ist wichtig, da Precision und Recall einzeln relativ leicht auf Kosten des anderen optimiert werden können [He03]. Daher ist stets eine gemeinsame Betrachtung durchzuführen. Das F-Maß geht auf Lewis und Gale [LG94][28] zurück und ist eine gewichtete Kombination von Precision und Recall. Im Falle des hier verwendeten F1-Maßes wird beiden gleiches Gewicht eingeräumt. Die Formel bildet auf den Bereich von 0 bis 1 ab und lautet wie folgt:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Die nachfolgende Abbildung 5 zeigt die Gesamtperformance des Systems, gemittelt über alle Testnutzer. Auffällig ist ein sehr starker Leistungsanstieg innerhalb der ersten drei Trainingseinheiten. Da nur wenige Nutzer mehr als drei Trainingseinheiten absolviert haben, können die weiteren Ergebnisse leider nicht im selben Graphen gezeigt werden. Es zeigte sich jedoch, daß die Performance zu Beginn immer stark anstieg und sich dann auf hohem Niveau einpendelte, da jeden Tag unterschiedliche Nachrichten das System erreichten. Sehr schön zu sehen ist in Abbildung 5 auch das Zusammenspiel der beiden Nutzermodelle. Das hybride Modell erreichte stets bessere Werte als die beiden Modelle alleine. Bei genauerer Untersuchung zeigte sich, daß das kurzfristige Modell zwar eine hohe Precision, jedoch nur einen geringen Recall aufweist. Im Gegensatz dazu die Stärken des langfristigen Modells: Niedrige Precision, dafür guter Recall. Die Kombination zum hybriden Modell erlaubt es nun, von den Stärken beider Modelle zu profitieren.

[28] Der eigentliche Ursprung des F-Maßes liegt sogar noch früher, es geht auf ein Buch von Van Rijsbergen zurück, der eine solche Kombination aus Precision und Recall erstmals als „E-measure“ in [Ri79] einführt.

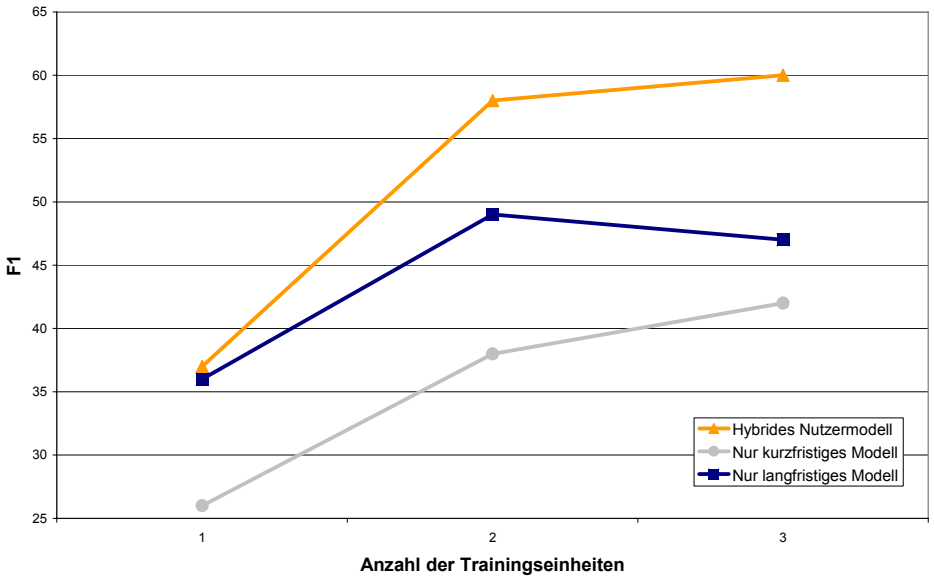


Abbildung 12: Gemittelte Leistung des Gesamtsystems

Um den Beitrag des zeit-kodierten Feedbacks zur Gesamtleistung zu messen, wurden die ermittelten Zeitwerte (*pl*) in Bool-Werte konvertiert und die Differenz gemessen. Betrachtet man nur die Klassifikationsgenauigkeit sowie F1, ließ sich kein großer Unterschied feststellen. Dies scheint logisch, da dies nur dann der Fall wäre, wenn eine Nachricht aufgrund eines zeit-kodierten Feedbacks ihre Klassenzugehörigkeit ändern würde. Die Hauptauswirkung dieses Feedback-Typs tritt in der relativen Reihenfolge der Nachrichten in der Recommendation Queue zutage. Hier zeigte sich, daß die Precision der Top 5-Nachrichten bei Verwendung des zeit-kodierten Feedbacks um gute 8% zulegen konnte. Sichtbar wird dies in Abbildung 6.

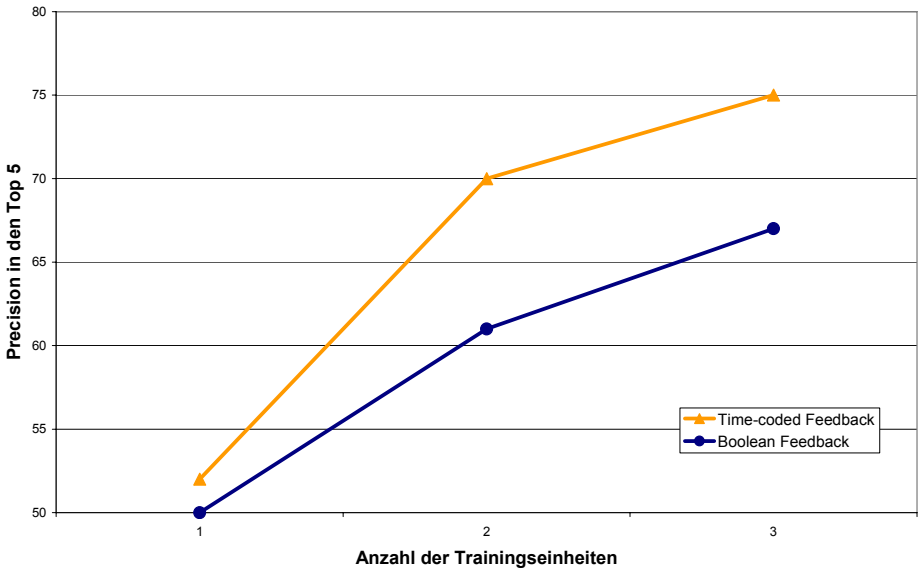


Abbildung 13: Einfluß des zeit-kodierten Feedback

Zu guter Letzt wurde noch der Effekt des Konzeptfeedbacks auf die Gesamtperformance ausgewertet. Dieser läßt sich schwer visualisieren, da der Einfluß stark von der Zahl der Rückmeldungen des einzelnen Nutzers abhängt. Billsus und Pazzani behalfen sich damit, alle eingegebenen Trainingsdaten eines Testers zur Vorhersage zu nutzen, um damit auf die Klassenzugehörigkeiten der Nachrichten aus der letzten Trainingseinheit zu schließen. Gemessen wurde die Performance mit und ohne Konzeptfeedback-Aktionen und gemittelt über alle Testnutzer. Es zeigte sich, daß die Genauigkeit mit Konzeptfeedback von 72,5% auf 77,1% zulegen konnte, F1 in ähnlicher Größenordnung immerhin von 60,1% auf 64,7%.

7 Zusammenfassung und Fazit

Billsus und Pazzani stellen mit ihrem „News Dude“ eine Audio-Agenten vor, der in vielen Bereichen Neuland betritt. Dies beginnt bei der audiobasierten Präsentation der Nachrichten, geht über die Integration zweier Nutzermodelle zur Repräsentation der Nutzerinteressen, weiter über die Erklärungskomponente für Empfehlungen bis hin zum Konzeptfeedback, das erstmalig direkte Eingriffe in das verwendete Modell ermöglicht. Sie zeigen, daß sich das kurzfristige und das langfristige Modell sehr gut ergänzen und daß eine explizit Modellierung bereits bekannter Nachrichten sich positiv in der Systemleistung bemerkbar macht. Damit gelingt es auch, die Forderung Belkins aus [Be97] umzusetzen: Der Agent „News Dude“ berücksichtigt, daß sich das Informationsbedürfnis des Nutzers durch Interaktion mit Informationen verändert. Ähnliche Nachrichtentexte werden nicht mehrfach präsentiert, dies erhöht die Treffsicherheit und somit die Vorhersagekraft des Systems.

Literaturverzeichnis

[AC+98] Allan, J., Carbonell, J.G., Doddington, G., Yamron, J. and Yang, Y. Topic Detection and Tracking Pilot Study Final Report, Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, Virginia, 1998.

[Be97] N. Belkin, User Modeling in Information Retrieval, http://www.scils.rutgers.edu/~belkin/um97oh/, Tutorial Overheads, Sixth International Conference on User Modeling, Chia Laguna, Sardinia, 1997.

[BP97] D. Billsus, M. J. Pazzani. Learning and Revising User Profiles: The identification of interesting web sites. Machine Learning 27, 313-331, 1997.

[BP99a] D. Billsus, M. J. Pazzani. A hybrid user model for news story classification. In Proc. of the 7th Intl. Conf. On User Modeling., Banff, Canada, June 20-24 1999.

[BP99b] D. Billsus, M. J. Pazzani. A personal news agent that talks, learns and explains. In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, Proc. of the 3rd Intl. Conf. on Autonomous Agents (Agents'99), pages 268–275, Seattle,WA, USA, 1999. ACM Press.

[CH98] W. Cohen and H. Hirsh. Joins that generalize: text classification using WHIRL. In Proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining, New York, New York, 169-173, 1998.

[CW98] B. Chiu and G. Webb. Using decision trees for agent modeling: improving prediction performance. User Modeling and User-Adapted Interaction 8:131-152, 1998.

[He03] A. Henrich. Information Retrieval. Begleitlektüre zum Kurs Information Retrieval an der Universität Bamberg., Bamberg, 2003.

[KG00] A. Kaban, M. Girolami. Initialized and Guided EM-Clustering of Sparse Binary Data with Application to Text Based Documents. International Conference on Pattern Recognition (ICPR'00)-Volume 2. Barcelona, Spain, p. 2744, 2000.

[LG94] D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 1994, London, Springer Verlag.

[Li95] H. Lieberman. Letizia: An agent that assists web browsing. Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, August 1995, 924-929.

[Ri79] C. J. van Rijsbergen. Information Retrieval. Butterworths, London, second edition, 1979.

[Sa89] G. Salton. Automatic Text Processing. Addison-Wesley, 1989.

[SB88] G. Salton, C. Buckley. Term-weighted approaches to automatic text retrieval. Inf. Proc. & Mngmt., 24(5):513-523, 1988.

Online-Hilfe bei Anwendungsprogrammen

Dominik Englert, René Katzenberger

dominik.englert@stud.uni-bamberg.de
rene.katzenberger@stud.uni-bamberg.de

Danksagung: An dieser Stelle danken wir Dr. E. Horvitz für die freundliche Unterstützung.

Abstract: Im Rahmen des Lumière-Projekts [Ho98] wurde ein Prototyp eines benutzeradaptiven Assistenzsystems zur Tabellenkalkulationssoftware EXCEL des Unternehmens Microsoft entwickelt. Dieses Assistenzsystem soll den Nutzer bei seiner Aufgabendurchführung unterstützen, indem es ihm Hilfstexte anbietet, ohne dass der Nutzer diese explizit angefordert hat. Dazu werden dynamische Bayes' sche Netze in Kombination mit Einflussdiagrammen verwendet. Auf Basis der von einem Nutzer getätigten Aktionen innerhalb des Anwendungssystems sowie dem aktuellen Systemzustand der Tabellenkalkulationssoftware werden die Nutzerziele inferiert und es wird mittels entsprechender Situationsbewertungen eingeschätzt. ob der Nutzer zu dem jeweiligen Zeitpunkt von einer Hilfe profitieren wird. Horvitz et al. präsentieren weiterhin ein System auf Basis temporaler Logik. Dabei werden temporale Aspekte durch Variablen im Modell abgebildet. Der Office Assistent in den Microsoft Office '97 Anwendungen entstammt der Lumière Forschung, wenn auch in wesentlich vereinfachter Form.

1 Einführung

Eric Horvitz ist Senior Researcher und Group Manager im Bereich „Adaptive Systems & Interaction" des Unternehmens Microsoft. Die Schwerpunkte seiner Forschung liegen in den Bereichen intelligente Wahrnehmung, Schlussfolgerung und Aktion. Eric Horvitz beschäftigt sich in diesem Kontext ebenso mit Unvollständigkeit in Repräsentationen sowie Unsicherheit bzgl. Umgebungen oder Situationen. Dabei versuchen die Microsoft Forscher die Wahrscheinlichkeitstheorie, die Entscheidungstheorie, die Entscheidungsanalyse und insbesondere die Verwendung von Bayes' sche sowie Entscheidungstheoretische Prinzipien bei der Konstruktion und Erforschung von Schlussfolgerungssystemen einzusetzen. Dieser Forschungsbereich umfasst nicht nur theoretische Problemstellungen, sondern auch konkrete Anwendungen, wie beispielsweise Betriebssysteme, Information Retrieval-Systeme und auch Kommunikationssysteme.


Abbildung 14 und 2: Dr. Eric Horvitz (Quelle: Microsoft)

Im Rahmen des Lumière-Projekts entwickelten Dr. Horvitz et al. einen Prototyp eines benutzeradaptiven Assistenzsystems zur Tabellenkalkulationssoftware EXCEL des Unternehmens Microsoft. Dieses Assistenzsystem soll den Nutzer bei seiner Aufgabendurchführung unterstützen, indem es ihm Hilfstexte anbietet, ohne dass der Nutzer diese explizit angefordert hat. Um dies zu erreichen, musste das Forschungsteam um Eric Horvitz zwei wesentliche Probleme lösen. Einerseits stellt sich die Frage wie sich die aktuellen Ziele des Nutzers erkennen lassen, d.h. was will der Nutzer mit seinen bisherigen Aktionen erreichen. Andererseits müssen entsprechende Zeitpunkte ermittelt werden, zu denen das Assistenzsystem dem Nutzer eine geeignete Hilfestellung leistet. Um diese Probleme zu lösen werden dynamische Bayes' sche Netze in Kombination mit Einflussdiagrammen verwendet. Auf Basis der von einem Nutzer getätigten Aktionen innerhalb des Anwendungssystems sowie dem aktuellen Systemzustand der Tabellenkalkulationssoftware werden die Nutzerziele inferiert und es wird mittels entsprechender Situationsbewertungen eingeschätzt. ob der Nutzer zu dem jeweiligen Zeitpunkt von einer Hilfe profitieren wird. Neben der Modellierung unter Verwendung dynamischer Bayes' scher Netze werden temporale Aspekte explizit durch Variablen im Modell abgebildet.

1.1 Der historische Verlauf

- 1993: Initiierung des Lumière-Projekts
- 1994: Der anfängliche Lumière/Excel Prototyp wird erstmals im Januar der Microsoft Office Abteilung präsentiert.
- 1997: Der Office Assistent in den Microsoft Office '97 Anwendungen resultiert aus den Ergebnissen der Lumière Forschung.
- 2004: Assistent in Office XP oder Office 2003 und Windows XP.

Abbildung 3: Kerzenständer Lumière (Quelle: Die Schöne und das Biest, Walt Disney 1991)

Das Projekt wurde aus zwei Gründen „Lumière-Projekt“ genannt. Einerseits sollte der Prototyp Licht in die Dunkelheit um die Bayes’ sche Nutzermodellierung bringen (franz.: lumière = Licht). Andererseits wurde das Projekt nach einem Charakter, dem Butler Lumière, aus dem Walt Disney Film „die Schöne und das Biest“ aus dem Jahr 1991 benannt.

1.2 Benutzeradaptive Systeme

In den vergangenen Jahren ist ein wachsendes Interesse an der Verwendung Bayes’ scher Netze und Entscheidungstheoretischer Methoden im Bereich der intelligenten Systeme zu beobachten. Bayes’ sche Netze werden dabei zum Schlussfolgern unter Unsicherheit eingesetzt und bilden somit die Inferenzkomponente dieser Systeme. Ziel ist es Systeme zu konstruieren, die auf den Nutzer reagieren und sich auf dessen Bedürfnisse einstellen. Ein sog. benutzeradaptives System passt sein Verhalten an den individuellen Nutzer auf der Grundlage nicht-trivialer Inferenzen über Information über den Nutzer an. Mittlerweile ist ein immer stärker wachsender Bedarf an benutzeradaptiven Systemen zu beobachten. Jameson et al. nennen eine Reihe von Gründen, die diese Entwicklung hervorrufen [Ja02]:

- Vielzahl unterschiedlicher Benutzer und Anwendungskontexte: Die Entwicklung von Anwendungen, denen die Fähigkeit fehlt auf einen Nutzer zu reagieren und sich auf dessen Bedürfnisse einzustellen, die aber gleichzeitig unterschiedlichen Nutzern gerecht werden müssen, wird immer schwieriger.

- Anzahl und Komplexität interaktiver Systeme: Die verfügbare Anzahl an neuen Systemen wächst permanent an. Aus diesem Grund kann von einem Nutzer nicht erwartet werden, dass er sich in ein neues System zeitintensiv einarbeitet, bis er die für ihn relevanten Funktionen und Optionen erlernt hat.

- Vielzahl unterschiedlicher Informationstypen: Ein Nutzer hat die Möglichkeit auf eine Vielzahl von Informationsobjekten, z.B. Texte, zuzugreifen. Diese Informationsobjekte können für einen bestimmten Nutzer relevant oder nicht relevant sein. Dabei erscheint es sinnvoll den Umgang mit diesen Objekten durch das System durchzuführen. Ein Beispiel hierfür ist ein Information Retrieval-System, das eine Dokumentkollektion in für den Nutzer relevante und nicht relevante Dokumente, bzgl. einer Anfrage, unterteilt.

Die Basis eines derartigen benutzeradaptiven Systems ist eine Wissensquelle, die explizite Annahmen über alle Aspekte des Nutzers enthält, die für das Interaktionsverhalten eines Systems relevant sein könnten: das Benutzermodell [WK89]. Dieses Benutzermodell kann entsprechend der Interaktion zwischen Nutzer und System verändert werden, d.h. es kann erweitert und auch angepasst werden. Somit besteht für das benutzeradaptive System die Möglichkeit die Interaktion mit dem Nutzer individuell zu gestalten.

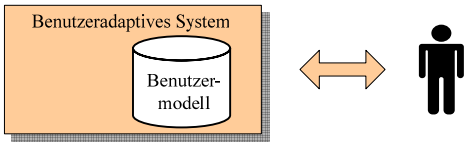


Abbildung 4: Allg. Schema eines benutzeradaptiven Systems

1.3 Bayes' sche Netze in benutzeradaptiven Systemen

Bayes' sche Netze [Pe88] finden eine immer stärkere Verwendung im Bereich der Repräsentation von Benutzermodellen in benutzeradaptiven Systemen. Wittig (2002) zeigt die folgenden vorteilhaften Eigenschaften Bayes' scher Netze, in Bezug auf benutzeradaptive Systeme, auf:

- Bayes' sche Netze ermöglichen den Umgang mit Unsicherheit in der jeweiligen Domäne. Unsicherheit in der Domäne ist ein wesentliches Merkmal in der Benutzermodellierung.

- Außerdem können die gerichteten Kanten eines Bayes' schen Netzes kausal interpretiert werden. Dies vereinfacht die Konstruktion und Interpretierbarkeit eines Nutzermodells.

- Bayes' sche Netze stellen eine kompakte Repräsentation einer gemeinsamen Wahrscheinlichkeitsverteilung über der Menge der relevanten Variablen einer Domäne dar. Bayes' sche Netze ermöglichen es probabilistische Beziehungen zwischen den Variablen zu modellieren.

- Bayes' sche Netze besitzen die Fähigkeit, Wahrscheinlichkeitsverteilungen über beliebige Teilmengen der betrachteten Variablen konditioniert auf die verfügbare Information zu liefern. Andere Formalismen, wie beispielsweise Entscheidungsbäume, erlauben oftmals lediglich Aussagen über eine festgelegte Menge an Vorhersagevariablen.

- Eine weitere vorteilhafte Eigenschaft Bayes' scher Netze findet sich in der relativ einfachen Erweiterung zu Einflussdiagrammen[29]. Dadurch haben benutzeradaptive Systeme die Möglichkeit Entscheidungen, welche die verfügbare Information berücksichtigen, anhand des Benutzermodells zu treffen.

- Temporale Aspekte können mit dynamischen Bayes' schen Netzen modelliert werden. Dadurch besteht die Möglichkeit zeitlich variable Bereiche des Benutzermodells zu repräsentieren.

- Weiterhin existieren maschinelle Lernverfahren für Bayes' sche Netze [He98], die die Möglichkeit einräumen, gesammelte Daten zur Konstruktion und Pflege eines Benutzermodells in Form eines gelernten bzw. permanent aktualisierten Bayes' schen Netzes auszunutzen.

1.4 Das Ziel des Lumière-Projekts

Das übergeordnete Ziel, welches das Forscherteam um Eric Horvitz verfolgte ist durch das Einflussdiagramm in Abbildung 5 dargestellt, welches die wesentlichen Aspekte der Nutzermodellierung sowie automatisierter Assistenzsysteme im Bereich von Anwendungssystemen abdeckt. Betrachtet werden die Ziele sowie die Bedürfnisse eines Nutzers. Ziele sind die durch den Nutzer fokussierten und durchzuführenden Aufgaben bzw. Teilaufgaben. Bedürfnisse sind in diesem Zusammenhang Informationen aber auch automatisierte Aktionen, die die Zeit bzw. den Aufwand reduzieren, welche bzw. welcher notwendig ist, um die entsprechenden Ziele zu erreichen.

[29] Eine Einführung in die Thematik der Einflussdiagramme ist unter 2.3 zu finden.

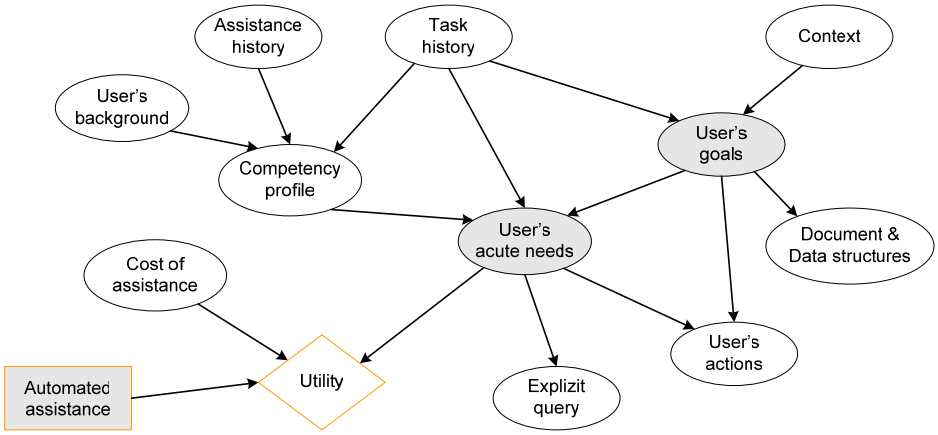


Abbildung 5: Einflussdiagramm [Ho98]

Die Bedürfnisse eines Nutzers werden sowohl durch seine Ziele als auch durch seine Kompetenz bzgl. der Softwarenutzung beeinflusst. Angedeutet wird dies durch die Variablen und Abhängigkeiten im Einflussdiagramm. Die Kompetenz des Nutzers wird nun ihrerseits durch den Hintergrund des Nutzers, z.B. seine Erfahrung mit Software Anwendungen, und durch vorhergehende Assistenzsysteme, z.B. eine Online-Hilfe, beeinflusst. Die Bedürfnisse eines Nutzers haben einen direkten Einfluss auf seine Aktionen. Sobald der Nutzer mit dem System interagiert, beispielsweise mittels Maus- oder Tastatureingaben, können die Folgen von Nutzeraktionen, also seine Aktivitäten, vom System aufgezeichnet werden. Hieraus lassen sich spezifische Aktivitätsmuster erkennen. Weiterhin beeinflussen die Ziele des Nutzers u.a. die Menge der aktuell verwendeten Dokumente.

In vielen Fällen fragt der Nutzer explizit eine Hilfestellung nach. Dabei werden die durch den Benutzer in der Anfrage an das System explizit verwendeten Begriffe direkt durch seine Bedürfnisse beeinflusst. Wie im Einflussdiagramm angedeutet besteht das übergeordnete Ziel darin, dem Nutzer ein automatisiertes Assistenzsystem zur Seite zu stellen, das den erwarteten Nutzen des Nutzers optimiert.

2 Grundlagen Bayes' scher Netze

Bevor wir uns weiter mit dem Lumière-Projekt beschäftigen soll an dieser Stelle ein Überblick über die theoretischen Grundlagen vermittelt werden. Hierzu zählen insbesondere die bedingte Wahrscheinlichkeit, Bayes' sche Netze, Einflussdiagramme sowie dynamische Bayes' sche Netze.

2.1 Bedingte Wahrscheinlichkeit

Die bedingte Wahrscheinlichkeit P(A | B) („Wahrscheinlichkeit für das Eintreten des Ereignisses A unter der Bedingung, dass das Ereignis B eingetreten ist oder gleichzeitig eintritt.") [Vo00]:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}; \text{ für } P(B) > 0.$$

Entsprechend ist: $$P(B \mid A) = \frac{P(A \cap B)}{P(A)}; \text{ für } P(A) > 0.$$

Die bedingten Wahrscheinlichkeiten genügen den Kolmogorov' schen Axiomen.[30]

Löst man diese Definition auf, so erhält man mit

$$P(A \cap B) = P(A \mid B) \cdot P(B)$$

$$= P(B \mid A) \cdot P(A)$$

das Produktgesetz der Wahrscheinlichkeit [Vo00]. Diese Gleichung kann man für n Variablen verallgemeinern. Dadurch ergibt sich die Kettenregel, die die Berechnung der gemeinsamen Wahrscheinlichkeitsverteilung von n Variablen ermöglicht. $A_1,\ldots,A_n$ seien Zufallsvariablen, dann gilt:

$$P(A_1 \cap \ldots \cap A_n) = \prod_{i=1}^{n-1} (P(A_i \mid A_{i+1} \cap \ldots \cap A_n) \cdot P(A_n))$$

Gemäß dem Produktgesetz gilt:

$$P(A \mid B) \cdot P(B) = P(A \cap B) = P(B \cap A) = P(B \mid A) \cdot P(A)$$

[30] Siehe beispielsweise [Vo00], Kapitel 10.

Mittels einfacher Umstellung erhält man das Bayes-Theorem [Pu91]:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Das Bayes-Theorem trägt bei der Ermittlung der bedingten Wahrscheinlichkeiten eine wesentliche Bedeutung. Ist die bedingte Wahrscheinlichkeit P(A | B) im Bayes' schen Netz nicht verfügbar, so lässt sich diese mit dem Bayes-Theorem beispielsweise aus der bedingten Wahrscheinlichkeit P(B | A) ableiten.

Eine zentrale Bedeutung für Bayes' sche Netze nimmt die Theorie der bedingten Unabhängigkeit ein. Zwei Zufallsvariablen A und B heißen unabhängig, falls gilt:

$$P(A) = P(A \mid B)$$

Zwei Zufallsvariablen A und B heißen bedingt unabhängig bezüglich einer dritten Zufallsvariablen C, falls gilt:

$$P(A \mid C) = P(A \mid C \cap B)$$

Unabhängigkeit bedeutet, dass sich die Variablen nicht beeinflussen, d.h. die Wahrscheinlichkeitsverteilung der einen Zufallsvariable hängt nicht von der der anderen Zufallsvariable ab. Bedingte Unabhängigkeit bedeutet, dass zwei Zufallsvariablen bezüglich einer bestimmten dritten Zufallsvariable unabhängig sind.

2.2 Bayes' sche Netze

Die Grundlagen Bayes' scher Netze wurden 1763 von Thomas Bayes, einem englischen Mathematiker, presbyterianischen Priester und Schüler von de Moivre, in seinem Werk „An essay towards solving a problem in the doctrine of chances" geschaffen.

Abbildung 6: Thomas Bayes (1702-1761)

Die Konzepte für heutige Bayes' sche Netze lassen sich auf Judea Pearl [Pe88] zurückführen. In der ersten Hälfte der 1980er Jahre wurden Bayes' sche Netze in den Bereich der Expertensysteme eingeführt. Gegen Ende der 1980er fanden sie erstmals Eingang in Real-World-Anwendungen.

2.2.1 Wesentliche Begriffe

- Ein gerichteter Graph G = (V,E) besteht aus einer Menge V = $\{v_1,\ldots,v_n\}$ von Knoten und einer Menge E = $\{e_1,\ldots,e_m\}$ von Pfeilen. Ein Paar $(v_i,v_j) \in E$ heißt Pfeil (= gerichtete Kante) von v_i nach v_j. v_i ist dabei Anfangs- und v_j Endknoten des Pfeils. Weil E eine Menge ist, kann in diesen Graphen jeder Pfeil höchstens einmal auftreten, d.h. es sind keine parallelen Pfeile erlaubt [OW96].

- Eine Folge $v_0,\ldots,v_k$ von Knoten, die die Bedingung erfüllt, dass v_{i+1} Sohn bzw. Kindknoten von v_i ist für $0 \leq i < k$, heißt Pfad mit Länge k, der v_0 mit v_k verbindet [OW96].

- Ein Zyklus ist ein Pfad, der am Ausgangsknoten endet, also ein Pfad von einem Knoten v nach v. Ein Graph, heißt zyklenfrei oder azyklisch, wenn er keinen Zyklus enthält [OW96].

- v_i ist Vorgänger, auch Elternteil genannt, von v_j, genau dann wenn ein Pfeil $(v_i,v_j) \in E$ existiert. v_j ist Nachfolger, auch Sohn bzw. Kindknoten genannt, von v_i [OW96]. Die Menge aller Elternteile eines Knotens v_j wird mit $Pa(v_j)$ bezeichnet [Wi02].

2.2.2 Definition

Ein Bayes' sches Netz ist ein gerichteter azyklischer Graph, dessen Knoten Zufallsvariablen und dessen Kanten kausale Abhängigkeiten zwischen diesen Zufallsvariablen darstellen. Die Stärke der kausalen Abhängigkeiten wird dabei durch bedingte Wahrscheinlichkeiten angegeben. Die Vorgänger eines Knoten, der in diesem Sinne selbst als Sohn bzw. Kindknoten betrachtet wird, in einem Bayes' schen Netz sind dabei die Knoten, welche als direkte Ursachen für den Kindknoten betrachtet werden können [He03].

Definition (Bayes' sches Netz) [Wi02]: Ein Bayes' sches Netz $B = (G,\theta)$ für eine Menge X = $\{X_1,\ldots,X_n\}$ von Zufallsvariablen besteht aus zwei Teilen:

- Einem gerichteten azyklischen Graphen G = (X,E), dessen Knoten den Zufallsvariablen entsprechen und mit dessen Kanten die bedingten Unabhängigkeiten zwischen den Variablen kodiert werden. Man spricht von G als der Struktur von G.

- Einer Menge $\theta = \{\theta_1,\ldots, \theta_n\}$ von mit den Variablen assoziierten Tabellen bedingter Wahrscheinlichkeiten (conditional probability tables, CPTs) $\theta i = P(X_i \mid Pa(X_i))$, $i = 1,\ldots,n$. Sie beinhalten als Einträge die bedingten Wahrscheinlichkeiten $\theta_{ijk} = P(x_{ij} \mid pa_k(X_i))$ der n_i Zustände x_{ij}, $j = 1,\ldots,n_i$, der Variablen X_i konditioniert auf die möglichen Zustandskombinationen $pa(X_i)$ der Eltern $Pa(X_i)$. Besitzt ein Knoten keine Eltern, dann beinhaltet seine CPT unbedingte a-priori-Wahrscheinlichkeiten $P(x_{ij})$.

Die bedingten Unabhängigkeiten zwischen Variablen einer Domäne werden in der Struktur des Netzes durch das folgende Unabhängigkeits-Kriterium kodiert: Sind die Zustände der Elternvariablen $Pa(X_i)$ bekannt, dann ist eine Variable X_i unabhängig von den Zuständen ihrer Nicht-Nachfolger in der Struktur des Bayes' schen Netzes. Aus der Struktur eines Bayes' schen Netzes lassen sich unter Verwendung des d-Separationskriteriums weitere Unabhängigkeitsannahmen ablesen [Pe88]. Die d-Separation erlaubt eine allgemeine Aussage darüber, ob eine Knotenmenge unabhängig von einer anderen Knotenmenge ist.

Definition (d-Separation): Zwei unterschiedliche Variablen A und B sind d-separated (direction-dependent-separated), falls auf allen Pfaden zwischen A und B eine Variable Z existiert, so dass:

- die Verbindung seriell oder divergierend und Z ein Evidenzknoten ist, oder
- die Verbindung konvergierend und weder Z noch die Nachfolger von Z Evidenzknoten sind.

Sind zwei Knoten nicht d-separated, so werden sie auch als d-connected bezeichnet.

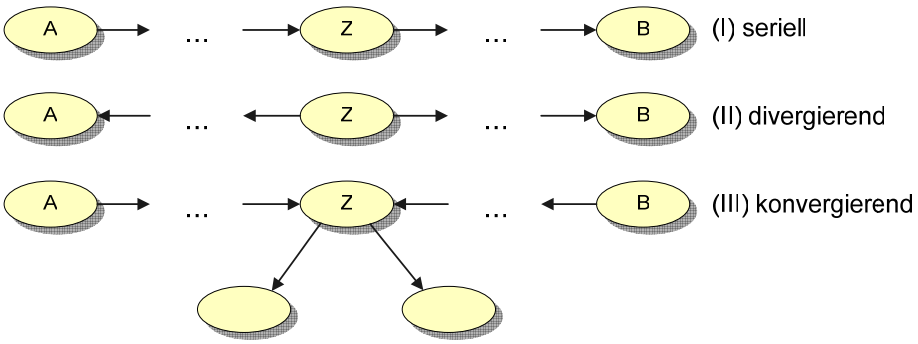


Abbildung 7: d-Separation (Verbindungsmöglichkeiten)

Aus der Definition eines Bayes' schen Netzes ist ersichtlich, dass jeder Pfeil innerhalb des Bayes' schen Netzes eine bedingte Abhängigkeit bzgl. aller Vorgänger darstellt. Somit ergeben sich aus dem Bayes' schen Netz Angaben zu den bedingten Wahrscheinlichkeiten, die notwendig sind um die Kettenregel der Wahrscheinlichkeit anwenden zu können, so dass sich die gesamte Verbundwahrscheinlichkeit darstellen lässt. Zur Veranschaulichung sei folgende Abbildung gegeben.

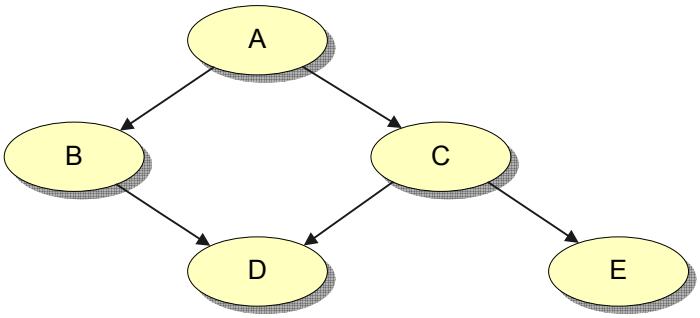


Abbildung 8: Beispiel eines Bayes' schen Netzes [He03] [31]

Die Abbildung 8 stellt ein Bayes' sches Netz für die verbundene Wahrscheinlichkeit P(a, b, c, d, e) dar. Die verbundene Wahrscheinlichkeitsverteilung wird nun durch lokale bedingte Wahrscheinlichkeiten angegeben:

$$P(a,b,c,d,e) = P(a) \cdot P(b \mid a) \cdot P(c \mid a) \cdot P(d \mid b,c) \cdot P(e \mid c)$$

Die Wahrscheinlichkeit P(a) wird auch als Ausgangswahrscheinlichkeit bezeichnet. Für jede Knotenvariable ist eine bedingte Wahrscheinlichkeit notwendig. Die direkten Vorgänger einer Knotenvariablen treten als Bedingung in der bedingten Wahrscheinlichkeit der Knotenvariablen auf. Für die Wurzelknoten müssen unbedingte a-priori-Wahrscheinlichkeiten gegeben sein.

2.2.3 Ein Beispiel

An dieser Stelle wollen wir ein einfaches Beispiel für ein Bayes' sche Netz darstellen. Hierzu soll die Motivation eines Studenten in Bezug auf sein Lernverhalten im Vorfeld einer Klausur untersucht werden. Dabei kommen folgende Variablen zum Einsatz, die lediglich binär sind. Die Variable „Klausurstoff" kann die Ausprägung „interessant" oder „uninteressant" annehmen. Eine „Klausur" wird entweder bewertet, z.B. in Form einer Diplomklausur, oder nicht bewertet, hierbei kann es sich beispielsweise um einen unbewerteten Schein handeln. Weiterhin wird die Variable „Nervosität" verwendet, die die Ausprägung „ja" oder „nein" annehmen kann. Abschließend sei noch die Variable „Motivation zu Lernen" genannt, welche „hoch" oder „niedrig" sein kann. Zunächst erfolgt eine Auflistung der in diesem Modell enthaltenen Zufallsvariablen:

Klausurstoff:	A ∈ {interessant, uninteressant},
Klausur:	B ∈ {bewertet, nicht bewertet},
Nervosität:	C ∈ {ja, nein},
Motivation zu Lernen:	D ∈ {niedrig, hoch}.

[31] Es gibt eine Reihe von verschiedenen Applikationen zur Modellierung Bayes' scher Netze. Für die vorliegende Arbeit haben wir die Software „Hugin Development Environment" verwendet, um neben Bayes' sche Netze auch Einflussdiagramme zu modellieren.

Nun lassen sich die Knoten in Wirkungsrichtung sortieren und mit den Abhängigkeiten versehen, dabei darf kein Zyklus entstehen. Die Variablen Klausurstoff und Klausur haben einen direkten Einfluss auf die Variable Motivation zu Lernen. Weiterhin wird Nervosität durch Klausur beeinflusst. Hierbei ist es relativ einfach die Kausalstruktur, also den qualitativen Aspekt, zu beschreiben, aufgrund der kausalen Interpretierbarkeit der Pfeile.

Die beschriebene Struktur wird durch das Bayes' sche Netz in Abbildung 9 dargestellt.

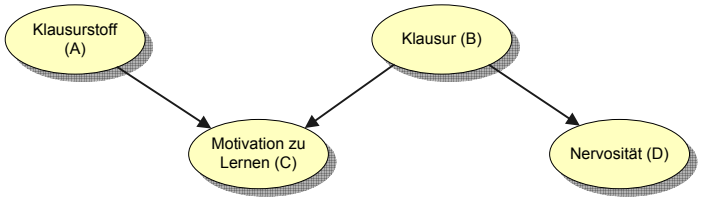


Abbildung 9: Bayes' sches Netz

Im weiteren Verlauf müssen nun die conditional probability tables (CPTs), also der quantitative Aspekt eines Bayes' schen Netzes, erstellt werden. Dabei ordnen die CPTs jedem Knoten seine bedingte Wahrscheinlichkeitsverteilung zu. In Bezug auf das Beispiel sind die Wahrscheinlichkeitsverteilungen P(A), P(B), P(D | B) und P(C | A ∩ B) in jeweils einer CPT zu spezifizieren. Dies gestaltet sich wesentlich schwieriger, da hierfür eine Festlegung auf spezifische Werte erfolgen muss.

Klausurstoff:

interessant	0,50
uninteressant	0,50

Klausur:

bewertet	0,50
nicht bewertet	0,50

Nervosität:

Klausur	bewertet	nicht bewertet
ja	0,90	0,20
nein	0,10	0,80

Motivation zu Lernen:

Klausurstoff	interessant		uninteressant	
Klausur	bewertet	nicht bewertet	bewertet	nicht bewertet
niedrig	0,20	0,40	0,40	0,90
hoch	0,80	0,60	0,60	0,10

Tabelle 1: CPTs der Variablen des Bayes' schen Netzes

Den CPTs liegen die qualitativen Annahmen zugrunde, dass sowohl ein interessanter Klausurstoff, als auch eine Bewertung in der Klausur zu einer höheren Motivation zum Lernen führen. Entsprechend gelten die umgekehrten Zusammenhänge.

Das so modellierte Bayes' sche Netz repräsentiert nun eine probabilistische Wissensbasis. Die verbundene Wahrscheinlichkeitsverteilung dieser Wissensbasis lässt sich mittels der Kettenregel der Wahrscheinlichkeit berechnen, wobei lediglich die bedingten Wahrscheinlichkeiten, die den einzelnen Knoten zugeordnet sind, miteinander multipliziert werden müssen:

$$P(A,B,C,D) = P(A) \cdot P(B) \cdot P(C \mid A \cap B) \cdot P(D \mid B)$$

Nachdem die Wissensbasis durch ein Bayes' sches Netz erstellt worden ist, lässt sie sich anwenden, um probabilistische Schlussfolgerungen zu ziehen. Dabei stellt eine Anfrage an die Wissensbasis letztendlich eine Berechnung von Verbundwahrscheinlichkeiten dar. Die Aufgabe des Inferenzverfahrens für Bayes' sche Netze besteht nun darin, anhand von partiellen Beobachtungen in der betrachteten Domäne, den so genannten Evidenzen, Aussagen über andere Teile der Domäne zu machen [Wi02]. D.h. Variablen eines Modells gehen in einen bestimmten Zustand und auf dieser Grundlage will man nun Wahrscheinlichkeiten für die Zustände anderer Variablen, die nicht beobachtet werden, ermitteln. Dies wird dadurch erreicht, indem eine Wahrscheinlichkeit für jede beliebige Zustandskombination einer Teilmenge dieser Variablen durch Marginalisieren ($\approx$Heraussummieren) aus der expliziten Repräsentation berechnet werden kann [Wi02].

Im Folgenden werden wir beispielhaft einige Berechnungen vornehmen. Nehmen wir an, dass C = hoch gilt. Nun lassen sich die a-posteriori Wahrscheinlichkeiten ($\approx$ „nach Beobachtung", d.h. mit Wissen über die Zustände bestimmter Variablen, in diesem Fall der Variable „Motivation zu Lernen") der verbleibenden Variablen folgendermaßen berechnen. Wir suchen die Wahrscheinlichkeit für A = interessant:

$$P(A = \text{interessant} \mid C = hoch)$$
$$= \frac{P(C = hoch \mid A = \text{interessant}) \cdot P(A = \text{interessant})}{P(C = hoch)}$$

Da $P(A \mid B) = \frac{P(A \cap B)}{P(B)}$; für $P(B) > 0$; und $P(A \cap B) = P(A \mid B) \cdot P(B)$ gilt:

$$\Rightarrow P(A = \text{interessant} \mid C = hoch) = \frac{P(C = hoch \cap A = \text{interessant})}{P(C = hoch)}$$

Die Wahrscheinlichkeit P(C = hoch) lässt sich durch Marginalisieren berechnen:

$$P(C = hoch) = \sum_{A,B} P(C = hoch \mid A \cap B) \cdot P(A) \cdot P(B)$$

$$P(C = hoch) = 0{,}8 \cdot 0{,}5 \cdot 0{,}5 + 0{,}6 \cdot 0{,}5 \cdot 0{,}5 + 0{,}6 \cdot 0{,}5 \cdot 0{,}5 + 0{,}1 \cdot 0{,}5 \cdot 0{,}5 = \underline{\underline{0{,}525}}$$

Weiterhin müssen die restlichen bedingten Wahrscheinlichkeiten durch Summen von Produkten bedingter Wahrscheinlichkeiten aus den CPTs ersetzt werde:

$$P(C = hoch \cap A = \text{int}eressant)$$
$$= \sum_B P(C = hoch \mid A = \text{int}eressant \cap B) \cdot P(A = \text{int}eressant) \cdot P(B)$$
$$= 0{,}8 \cdot 0{,}5 \cdot 0{,}5 + 0{,}6 \cdot 0{,}5 \cdot 0{,}5 = \underline{\underline{0{,}350}}$$

Somit ergibt sich für die gesuchte Wahrscheinlichkeit:

$$P(A = \text{int}eressant \mid C = hoch) = \frac{0{,}350}{0{,}525} = \underline{\underline{0{,}670}}$$

Dies scheint plausibel, da sich die Wahrscheinlichkeit dafür, dass der Klausurstoff interessant ist, mit dem Wissen, dass die Motivation zu Lernen hoch ist, erhöht. Denn die a-priori Wahrscheinlichkeit dafür, dass der Klausurstoff interessant ist, beträgt laut CPT 0,50.

2.3 Einflussdiagramme

Einflussdiagramme stellen einen entscheidungstheoretischen Ansatz dar und sind mit Bayes' schen Netzen eng verwandt. Dabei lassen sich Entscheidungsdiagramme als Erweiterung Bayes' scher Netze auffassen.

Definition (Einflussdiagramm) [Je01]: Ein Einflussdiagramm besteht aus einem gerichteten azyklischen Graphen über der Vereinigung von drei verschiedenen Knotenmengen: einer Menge von Zufallsknoten $X = \{X_1,\ldots,X_n\}$, Entscheidungsknoten $D = \{D_1,\ldots,D_m\}$ und Bewertungsknoten $U = \{U_1,\ldots,U_k\}$. Folgende Eigenschaften gelten:

- es existiert ein gerichteter Pfad, der alle Entscheidungsknoten D enthält,
- die Bewertungsknoten U haben keine Kinder.

Zusätzlich gilt:

- Die Zufalls- und Entscheidungsknoten X und D besitzen je eine Menge von Zuständen, die sich gegenseitig ausschließen,
- jedem Zufalls- und Entscheidungsknoten ist eine CPT zugeordnet,
- die Bewertungsknoten U haben keine Zustände,
- jedem Bewertungsknoten U_i ist eine Bewertungsfunktion f_{U_i} über $pa(U_i)$ zugeordnet.

Die Zufallsknoten entsprechen den Knoten eines Bayes' schen Netzes. Entscheidungsknoten stellen Punkte eines Entscheidungsprozesse dar, die die Wahl alternativer Möglichkeiten, abhängig von zuvor angestellten Beobachtungen, ermöglichen. Bewertungsknoten dienen der Bewertung alternativer Optionen.

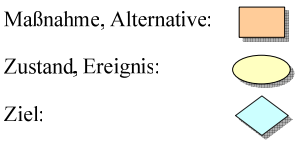


Abbildung 10: Komponenten eines Einflussdiagramms

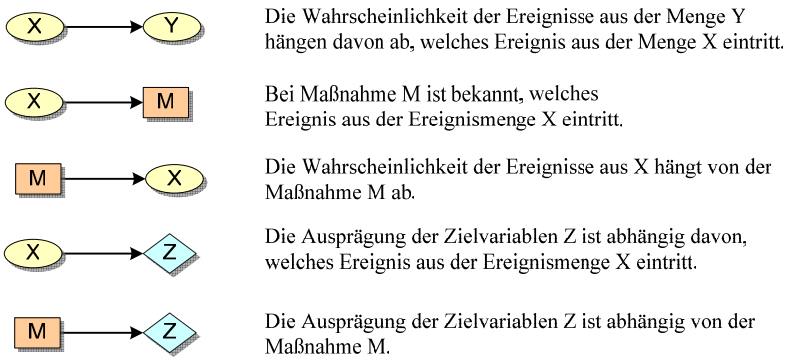


Abbildung 11: Beispielhafte Darstellungen im Einflussdiagramm

Zur Verdeutlichung sei an dieser Stelle folgendes kleines Beispiel gegeben, welches in Abbildung 12 dargestellt ist. Das Ziel besteht hierbei in der Maximierung des Gewinns. Der Gewinn jedoch ist direkt abhängig von der Nachfrage. Steigt beispielsweise die Nachfrage, so lässt sich über die Maßnahme „Kapazität" eine Erhöhung der Produktionskapazität erzielen, so dass die gestiegene Nachfrage gedeckt werden kann. Dies führt dazu, dass ein Anstieg des Gewinns zu verzeichnen ist (weitere Abhängigkeiten, z.B. in Bezug auf absatzpolitische Maßnahmen, seien hier nicht betrachtet).

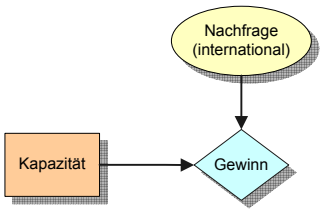


Abbildung 12: Beispiel eines Einflussdiagramms

2.4 Dynamische Bayes' sche Netze

Im Lumière-Projekt werden in Bezug auf die Inferenz auch solche Daten verwendet, die zu einem früheren Zeitpunkt erhoben worden sind. Aus diesem Grund muss ebenso ein temporärer Aspekt in die Modellierung eingebracht werden. Dies wird durch sog. dynamische Bayes' sche Netze ermöglicht. Dadurch wird es möglich, Eigenschaften, welche sich im Verlauf der Interaktion zwischen Nutzer und System verändern, zu berücksichtigen.

Ein dynamisches Bayes' sches Netz basiert auf sog. Zeitscheiben. Eine Zeitscheibe stellt dabei ein Bayes' sches Netz dar, das um Informationen bzgl. des Übergangs zur nächsten Zeitscheibe erweitert wird. Die einzelnen Zeitscheiben werden aneinandergereiht, wobei eine der Zeitscheiben den aktuellen Zeitpunkt widerspiegelt, die restlichen vorhergehende bzw. zukünftige Zeitpunkte [Wi02].

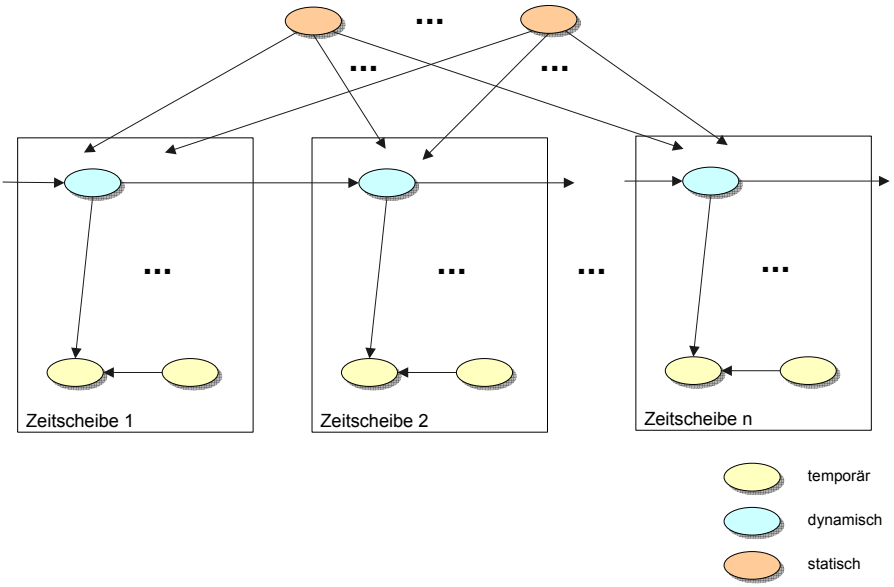


Abbildung 13: Dynamisches Bayes'sches Netz [Wi02]

Wie in Abbildung 13 zu sehen ist, werden drei verschiedene Arten von Knoten unterschieden:

- Temporäre Knoten: Dabei handelt es sich um Knoten, die lediglich in einer Zeitscheibe Geltung haben. Sie können Knoten in anderen Zeitscheiben nur indirekt, über andere Knoten, beeinflussen.
- Dynamische Knoten: Diese Knoten stellen Variablen dar, deren Zustand sich im Verlauf der Zeit ändern kann. I.d.R. liegt in jeder Zeitscheibe eine Instanz des dynamischen Knotens vor, wobei der jeweilige Zustand des Knotens der Zeitscheibe entspricht.

- Statische Knoten: Diese Knoten repräsentieren Variablen, deren Zustand sich im Verlauf der Zeit nicht verändert. Aus diesem Grund tritt ein statischer Knoten innerhalb eines dynamischen Bayes' schen Netzes nur ein Mal auf, wobei er außerhalb der Zeitscheiben modelliert wird.

Die zeitlichen Veränderungen in einem dynamischen Bayes' schen Netz werden mit Übergangs-CPTs modelliert. Dabei sind diese Übergangs-CPTs denjenigen Kanten zugeordnet, welche zwischen zwei dynamischen Knoten benachbarter Zeitscheiben verlaufen.

3 Entwurf und Konstruktion Bayes'scher Nutzermodelle

Nach dem Überblick über die theoretischen Grundlagen wollen wir nun näher auf das Lumière-Projekt eingehen und die Vorgehensweise bei Entwurf und Konstruktion Bayes'scher Nutzermodelle betrachten.

3.1 Entwurf Bayes' scher Nutzermodelle

Um von einer reinen Spezifikation des Problems der Bayes'schen Nutzermodellierung zu spezifischen Domänen und Prototypen zu gelangen, wird eine detaillierte Betrachtung der Unterschiede und Beziehungen bestimmter Software-Programme und –Konfigurationen benötigt.

Hierzu kam es im Rahmen des Lumière-Projekts zur Zusammenarbeit mit Psychologen des Microsoft usability laboratory's, um die unterschiedlichen Bedürfnisse und Verhaltensweisen menschlicher Probanden im Umgang mit Software-Anwendungen und dabei auftretender Probleme genauer zu untersuchen.

Die Ziele dieser Studien waren:

- Eine Beurteilung der Fähigkeit von Experten eine korrekte Einschätzung von Benutzerzielen durch reine Beobachtung abzugeben.

- Den Benutzern sollten, durch reine Beobachtung ihrer Aktionen, Hilfestellungen angeboten werden, um somit auftretende Probleme im Umgang mit Software-Anwendungen schneller zu bewältigen und zum gewünschten Ergebnis zu gelangen.

- Schwierigkeiten, die die Experten während der Studie bei der Ermittlung des korrekten Benutzerbedürfnisses hatten, wurden als ein Indiz für Probleme bei der Automatisierung der Hilfestellung angesehen ⇨ Aufdecken von Automatisierungs-Schwierigkeiten in bestimmten Anwendungsbereichen.

- Ermittlung spezifischer Anhaltspunkte, an denen zum einen erkennbar ist, dass ein Benutzer Hilfe benötigt und zum anderen, um welche Art von Hilfestellung es sich dabei thematisch handeln müsste.

Die Studien basierten dabei auf dem „Wizard of Oz[32] - Design". Dabei kommen Experten (Human wizards) zum Einsatz, die sich den Probanden gegenüber als ein automatisiertes Hilfe-System ausgeben. Dazu ist vor Allem eine räumliche und akustische Trennung der Experten von den Probanden von Nöten. Die Experten können schließlich über Monitore die Benutzeraktionen, bestehend aus Maus-Klicks, Tastatur-Eingaben, Pausieren (*Pause after activity*, siehe Evidenz-Klassen in Punkt 3.1) usw., ebenso wie die Benutzerbildschirme einsehen und somit abwägen, wann ein Benutzer Hilfe benötigt.

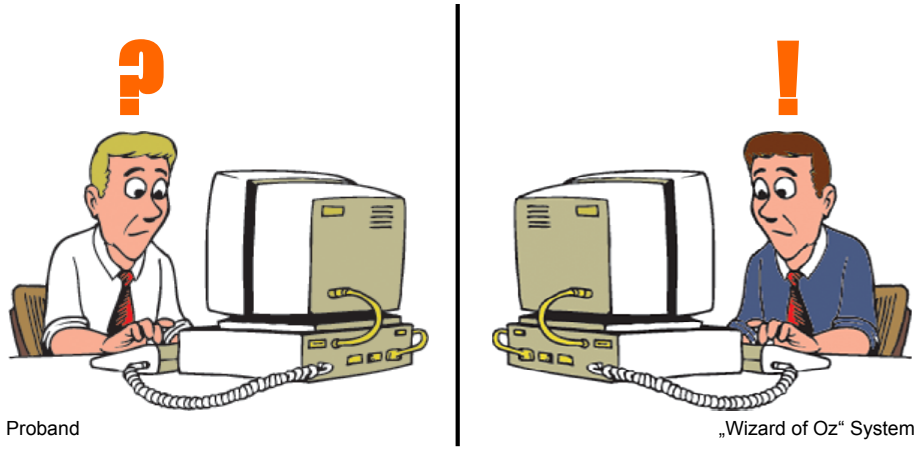


Abbildung 14: Wizard of Oz - Design

Wizard of Oz - Design:

- Human wizards (Experten), die sich den Probanden gegenüber als ein automatisches Hilfe-System ausgeben.
- Räumliche und akustische Trennung der Experten von den Probanden.
- Experten können die Aktivitäten der Probanden per Monitor einsehen.

[32] **Zauberer von Oz** - Nähere Informationen zur Namensgebung konnten leider nicht gefunden werden; das Design findet aber unter diesem Namen häufig Anwendung, da uns bei unserer Recherche einige Studien dieser Art begegnet sind.

Im Rahmen des Lumière-Projekts wurden nun Probanden mit unterschiedlichen Kompetenzgraden in Bezug auf die Verwendung von Microsofts Tabellenkalkulation Excel eingesetzt. Ihnen wurde dabei ein Set von Tabellenkalkulations-Aufgaben mit unterschiedlichen Schwierigkeitsgraden gegeben und sie wurden darüber informiert, dass ein experimentelles Hilfe-System ihre Aktivitäten aufzeichnen und gelegentlich Hilfestellungen für eine bessere Bewältigung der Aufgaben machen würde.

Die Hilfestellungen würden anschließend auf einem angrenzenden Monitor erscheinen und wären nicht verbindlich, sprich die Probanden konnten selbst entscheiden, ob sie die gegebenen Vorschläge verwenden wollten oder nicht.

Ein weiterer wichtiger Punkt war, dass die Experten die Tabellenkalkulations-Aufgaben nicht kannten und nur anhand der Benutzer-Aktionen erkennen mussten, welches Ziel dieser verfolgt.

Die ganze Studie, d.h. sowohl die Probanden als auch die Experten, wurde zur besseren Analyse auf Video aufgezeichnet.

Diese Studie führte nun zu mehreren Einblicken:

- Die Experten waren in der Lage, die Benutzer-Ziele und –Bedürfnisse durch reine Beobachtung zu identifizieren.

- Jedoch war das korrekte Erkennen dieser Ziele und Bedürfnisse eine schwierige Aufgabe, welche oftmals erst durch das Ausprobieren mehrerer Hilfestellungen bewältigt werden konnte.

- Eine weitere Erkenntnis war, dass die Probanden, obwohl sie darauf hingewiesen wurden, dass es sich um ein experimentelles Hilfe-System handelt, die gegebenen Hilfestellungen zwar für gewöhnlich sorgfältig überprüften, diese aber anschließend in die Tat umgesetzt haben; auch wenn es sich um, auf die Zielsetzung bezogen, falsche Hilfestellungen handelte und sie dadurch von ihrer eigentlichen Aufgabenstellung abgebracht wurden.

- Dies hatte auch zur Folge, dass eine Fehl-Interpretation der Benutzer-Aktionen seitens der Experten nicht sofort erkannt wurde und durch dieses Feedback, d.h. die Annahme auch falscher Hilfestellungen, der Fokus auf vermeintliche Probleme gesetzt wurde und zu einem hohen Verlust an Effizienz führte.

- Im Zeitverlauf der Studie fand bei den Experten ein Lernprozess statt und sie verhielten sich konservativer bei der Vergabe von Hilfestellungen. Diese wurden dann in Form von bedingten Aussagen formuliert, z.B. „Wenn Sie gerade versuchen dies zu tun, dann könnte Ihnen evtl. folgender Lösungsvorschlag … helfen“, und sie wurden in kleinere, leichter verständlichere Schritte zerlegt.

3.1 Konstruktion Bayes'scher Nutzermodelle

Die Studien an menschlichen Probanden haben schließlich dabei geholfen eine Kategorisierung der Evidenz[33] zu erstellen. Diese beobachtbaren Anhaltspunkte schienen dafür geeignet zu sein, Schlussfolgerungen über Probleme der Benutzer anzustellen und Aussagen über die vom Benutzer benötigte Hilfe treffen zu können. Diese Evidenz-Klassen beinhalten:

- *Suche*: Darunter versteht man beobachtbare, sich wiederholende Verhaltensmuster der Nutzer, die sich in der Form äußern, dass ein Nutzer auf der Suche nach etwas bestimmtem ist. Er wird z.B. dabei beobachtet, wie er die verschiedensten Menüs auf der Suche nach einer bestimmten Funktionalität erforscht, wie er auf der Suche nach einer bestimmten Passage durch einen Text scrollt oder mit der Maus auf viele Nicht-aktive Regionen der Anwendung klickt.

- *Fokus*: Fokus meint die Konzentration auf einen bestimmten Bereich. Man kann z.B. eine Auswahl und/oder das Verweilen auf grafischen Objekten, das Verweilen in Textpassagen oder in bestimmten Untertexten, nachdem durch ein Dokument gescrollt wurde, beobachten.

- *Introspektion (Selbstbeobachtung)*: Hierbei wird eine plötzliche Pause nach einer gewissen Zeitspanne der Benutzer-Aktivität oder eine signifikante Reduzierung der Interaktionsgeschwindigkeit des Benutzers beobachtet ⇨ *Pause after activity*. Dies kann zwar oftmals auf benötigte Hilfe zurückgeführt werden, da der Benutzer sich in solch einem Moment nicht schlüssig ist, wie er mit seiner Aufgabe fortfahren soll, aber es sind natürlich auch andere Gründe denkbar, die es zu differenzieren gilt.

- *Unerwünschte Effekte*: Darunter versteht man Versuche nach einer Benutzer-Aktion zu einem früheren Status zurückzukehren. Diese Beobachtungen beinhalten z.B. das Rückgängigmachen der Auswirkungen einer letzten Aktion durch den Benutzer – denkbar wären hier Tipp-Fehler, die der Benutzer beseitigt, wenn er sie erkennt. Ein Anwendungsgebiet dieser Klasse wäre momentan z.B. die automatische Tipp-Fehler-Erkennung in Microsoft Word, die typische Fehler automatisch korrigiert. Des Weiteren fällt in diese Klasse z.B. das Ausführen eines Undo-Kommandos oder das sofortige Schließen einer Dialogbox, nachdem sie sich geöffnet hat, ohne den Aufruf einer in diesem Dialog enthaltenen Operation.

---

[33] **Evi'denz**, *die; -, keine Mehrzahl* **1.**Augenscheinlichkeit, völlige Klarheit **2.** österreichisch in der Wendung 'in E. halten', auf dem Laufenden halten; im Auge behalten, vormerken

- *Ineffiziente Befehlssequenzen*: Hierbei werden Operationsfolgen und die gewünschten Ziele der Nutzer erkannt, die einfacher oder effizienter durch eine alternative Befehlssequenz oder durch Shortcuts durchgeführt werden können. Denkbar wären hier automatische Vorschläge von Shortcuts für z.B. das Drucken eines Dokumentes anstelle des umständlicheren Weges über die Navigation durch ein Menü der Anwendung.

- *Domänen-spezifisch syntaktisch und semantischer Inhalt*: Meint die Betrachtung der speziellen Unterschiede im Inhalt oder in der Struktur verschiedener Dokumente und wie der Benutzer mit diesen Eigenschaften umgeht. Diese schließen die Domänen-spezifischen Eigenschaften ein, die mit der Aufgabe verbunden sind.

Diese Klassen bieten nun, wenn sie weiterhin in spezifische Arten von Datenstrukturen und angezeigten Objekten unterteilt werden, einen reichhaltigen Satz von Beobachtungen mittels dieser mit hoher Wahrscheinlichkeit auf die beabsichtigten Ziele eines Benutzers geschlossen werden kann.

3.2 Struktur Bayes'scher Nutzermodelle

Auf Grundlage der Nutzerstudien wurden nun Bayes'sche Modelle, mit der Fähigkeit Nutzerbedürfnisse zu diagnostizieren, aufgestellt und bewertet.

Das Hauptaugenmerk lag dabei auf der Qualität der Schlussfolgerungen, die durch die Beobachtungen des Hintergrundes eines Nutzers (bestehend aus z.B. den Skills usw.), fortwährender und langfristiger Benutzertätigkeiten, Datenstrukturen, sowie Wörtern in einer expliziten Benutzeranfrage, erzielt werden.

Die Modellierung effizienter Nutzermodelle ist dabei abhängig von der Definition geeigneter Variablen und deren Zustände. Die bewerteten (oder erlernten) bedingten Wahrscheinlichkeiten und das Monitoring der Nutzer sind abhängig von klaren und passenden Definitionen der Zustände der Variablen. Z.B. muss man bei diskreten Bayes'schen Netzen genau die Dauer der Inaktivität bestimmen, so dass man von einer *Pause after activity* sprechen kann, wie bereits in der Evidenz-Klasse Introspektion erwähnt.

In Abbildung 15 ist ein kleines Bayes'sches Netz dargestellt, das den Zusammenhang zwischen einer *Pause after activity* und der Wahrscheinlichkeit, dass ein Nutzer Hilfe benötigt, darstellt.

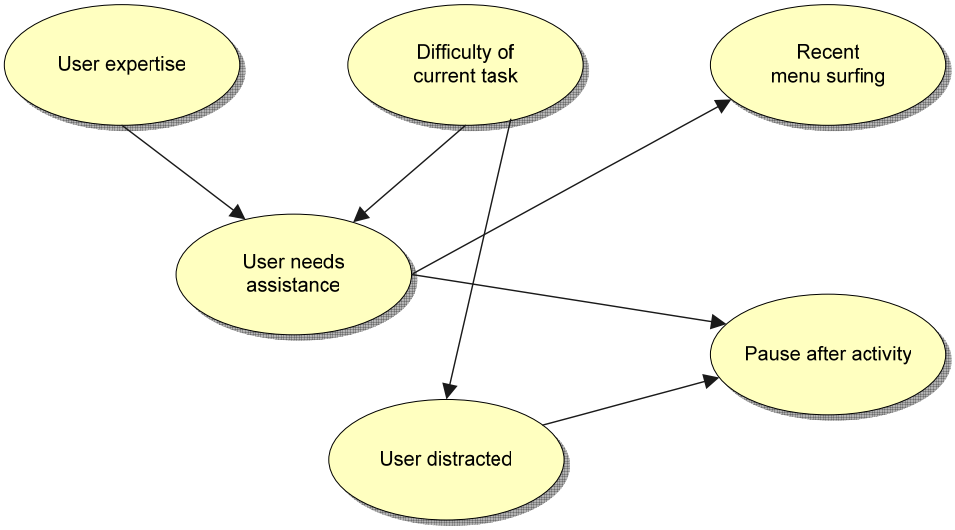


Abbildung 15: Beispiel eines Bayes´schen Nutzermodells [Ho98]

Gemäß diesem Modell wird die Wahrscheinlichkeit, den Zustand *Pause after activity* oder ein Scrollen durch verschiedene Menüs (⇨ *Recent menu surfing*) zu beobachten, erhöht, sobald ein Nutzer auf ein Problem stößt und Hilfe benötigt. Weiterhin wird der Einfluss der Sachkenntnis eines Nutzers (⇨ *User expertise*), sowie der Schwierigkeitsgrad einer Aufgabe (⇨ *Difficulty of current task*) auf die Wahrscheinlichkeit, dass ein Nutzer Hilfe benötigt betrachtet.
Eine weitere Ursache für den Zustand *Pause after activity* ist die Tatsache, dass ein Nutzer von Gegebenheiten abgelenkt werden kann, die nicht Gegenstand seiner momentanen Aufgabe sind (⇨ *User distracted*). Wie leicht sich ein Nutzer nun von seiner Aufgabe ablenken lässt, ist in Abbildung 15 durch die bedingte Wahrscheinlichkeit *P(User distracted | Difficulty of current task)* dargestellt, welche noch einen Bezug zum Schwierigkeitsgrad der Aufgabe darstellt.

Hierzu soll noch eine kurze Erläuterung des abgebildeten Bayes'schen Nutzermodells zu einem besseren Verständnis führen. Da auf die CPTs (Conditional probability tables) in der Quelle [Ho98] nicht eingegangen wurde und mir keine Werte aus dem Projekt bekannt sind, verweise ich für eine Berechnung von CPTs an dieser Stelle auf das Beispiel in 2.2.3:

- ***User expertise*** (hoch | niedrig) und ***Difficulty of current task*** (hoch | niedrig) bedingen die Wahrscheinlichkeit von ***User needs assistance*** (hoch | niedrig).

$$P(User\ needs\ assistance | User\ expertise \cap Difficulty\ of\ current\ task)$$

- ***User needs assistance*** (hoch | niedrig) bedingt des Weiteren die Wahrscheinlichkeit von ***Recent menu surfing*** (hoch | niedrig).

  $$P(Recent\ menu\ surfing \mid User\ needs\ assistance)$$

- Die Wahrscheinlichkeit von ***Pause after activity*** (hoch | niedrig) wird von ***User needs assistance*** (hoch | niedrig) sowie von ***User distracted*** (hoch | niedrig) bedingt.

  $$P(Pause\ after\ activity \mid User\ needs\ assistance \cap User\ distracted)$$

- ***Difficulty of current task*** (hoch | niedrig) bedingt wiederum die Wahrscheinlichkeit von ***User distracted*** (hoch | niedrig).

  $$P(User\ distracted \mid Difficulty\ of\ current\ task)$$

3.3 Beispiele Bayes'scher Nutzermodelle im Rahmen des Lumière-Projekts

Im Rahmen des Lumière-Projekts kam es schließlich, wiederum unter Zusammenarbeit mit Experten, zur Konstruktion, zum Testen und Bewerten mehrerer Bayes'scher Netze für unterschiedliche Anwendungen und Aufgabenstellungen.

- Einige dieser Modelle wurden als *application covering*, d.h. als permanente, die komplette Software-Anwendung umfassende Modelle,

- andere hingegen wurden als einfachere, kontext-sensitive Agenten, die nur bei Erkennung bestimmter Verhaltensmuster in Bezug auf die Benutzer-Aktionen aufgerufen wurden, entwickelt.

3.3.1 Agent im Betriebssystem Microsoft Windows 95

Eines dieser Bayes'schen Nutzermodelle war ein Agent in der Windows 95-Shell, welcher nur dann in Erscheinung trat, wenn bei einem Mouse-over auf den Start-Button und bei der Navigation durch das Start-Menü eine gewisse Zeitspanne überschritten wurde. Es wurde dann versucht dem Nutzer die Funktionalität der einzelnen Optionen zu erläutern.

Leider wurde für dieses Beispiel kein Bayes'sches Netz vorgestellt, so dass es hier bei einer reinen Beschreibung der Anwendung bleibt.

3.3.2 Ausschnitt des Bayes'schen Nutzermodells für Microsoft Excel

Ein Beispiel für ein *application covering* Modell stellt der folgende Ausschnitt aus einer frühen Version des Bayes'schen Netzes in Abbildung 16 dar, welches im Zuge des Lumière-Projektes für die Tabellenkalkulation Excel erstellt wurde. Hier wird noch einmal deutlich der Einfluss des *User background*, sprich seine Skills, die Aufgabenstellung usw. auf das *Primary goal* dargestellt.

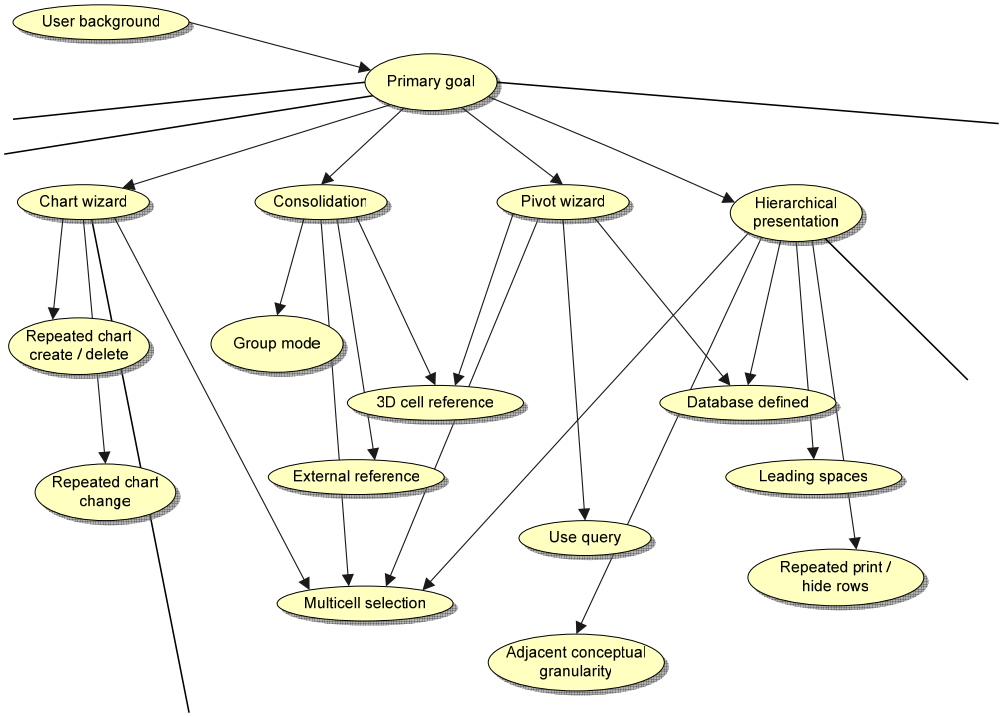


Abbildung 16: Ausschnitt aus dem Bayes´schen Nutzermodell für Microsoft Excel [Ho98]

4 Schlussfolgerungen über Nutzeraktionen im Zeitverlauf

Zwischen den Nutzerzielen und dem Nutzerverhalten zu unterschiedlichen Zeitpunkten der Beobachtung existieren temporale Abhängigkeiten. Um diese Tatsache bei der Berechnung der Inferenz zu berücksichtigen werden dynamische Bayes'sche Netze (s. *2.4 Dynamische Bayes'sche Netze*) zur Modellierung von temporalen Veränderungen der Variablen und ihrer Zustände bei der Interaktion zwischen Nutzer und System verwendet.

Ein dynamisches Bayes'sches Netz basiert, wie schon erwähnt, auf Zeitscheiben, welche dabei wiederum ein Bayes'sches Netz darstellen. Die einzelnen Zeitscheiben sind mit den angrenzenden über statische Knoten in Beziehung gesetzt, welche im Verlauf der Zeit unveränderlich sind und nur einmal in einem dynamischen Bayes'schen Netz vorkommen. Über den Zeitverlauf hinweg entsteht ein hier beispielhaft dargestelltes Nutzer-Modell.

In Abbildung 17 wird anhand eines Markov-Modells das Problem der temporalen Veränderung in der Bayes'schen Nutzer-Modellierung visualisiert. Es werden die Abhängigkeiten zwischen den Variablen der angrenzenden Zeitabschnitte deutlich und das Ziel zum aktuellen Zeitpunkt ($Ziel_{t0}$) mit den vorhergehenden ($Ziel_{t-i}$) in Beziehung gesetzt. Ebenso eine Veränderung der beobachtbaren Ereignisse (E_i) wird berücksichtigt.

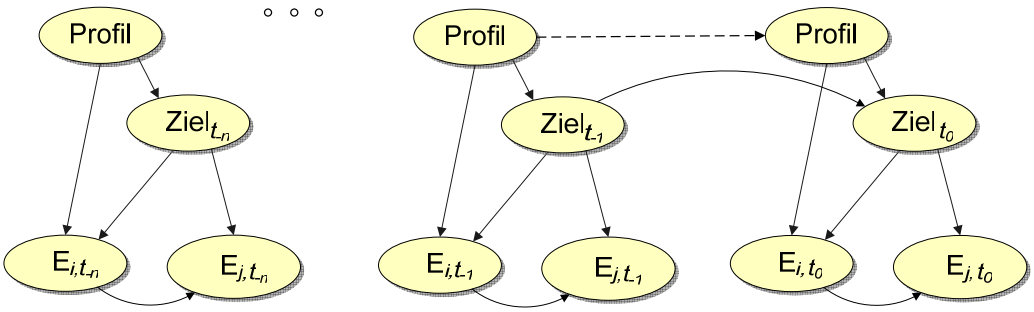


Abbildung 17: Auszug aus einem Markov-Modell [Ho98]

Hier eine kurze Erläuterung des Markov-Modells:

- Das Nutzerprofil verändert sich entweder nur langsam im Zeitverlauf, sprich die Skills der Nutzer nehmen durch den wiederholten Umgang mit einer Anwendung langsam zu, oder es bleibt gleich – hier dargestellt durch einen gestrichelten Pfeil.
  Somit ist die Variable *Profil* den dynamischen Knoten, wie in den Grundlagen unter 2.4 Dynamische Bayes'sche Netze näher erläutert, zuzuordnen. Ihr Zustand kann sich im Verlauf der Zeit verändern. Jede Zeitscheibe beinhaltet eine Instanz dieses dynamischen Knotens mit dem der Zeitscheibe entsprechenden Zustand.

- Aus dieser Weiterentwicklung des Nutzerprofils resultierend, können sich auch die Nutzerziele im Zeitverlauf ($Ziel_{t-1}$ ⇨ $Ziel_{t0}$) verändern. Ebenso die Variable *Ziel* ist der Kategorie der dynamischen Knoten zuzuordnen.

- Auch die Variablen der beobachtbaren Ereignisse *E*, die durch die Nutzerziele bedingt werden und aus den Nutzeraktionen resultieren, können sich über den Zeitverlauf hinweg unterscheiden.
  Bei diesen Variablen handelt es sich allerdings um temporäre Knoten, die lediglich in einer Zeitscheibe zum Tragen kommen und in einer anderen Zeitscheibe nicht zwingend existieren müssen. Diese temporären Knoten können Knoten in anderen Zeitscheiben nur indirekt, über andere Knoten beeinflussen.

- Letztlich variieren durch diese Veränderungen auch die in den Bayes'schen Netzen enthaltenen bedingten Wahrscheinlichkeiten $P(E_{i,t} \mid Goal_t)$ für das Eintreten der Ereignisse.

- Gemäß den bereits eingeführten dynamischen Bayes´schen Netzen kann man auch hier die Kanten erkennen, welche die benachbarten Zeitscheiben verbinden und in Beziehung setzen.

- Allerdings fehlen in diesem Markov-Modell die statischen Knoten, wie sie in den Grundlagen als Kennzeichen für dynamische Bayes'sche Netze eingeführt wurden, gänzlich. So könnte man eigentlich auch von einer Aneinanderreihung von einfachen Bayes'schen Netzen sprechen.

5. Zusammenführung von Systemereignissen und Nutzeraktionen

Ein kritisches Problem bei der Entwicklung von wahrscheinlichkeits- und entscheidungstheoretischen Verbesserungen für User-Interface-Applikationen ist es, eine Verbindung zwischen den resultierenden Systemereignissen und den eigentlichen Nutzeraktionen herzustellen.

Ein Grund dafür ist, dass Systeme wie auch Anwendungen meist nicht mit genügend Rücksicht auf die Nutzer-Modellierung entwickelt werden.

Deshalb kam es in Zusammenarbeit mit dem Excel Development Team zur Entwicklung spezieller Excel Versionen, die um Instrumentarien zur Informationsgewinnung erweitert wurden. Diese Instrumentarien erzeugten nun Informationen über:

- Maus-Klicks und Tastatur-Eingaben der jeweiligen Benutzer im Umgang mit der Tabellenkalkulationssoftware

- Den Zustand der Datenstrukturen in Excel Dateien

- Das Scrollen der Nutzer durch Menüs und Texte

- Das Öffnen und Schließen von Dialogboxen

- Die Auswahl von Grafik-Objekten, Charts, Zellen, Spalten, Zeilen und Textpassagen

5.1 Das Lumière Events System

Basierend auf diesen gewonnenen Informationen kam es zur Entwicklung des **Lumière Events Systems**, um eine Verbindung von atomaren Nutzeraktionen auf unterster Ebene (atomare Ereignisse) zu deren Bedeutung in Form von Systemereignissen auf höherer Ebene (modellierte Ereignisse) herzustellen. Das Lumière Events System zeichnet nun atomare Aktionen auf und versieht sie mit einem Zeitstempel. Anschließend werden sie in modellierte, Nutzeraktionen darstellende Ereignisse transformiert. Für diese Transformation werden detaillierte Analysen der atomaren Ereignisströme, sowie spezielle temporale Funktionen benötigt, die die atomaren Ereignisfolgen in Beobachtungen auf höherer Ebene (modellierte Ereignisse) abbilden können. Diese Beobachtungen (modellierten Ereignisse) bestehen z.B. aus Maus-Klicks, Scrollen durch Menüs usw.

5.2 Die Lumière Events Language

Um atomare Ereignisse nun effizienter, flexibler und direkt in modellierte Ereignisse zu transformieren, wurde im Zuge der Entwicklung des Lumière Events Systems eine spezielle Sprache generiert - die **Lumière Events Language**.

Diese Sprache kann zeitliche Muster erkennen und so atomare Ereignisse direkt in modellierte Ereignisse und atomare Ereignisströme direkt in Boolesche Werte transformieren. Ebenso erlaubt sie Systementwicklern neue modellierte Ereignisse aus bereits vorhandenen und atomaren Ereignissen zu erzeugen.

Ereignisse werden in Beobachtungen transformiert und mit time-stamps versehen. Anschließend werden diese modellierten Ereignisse als Input für das Bayes'sche Modell verwendet.

…. dar, aus dem anschließend eine Wahrscheinlichkeitsverteilung über die Nutzerbedürfnisse inferiert wird. Falls eine explizite Anfrage vorliegt, gehen sowohl die Ereignisse, als auch die Anfragebegriffe, kombiniert durch eine gewichtete Multiplikation, in das Bayes' sche Modell ein. Das für explizite Anfragen eingesetzte Information Retrieval-System erkennt ca. 600 Anfrageterme. Zusätzlich berechnet das System die Wahrscheinlichkeit, dass ein Nutzer zum aktuellen Zeitpunkt Hilfe benötigt. Diese Wahrscheinlichkeit wird benötigt, um die Anzeige des Assistenten zu kontrollieren. Dabei wurden verschiedene Kontroll-Schemata für das Lumière/Excel System untersucht:

Hier ein Auszug aus den Transformations-Funktionen dieser Sprache, die als Laufzeitfilter eingesetzt werden:

- ***Rate(x_i,t)***: Vorkommenshäufigkeit eines atomaren Ereignisses x_i in t Sekunden.
- ***Oneof({x_1,…,x_n},t)***: Auftreten mindestens eines Ereignisses in der Zeit t.
- ***Seq(x_1,…,x_n,t)***: Ereignisse treten in t in spezieller Reihenfolge auf.
- ***Dwell(t)***: Keine Nutzeraktion für mindestens t Sekunden.
- ***All({x_1,…,x_n},t)***: Alle Ereignisse einer festgelegten Ereignismenge erscheinen mindestens einmal in beliebiger Reihenfolge in der Zeit t.
- ***TightSeq(x_1,…,x_n,t)***: Ereignisse erscheinen in einer bestimmten Reihenfolge in der Zeit t und keine anderen Ereignisse erscheinen.

Diese und andere Operatoren können nun in der Lumière Events Language dazu verwendet werden, um Filter für Ereignisse auf höherer Ebene zu generieren; z.B. *user dwelled for at least t seconds following a scroll*, d.h. ein Nutzer hat nach der Aktion „Scrollen“ für mindesten t Sekunden keine andere Aktion ausgeführt. Eine weitere Vorgehensweise war es, Nutzeraktionen auf unterster Ebene in Ereignis-Klassen einzuordnen. Ein Beispiel hierfür ist die Zuordnung der Nutzeraktionen eine Datei über das entsprechende Icon in der Anwendung oder über einen Shortcut zu speichern, zur Klasse *Datei speichern*.

6 Das Lumière/Excel System

Der Lumière-Prototyp für die Tabellenkalkulationssoftware Excel diente insbesondere dem Zweck, das Potential der Bayes’ schen Nutzermodellierung für das Microsoft Office Paket zu demonstrieren. Horvitz und sein Team konstruierten ein Bayes’ sches Nutzermodell, welches in etwa 40 Problembereiche der Tabellenkalkulationssoftware Excel abdeckte. Hierbei wurde die „Lumière Events Language“ eingesetzt, um Beobachtungen als Funktion der Ereignisse, die durch Excel hervorgerufen wurden, zu modellieren. Weiterhin wurde das System eng mit einer „Bayes’ schen Term Spotting Methode“ gekoppelt, um über Freitext-Anfragen Schlussfolgerungen ziehen zu können.

6.1 Die Lumière/Excel System-Architektur

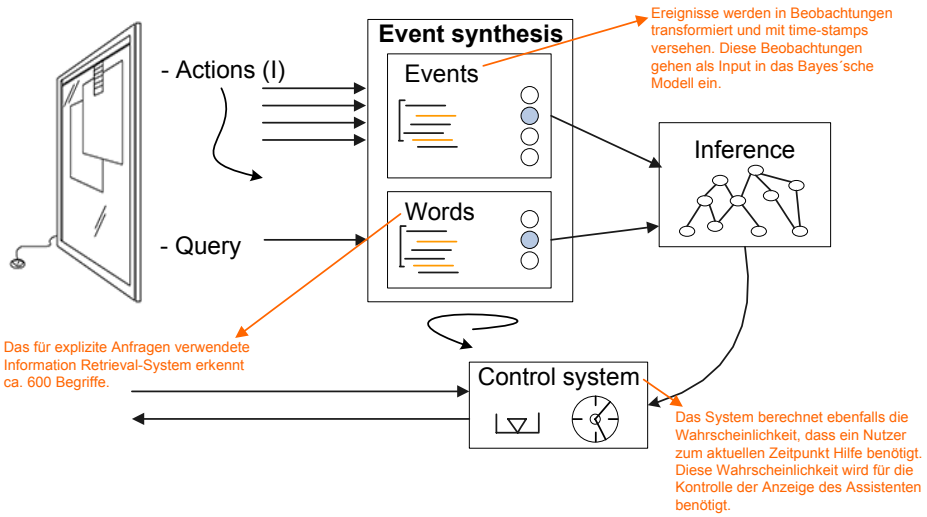


Abbildung 18: Lumière/Excel Architektur [Ho98]

Die allgemeine Lumière/Excel Architektur wird in Abbildung 18 dargestellt. Ereignisse werden in Beobachtungen transformiert und mit time-stamps versehen. Diese Beobachtungen stellen den Input für das Bayes' sche Modell dar, aus dem anschließend eine Wahrscheinlichkeitsverteilung über die Nutzerbedürfnisse inferiert wird. Falls eine explizite Anfrage vorliegt, gehen sowohl die Ereignisse, als auch die Anfragebegriffe, kombiniert durch eine gewichtete Multiplikation, in das Bayes' sche Modell ein. Das für explizite Anfragen eingesetzte Information Retrieval-System erkennt ca. 600 Anfrageterme. Zusätzlich berechnet das System die Wahrscheinlichkeit, dass ein Nutzer zum aktuellen Zeitpunkt Hilfe benötigt. Diese Wahrscheinlichkeit wird benötigt, um die Anzeige des Assistenten zu kontrollieren. Dabei wurden verschiedene Kontroll-Schemata für das Lumière/Excel System untersucht:

- Pulsed Strategy: Die Analyse erfolgt in bestimmten Zeitabständen.
- Event-driven Strategy: Die Analyse wird durch Trigger Events ausgelöst. Dabei handelt es sich um spezifische Mengen von atomaren und/oder modellierten Ereignissen.
- Augmented Pulsed Strategy: Kombination von Pulsed Strategy und Event-driven Strategy.
- Deferred Analysis: Die Analyse erfolgt zu bestimmten Zeiten, jedoch nur bei Fehlen von Benutzeraktivität.

Es gibt eine Reihe weiterer Möglichkeiten das Kontroll-System umzusetzen, beispielsweise mittels Entscheidungstheoretischer Analyse.

6.2 Lumière/Excel in Aktion

Lumière/Excel beobachtet kontinuierlich Ereignisse und aktualisiert eine Wahrscheinlichkeitsverteilung über die Bedürfnisse des Benutzers. Ausgehend von einem Strom von Ereignissen inferiert das System sowohl Bedürfnisse, als auch die Wahrscheinlichkeit, dass der Nutzer zum aktuellen Zeitpunkt Hilfe benötigt.

In Abbildung 19 sind mehrere Komponenten des Lumière-Prototyps zu sehen. Das kleine Fenster im Hintergrund, der Event Monitor, beinhaltet den Strom der atomaren Ereignisse, und die aus diesen Ereignissen gewonnenen Beobachtungen. Daran fügt sich ein Fenster mit der Wahrscheinlichkeitsverteilung über Bedürfnissen an, das User Needs Profile. Ein weiteres Fenster stellt die Wahrscheinlichkeit dafür dar, dass der Nutzer zum aktuellen Zeitpunkt eine Hilfestellung benötigt, der Assistance Monitoring Agent. Letztlich bildet die Komponente im Vordergrund das User Interface des Prototyps. Diese Komponente präsentiert die Ergebnisse und ermöglicht dem Nutzer die Interaktion mit dem System. Die Text Box auf der linken Seite beinhaltet die empfohlene Hilfe, sortiert nach der jeweiligen Wahrscheinlichkeit. Darunter fügt sich ein Eingabefeld an, das der Nutzer für die Eingabe von Freitext-Anfragen nutzen kann. Außerdem hat der Nutzer die Möglichkeit seinen Kompetenzlevel zu spezifizieren.

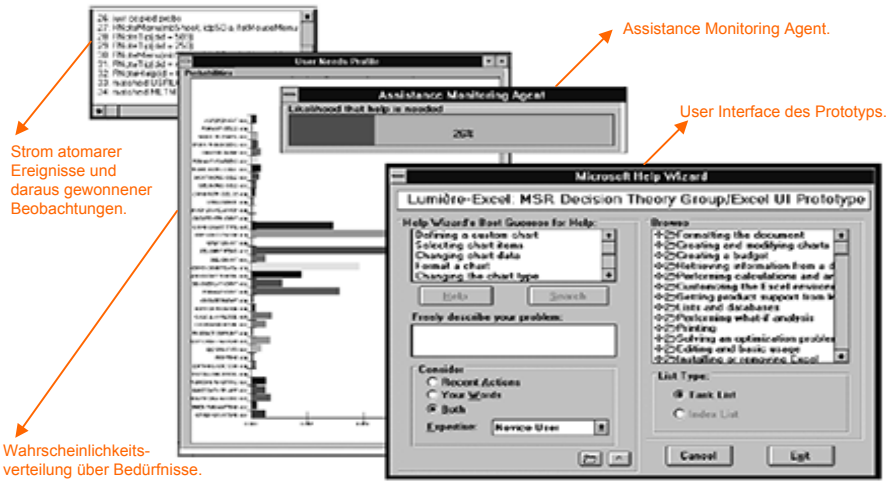


Abbildung 19: Lumière Komponenten [Ho98]

Abbildung 20 zeigt wiederum das User Interface des Lumière Prototyps. In Abbildung 20(a) sind die Analyseergebnisse, basierend auf Aktionen, dargestellt, bevor eine explizite Anfrage durchgeführt wurde. Abbildung 20(b) zeigt die aktualisierte Wahrscheinlichkeitsverteilung nach Berücksichtigung zusätzlicher Information über die Anfrageterme aus der expliziten Benutzeranfrage.

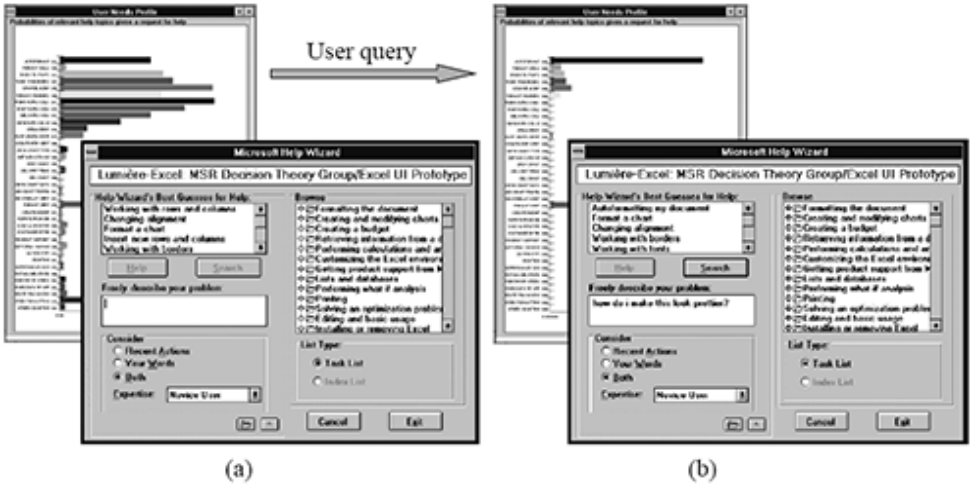


Abbildung 20: Berücksichtigung expliziter Anfragen [Ho98]

7 Microsoft Office 97 Assistent

Der Microsoft Office 97 Assistent kann als “Light-Version“ des Lumière-Projekts angesehen werden. Unterschied zum Lumière Prototyp:

- Lediglich eine geringe Anzahl von Nutzeraktionen
- Kann nur wenige Systemereignisse über den Zeitverlauf in Kombination verarbeiten.
- Trennung von Ereignissen und expliziten Nutzeranfragen in Bezug auf die Inferenz.
- Keine Nutzerprofile und kein Monitoring der Skills des Nutzers.

- In ähnlicher Form heute noch existent.

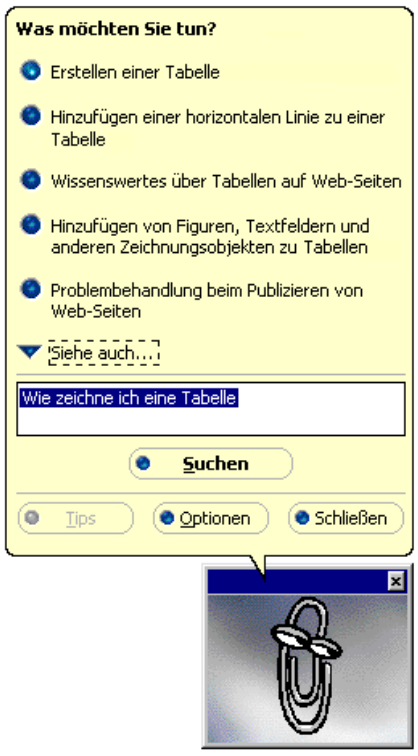


Abbildung 21: Office 97 Assistent

7.1 Treffen der Generationen

- Microsoft Office 2003 Assistent trifft seinen Großvater.
- Leider konnte er nach der Installation nicht mehr entfernt werden und gab im Hintergrund permanent scharrende Geräusche von sich.

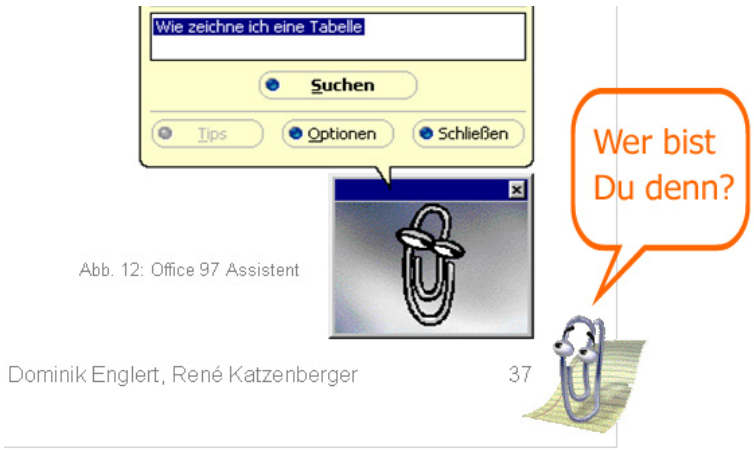


Abbildung 22: Treffen der Generationen

Literaturverzeichnis

[He98] Heckerman, D.: A tutorial on learning with Bayesian networks. In: Jordan, M.I.: Learning in Graphical Models. Cambridge, 1998.

[He03] Henrich, A.: Information Retrieval. Begleitlektüre zum Kurs Information Retrieval an der Universität Bamberg, 2003.

[Ho98] Horvitz, E.; Breese, J.; Heckerman, D.; Hovel, D.; Rommelse, K.: The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In: Proc. of the 14th Int. Conf. on Uncertainty in Artificial Intelligence. Madison, WI, 1998, Morgan Kaufmann, pp. 256-265.

[Ja02] Jameson, A.; Wahlster, W.; Bohnenberger, T.; Brandherm, B.; Großmann-Hutter, B.; Wittig, F.: READY: Lernen, Modellierung und Entscheidung für situierte Interaktion. Saarbrücken, 2002, Universität des Saarlandes.

[Je01] Jenson, F.V.: Bayesian Networks and Decision Graphs. New York, 2001.

[LS88] Lauritzen, S.L.; Spiegelhalter, D.J.: Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. Journal of the Royal Statistical Society Series, 50(2), pp.157-224, 1988.

[OW96] Ottmann, T.; Widmayer, P.: Algorithmen und Datenstrukturen. Heidelberg, 3. Auflage, 1996.

[Pe88] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, California, 1988.

[Pu91] Puppe, F.: Einführung in Expertensysteme. Berlin, 2. Auflage, 1991.

[Vo00] Vogel, F.: Beschreibende und schließende Statistik: Formeln, Definitionen, Erläuterungen, Stichwörter und Tabellen. Oldenbourg, 12. Auflage, 2000.

[Wi02] Wittig, F.: Maschinelles Lernen Bayes' scher Netze für benutzeradaptive Systeme. Saarbrücken, 2002.

[WK89] Wahlster, W.; Kobsa, A.: User models in dialog systems. In: User Models in Dialog Systems. Berlin, 1989, pp. 4-34.

Empfehlungssysteme in E-Business und E-Commerce

Tobias Fuchs
tobias.fuchs@feki.de

Abstract: Mit Empfehlungssystemen werden im E-Business Allgemein und im E-Commerce im Speziellen der Weg von der kurzfristig orientierten Neukundenakquisition zu einer langfristigen Kundenbindung hin unterstützt. Dabei werden für einzelne oder Gruppen von Kunden aus deren historischen Transaktionen Profile erstellt, die wiederum zur Vorhersage potentieller zukünftiger Wünsche und Bedürfnisse ausgeweitet werden. Ein Beispiel für den Einsatz eines Empfehlungssystems stellt Amazon.com dar.

1 Einleitung

Neben dem Vorteil, alles von zuhause aus kaufen zu können, hat E-Business und E-Commerce einen großen Nachteil: Der Kunde ist anonym und hat nur noch wenig Möglichkeiten sich von einem Verkäufer beraten zu lassen. Ein fast vergessener Effekt der Tante-Emma-Läden, dem bekannten Kunden nur das zu empfehlen, was er braucht und will, wird gerade neu entdeckt.

Ein System, das dem Kunden beim Einkauf im Webshop gezielt Hinweise gibt und ihm so hilft die Flut an Informationen zu bewältigen, ist bei großen Internethändlern wie beispielsweise Amazon heute Standard.

Empfehlungssysteme sind Systeme, deren Aufgabe vornehmlich darin besteht, Empfehlungen und Prognosen für den Benutzer abzugeben [Bo04]. Die Systeme werden dabei in mehrere Arten eingeteilt. Diese Arbeit stellt die unterschiedlichen Systeme vor und geht dabei auf die höchste Form, das Collaborative Filtering genauer ein. Folgende Aussage aus dem US-Patent 4.870.579 aus dem Jahr 1989 beschreibt, was von einem Empfehlungssystem auf Basis des Collaborative Filtering erwartet wird:

"This invention relates to a system and method of predicting reactions to items not yet sampled by a user, and more particularly to such a system and method which adjust the reaction prediction for each unsampled item for that user based on the similarity in reaction of other users to that user."[He89]

Ziel der Methode ist es, Reaktionen eines Benutzers auf Objekte vorherzusagen und sich dabei auf Reaktionen von zu diesem Benutzer ähnlichen Benutzern zu stützen [Ru00]. „Kunden, die dieses Buch gekauft haben, haben auch diese Bücher gekauft“ lautet der Hinweis auf Amazon[34], sobald man ein Buch anklickt. Dieser einfache Satz beschreibt die Grundidee des Collaborative Filtering am besten.

2 Arten von Empfehlungssystemen

Empfehlungen können genau auf eine Person bezogen sein oder sich generell, übergeordnet für alle eignen. So kann man seinen Kunden beispielsweise den Rat geben, bei zwei baugleichen Geräten das günstigere zu nehmen. Jedoch kann eine Empfehlung bezüglich eines Kleidungsstücks immer nur auf eine Person bezogen ausgesprochen werden.

Aus diesem Grund lassen sich auch verschiedene Empfehlungssysteme abgrenzen. So kann man im ersten Schritt *individualisierte* und *nicht-individualisierte* Systeme unterscheiden. Bei den individualisierten Systemen lassen sich zudem, im nächsten Schritt, *eigenschaftsbasierte Systeme* und *Empfehlersysteme* unterscheiden.

Abbildung 2.1 zeigt einen Überblick über die verschiedenen Arten von Empfehlungssystemen nach RUNTE [Ru00].

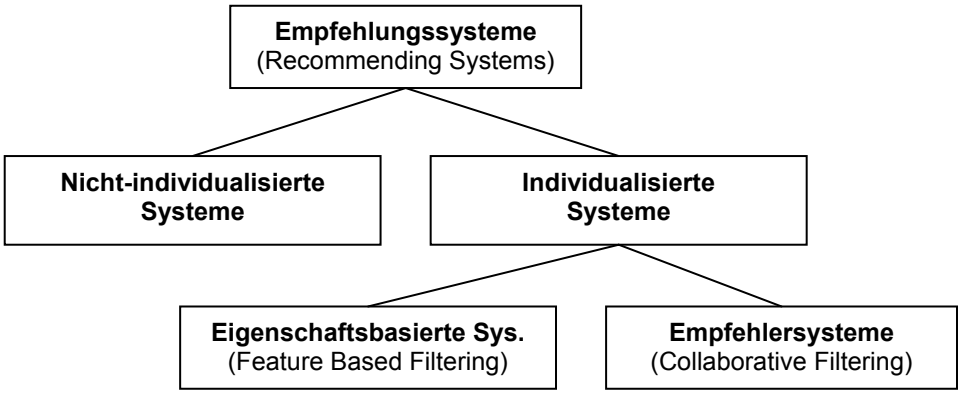


Abbildung 2.1: Arten von Empfehlungssystemen [Ru00]

In diesem Kapitel werden die verschiedenen Arten der Empfehlungssysteme vorgestellt und die Vorteile und Nachteile kurz aufgezeigt.

[34] Amazon: Online-Buchhändler. http://www.amazon.de/

Zur Erklärung der verschiedenen Systeme wird als Fallbeispiel ein Webshopsystem angenommen. In diesem Webshop können Kunden Elektronikartikel und unter anderem Telefongeräte kaufen. Die einzelnen Szenarien beziehen sich auf Kunden, die eine Empfehlung erhalten wollen, ob für sie ein Smartphone, ein Mobiltelefon (Handy) oder ein Festnetztelefon das Richtige sei.

2.1 Nicht-individualisierte Systeme

Wie der Name schon sagt, betrachten die nicht-individualisierten Systeme den Benutzer nicht. Es gibt daher für jeden Benutzer die gleiche Empfehlung ab, wenn eine Anfrage zu einem bestimmten Produkt gestellt wird. Als Ausgangsdaten zur Empfehlung werden beispielsweise die Kinofilm-Präferenzen einer breiten Masse von Benutzern gespeichert und anhand des Mittelwerts der Präferenzen wird eine Empfehlung ausgesprochen [Ru00].

Abbildung 2.2 zeigt eine typische Situation beim Einsatz von nicht-individualisierten Systemen. Eine Gruppe von Benutzern geben ihre Präferenzen zu den möglichen Geräten ab (dabei soll ein großes Symbol einer hohen Präferenz entsprechen). Der Server speichert die Präferenzen der Benutzergruppe und berechnet daraus, für Anfragen nach einem Telefongerät, die Mittelwerte und empfiehlt daher das Smartphone.

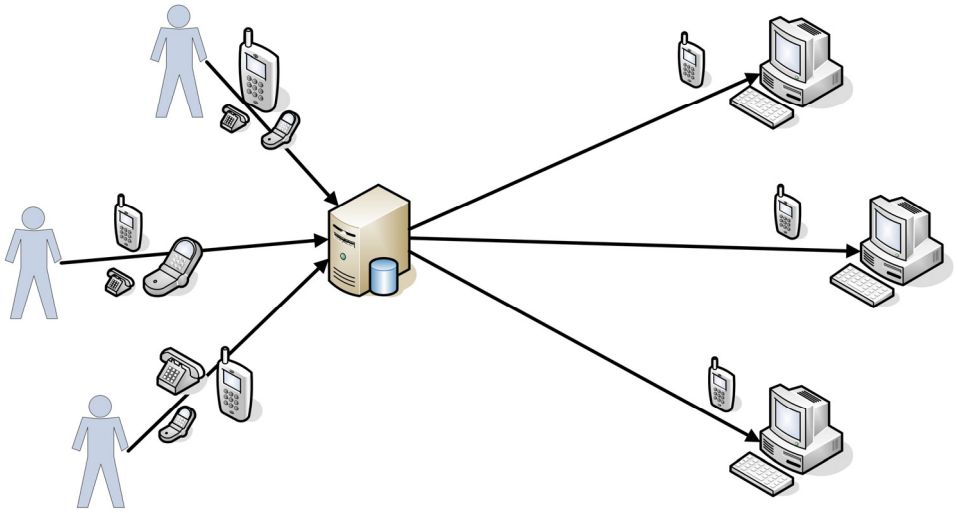


Abbildung 2.2: Nicht-individualisiertes System, eigene Darstellung

Da die nicht-individualisierten Systeme laut BOHNERT nur sehr einfache Algorithmen besitzen und gegenüber den individualisierten Systemen die Möglichkeiten der interaktiven Medien nur beschränkt nutzen [Bo04], werden diese Systeme im Folgenden vernachlässigt.

2.2 Individualisierte Systeme

Die individualisierten Systeme sind gegenüber den nicht-individualisierten Systemen besser für den Einsatz als Empfehlungssysteme geeignet. Da sie den Benutzer „kennen“, werden die Empfehlungen anhand der Präferenzen des Anfragers erstellt. Somit wird in einem Shopsystem versucht genauer auf die Kundenwünsche zu reagieren.

Die individualisierten Systeme lassen sich, wie Abbildung 2.1 zeigt, noch einmal aufteilen. Diese Aufteilung in *eigenschaftsbasierte Systeme* und *Empfehlersysteme* hängt mit dem Mechanismus, der zur Individualisierung verwendet wird zusammen [Ru00]. Die beiden Arten werden im Folgenden genauer erklärt.

2.2.1 Eigenschaftsbasierte Systeme

Die eigenschaftsbasierten Systeme, auch *Feature Based Filtering* genannt, beziehen sich auf die Eigenschaften der prinzipiell empfehlbaren Objekte, der vorhandenen Objektkategorie. Das heißt, dass hier die Eigenschaften eines Objekts untersucht werden und mit den Eigenschaften, die ein Benutzer erwartet verglichen werden. Der Mechanismus kann dann beispielsweise Produkte, die Eigenschaften aufweisen, die der Kunde ausgeschlossen hat, in der Empfehlung unberücksichtigt lassen.

Abbildung 2.3 geht von dem vorgestellten Fallbeispiel aus und zeigt ein eigenschaftsbasiertes System, an das drei unterschiedliche Kunden jeweils eine Anfrage stellen. Zudem sind die Eigenschaften der Geräte in der Datenbank gespeichert. So hat beispielsweise das Smartphone die Eigenschaften: Drahtlos, PDA[35]-Funktionen, Telefonfunktionen und hoher Preis. Die Anfragen der einzelnen Kunden, sichtbar an den Pfeilen von Kunde zu Server, enthalten nun die Kunden-Präferenzen. So ist über den roten Kunden bekannt, dass er auf Telefoneigenschaften Wert legt und sehr stark (großes Symbol) die Forderung nach PDA-Funktionen hat Die Empfehlung eines Smartphones ist daher die einzige mögliche Antwort des Systems.

[35] PDA: Abkürzung für Personal Digital Assistant; allgemeiner Begriff für kleine tragbare Computer z.B. in Form eines Organizers [Le03]

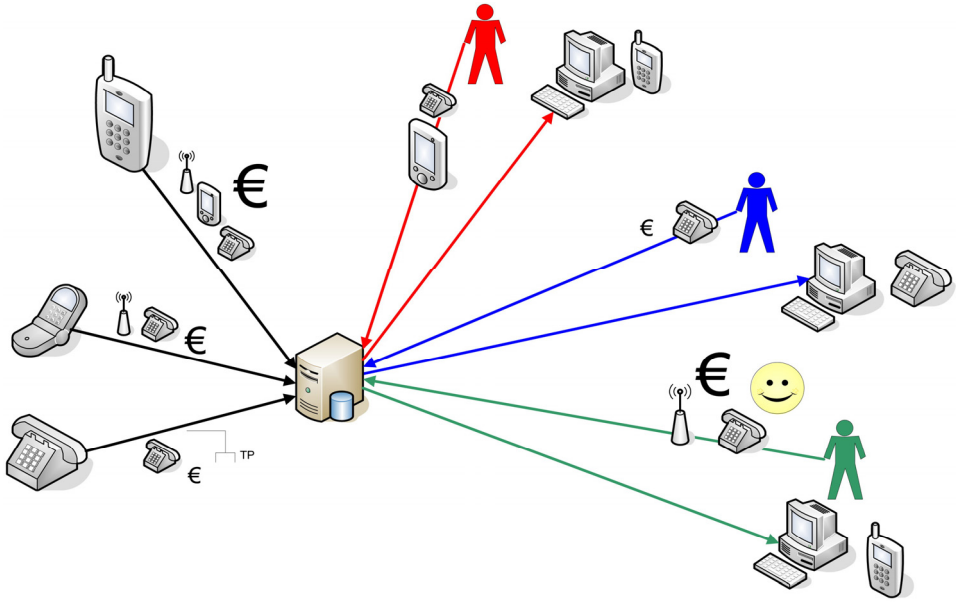


Abbildung 2.3: Eigenschaftsbasiertes System, eigene Darstellung

Die Anfrage des grünen Kunden in Abbildung 2.3 zeigt die Schwächen des eigenschaftsbasierten Ansatzes auf. Der Kunde will ein Gerät, dass neben dem Aspekt der Drahtlosigkeit und der Telefoneigenschaft auch viel Freude bereitet, zudem weiß man über den Kunden, dass er viel Geld für das Gerät ausgeben würde. Da das vorgestellte System keinerlei subjektive Eigenschaften, wie „Freude bereiten“ kennt, werden nur die drei restlichen Anforderungen an das Gerät untersucht und so erhält der Kunde die Empfehlung, das Smartphone zu kaufen.

Ein besserer Ansatz wäre es, auch subjektive Eigenschaften in die Berechnung mit einfließen zu lassen. Um diese Eigenschaften zu ermitteln ist es nötig anhand einer großen Menge von Benutzerdaten und subjektiven Eindrücken die Präferenzen zu ermitteln. Dieser Ansatz wird im nächsten Abschnitt genauer erklärt.

2.2.2 Empfehlersysteme

Empfehlersysteme oder auch *Collaborative Filtering* gehen davon aus, dass sich aus einer großen Anzahl von Benutzern, Zusammenhänge und Gemeinsamkeiten finden lassen. Werden ähnliche Benutzer (*Empfehler* oder *Mentoren* genannt) entdeckt, so kann man von einer ähnlichen Präferenz ausgehen. Damit werden die Empfehlungen, die das System gibt, je größer die untersuchte Benutzergruppe ist, immer genauer auf den einzelnen abgestimmt.

Das Beispiel in Abbildung 2.4 zeigt ein Empfehlersystem, an das der grüne Kunde aus Abbildung 2.3 die gleiche Anfrage stellt. Da dieses System auf die Präferenzen der anderen Benutzer beruht und hier dem Mobiltelefon (Handy) die subjektive Eigenschaft „viel Freude bereiten“ unterstellt wird, ist die Empfehlung, anders als im vorherigen Beispiel, das Mobiltelefon. Hier zeigt sich, dass die anderen Kunden die Empfehlung aussprechen und das System sich nicht nur auf reines Faktenwissen bezieht. Empfehlersysteme gehen somit von einer Gruppenmeinung aus und werden daher laut BOHNERT auch als *Social Filtering* (gesellschaftliches Filtern) [Bo04] bezeichnet.

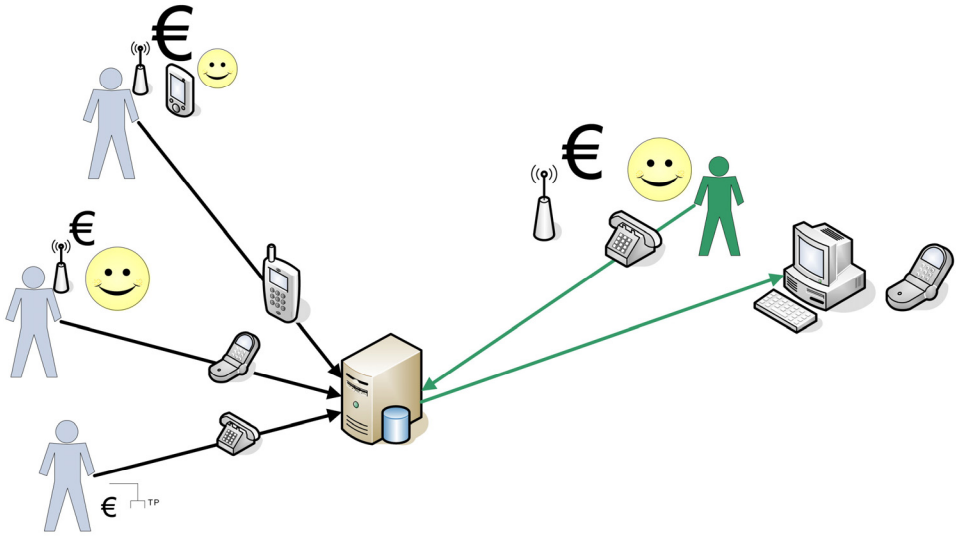


Abbildung 2.4: Empfehlersystem, eigene Darstellung

Da Empfehlersysteme oder Collaborative Filtering die höchste Form von Empfehlungssystemen darstellen und die Systeme noch relativ neu sind [Ru00], wird diese Art in der vorliegenden Arbeit genauer untersucht.

3 Collaborative Filtering

Das Collaborative Filtering (im Folgenden CF) lässt sich in zwei Arten einteilen: Das *Active Collaborative Filtering* und das *Automated Collaborative Filtering*. Automated CF unterscheidet zusätzlich in *Memory Based* CF und *Model Based* CF. Abbildung 3.1 zeigt die Zusammenhänge der Verfahren.

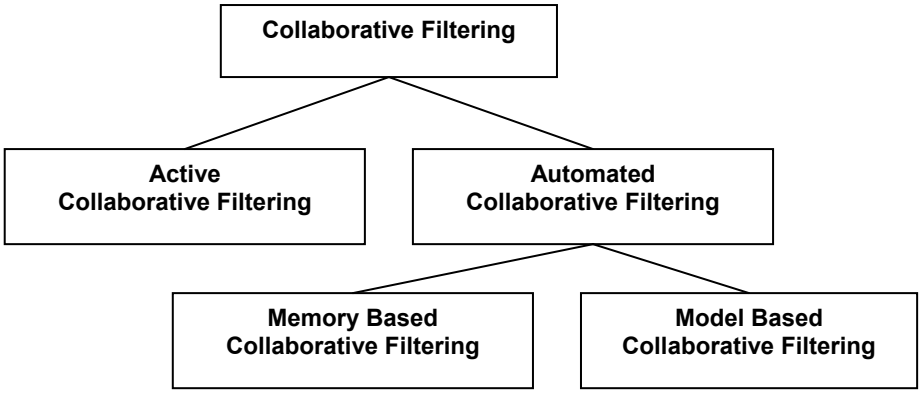


Abbildung 3.1: Active und Automated Collaborative Filtering [Ru00]

In diesem Kapitel werden die verschiedenen Ansätze genauer vorgestellt.

3.1 Active Collaborative Filtering

Das Active CF bezeichnet Ansätze, bei denen sich die Benutzer gegenseitig kennen und wurde erstmals von MALTZ und EHRLICH als Beitrag 1995 [ME95] vorgestellt. Das Verfahren geht davon aus, dass sich die einzelnen Benutzer persönlich kennen und sich gegenseitig *aktiv* Empfehlungen geben. RUNTE führt dazu den Begriff der *Push-Kommunikation* ein [Ru00]. Der Ansatz zeigt wenig innovative Eigenschaften und kann als Hilfsmittel für ein Netzwerk von Benutzern gesehen werden, verglichen mit einem Gespräch zwischen zwei Kunden an der Ladentheke.

3.2 Automated Collaborative Filtering

Interessanter sind die Automated CF-Ansätze, bei denen sich die einzelnen Benutzer nicht zwingend kennen müssen, da die Empfehlung der *Pull-Kommunikation* [Ru00] unterliegt und sich somit der Einzelne seine Empfehlung aus einer Datenbank holt und diese durch *mathematische* oder *regelbasierte Verfahren* [Ru00] berechnet wird. Diese Verfahren werden im Folgenden genauer betrachtet.

Bei Automated CF kann man, wie Abbildung 3.1 zeigt, *speicherbasierte* (Memory Based) und *modellbasierte* (Model Based) Systeme voneinander abgrenzen. Dabei hängt die Bezeichnung mit der Art der verwendeten Algorithmen zusammen.

Um CF betreiben zu können müssen Datensätze über eine große Anzahl von Benutzern vorliegen. Diese Datensätze enthalten die persönlichen Präferenzen der Benutzer im Bezug auf einzelne Objekte. Die Daten können dabei auf zwei verschiedene Arten erhoben werden, entweder *explizit* über Befragungen oder *implizit* durch permanente Überwachung und Aufzeichnung von Benutzeraktivitäten. Wenn ein aktiver Benutzer eine Empfehlung erhalten möchte, so können seine persönlichen Daten mit den Daten anderer Benutzer verglichen und ähnliche Benutzer (Mentoren) gefunden werden. Durch das Entdecken von Mentoren können dann die Empfehlungen dieser Mentoren als Basis zur Berechnung der Prognose dienen.

3.2.1 Model Based Collaborative Filtering

Die modellbasierten Verfahren des CF schätzen die Parameter eines Modells anhand einer Datenmatrix. Durch das gewonnene Modell werden dann Prognosen für den aktiven Benutzer abgegeben. Der Vorteil dieser Verfahren liegt darin, dass nur während der Erstellung des Modells auf die Daten zugegriffen wird. Während der Empfehlungsabgabe ist kein großer Zugriff nötig und der Rechenaufwand bleibt gering. RUNTE sieht den Vorteil wie folgt:

„Der Vorteil des modellbasierten Collaborative Filtering liegt in der Tatsache begründet, dass die teilweise sehr aufwendige Benutzermodellierung off-line erfolgen kann, also ohne dass eine zeitkritische Anfrage an das Recommender-System vorliegt [Ru00].“

Allerdings zeigt der modellbasierte Ansatz auch Schwächen. So werden die Informationen bei der Modellbildung verdichtet, was einen Informationsverlust zur Folge hat. Dieser Informationsverlust tritt bei speicherbasierten Verfahren nicht auf, da diesen Verfahren zu jedem Zeitpunkt die komplette Datenbasis zur Verfügung steht.

Im Folgenden werden vier mögliche Verfahren des modellbasierten Ansatzes nur kurz vorgestellt, da der Schwerpunkt dieser Arbeit auf speicherbasierten Verfahren liegt.

- **Cluster-Modelle:** Cluster-Modelle gehen davon aus, dass sich die einzelnen Benutzer in Gruppen einteilbar sind und es innerhalb einer Gruppe annähernd die gleichen Präferenzen gibt. Da einem Mitglied einer Gruppe Empfehlungen nach einem Modell gegeben werden, beschränkt sich der eigentliche Algorithmus auf die probabilistische Zuordnung der Benutzer zu den einzelnen Gruppen und der darauf aufbauenden Berechnung der *Präferenz-Wahrscheinlichkeit* eines Objektes. Verschiedene Cluster-Verfahren werden unter anderem von BOUTILIER und ZEMEL [Bo03] sowie von UNGAR und FOSTER [Un98] vorgestellt.

- **Bayes'sche Netze:** Als weitere Möglichkeit werden in der Literatur Bays'sche Netze vorgestellt [Br98]. Bays'sche Netze werden zur Erstellung von Entscheidungsbäumen mittels geeigneter Lernalgorithmen eingesetzt. Ein Entscheidungsbaum zeigt nach seiner Erstellung die Wahrscheinlichkeiten an, mit der beim durchlaufen des Baums ein bestimmter Weg genommen wird. Beim Einsatz dieses Verfahrens im CF bedeuten Knoten Objekte und die Kanten stehen für die Ratings der nächsten Knoten. Eine Einführung in Bays'sche Netze gibt beispielsweise SCHLIEDER [Sc03]. Ein einfaches Verfahren zum Einsatz im CF zeigen MIYAHARA und PAZZANI [Mi02].

- **Hauptkomponenten-Analyse**: Die Hauptkomponenten-Analyse wird von GUPTA ET AL. 1999 vorgestellt und im Recommender-System Jester eingesetzt [Gu99]. Bei diesem Verfahren werden die Präferenzen der Benutzer von zehn Globalurteilen auf zwei Eigenvektoren verdichtet, anhand derer die individuelle Prognose erstellt wird [Ru00].

- **Neuronale Netze:** Neuronale Netze stellen das letzte modellbasierte Verfahren dar, das in dieser Arbeit vorgestellt wird. Der Einsatz Neuronaler Netze im CF ist noch wenig untersucht, jedoch besteht die Möglichkeit, dass auch diese Verfahren einsetzbar sind. Diese Verfahren bilden ein menschliches Gehirn ab und versuchen die Vorgänge im Gehirn nachzuahmen. Das Netz besteht aus Neuronen, welche einen Input und einen Output besitzen. Der Input wird gewichtet und danach berechnet sich der Output über vorgegebene Algorithmen. Die Forschung im Bereich der Neuronalen Netze steht noch am Anfang. Eine Einführung in Neuronale Netze geben RUSSEL und NORVIG [Ru03].

3.2.2 Memory Based Collaborative Filtering

Neben dem Vorteil der off-line Verarbeitung der Daten haben die modellbasierten Systeme den Nachteil der Informationsverdichtung [Ru00]. Daher werden im Folgenden speicherbasierte Verfahren genauer betrachtet. Dabei wird der Ablauf von speicherbasierten Verfahren kurz skizziert. Danach werden *distanzbasierte* und *korrelationsbasierte* Ansätze vorgestellt und mit einem *naiven Ansatz* verglichen.

3.2.2.1 Ablauf des Memory Based Collaborative Filtering

Der Ablauf des Memory Based Collaborative Filtering ist bei allen speicherbasierten Verfahren gleich und lässt sich in vier Schritte einteilen, die in diesem Abschnitt genauer erklärt werden.

Speicherung der Daten

Über jeden Benutzer sind unterschiedliche Fakten bekannt und nicht jeder Benutzer kennt jedes Objekt. Um die Benutzer vergleichen zu können braucht man eine Basis, auf der man arbeiten kann. Diese Datenbasis ist die Objektmenge. Den einzelnen Objekten müssen nun Ausprägungen zugeordnet werden. Alle Merkmalsausprägungen der Objekte werden in einer Datenmatrix $U = (u_{ij})_{M,N}$ zusammengefasst, deren Zeilen durch die Objekte und deren Spalten durch die Merkmale repräsentiert werden [Ba95].

$$U = (u_{ij})_{M,N} = \begin{bmatrix} u_{11} & \cdots & u_{1j} & \cdots & u_{1N} \\ \vdots & & \vdots & & \vdots \\ u_{i1} & \cdots & u_{ij} & \cdots & u_{iN} \\ \vdots & & \vdots & & \vdots \\ u_{M1} & \cdots & u_{Mj} & \cdots & u_{MN} \end{bmatrix}$$

Die Matrix zeigt also alle Objekte und Merkmale. Ist ein Merkmal unbekannt, so wird ein (·) eingefügt. Da die Matrix nicht alle Werte zwingend enthalten muss, wird diese Art der Analyse auch *Missing-Value-Analyse* genannt.

Bei den Empfehlersystemen liegen keine Merkmalsausprägungen im Sinne der eben vorgestellten Missing-Value-Analyse vor. Auch die Objekte werden anders verstanden. So ist das Ziel eines CF-Systems, dem Benutzer eine Empfehlung zu geben und das Rating eines ähnlichen Benutzers heranzuziehen. Um dies zu erreichen, werden die Spalten und Zeilen neu zugeordnet. Die Matrix U stellt nun die Benutzer $I = \{1,...M\}$ und deren Rating der Objekte $J = \{1,..., N\}$ dar.

Anhand der Matrix und eines Anfragevektors lässt sich dann ein aktiver Benutzer mit den Mentoren vergleichen. Das Typische Verfahren zur Empfehlung von Objekten lässt sich vereinfacht in vier Schritte einteilen, die RUNTE [Ru00] wie folgt darstellt:

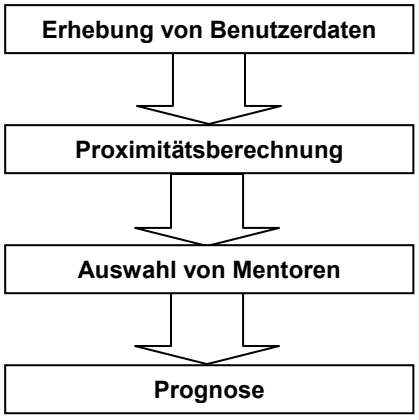


Abbildung 3.2: Typische Verfahrenselemente beim Collaborative Filtering [Ru00]

Erhebung von Benutzerdaten

Die Erhebung der Benutzerdaten ist der erste Schritt des CF. Um später möglichst genaue Empfehlungen zu geben, ist es wichtig, dass die Daten über den aktiven Benutzer sehr sorgfältig ausgewählt werden. Daher empfiehlt sich eine explizite Befragung des aktiven Benutzers.

Da die erhobenen Daten im nächsten Schritt mit den vorhandenen Mentoren-Daten verglichen werden sollten, um so ähnliche Benutzer zu finden, ist es von Vorteil, wenn die Rating-Vektoren sich stark überlappen. RUNTE schlägt daher vor, dass man:

- Objekte bewerten lässt, von denen man sich eine *hohe Bekanntheit* verspricht, um so möglichst viele vergleichbare Benutzer zu entdecken,
- möglichst viele *gegensätzlich bewertete Objekte* betrachtet, um die Ähnlichkeiten und Unähnlichkeiten der Benutzer stärker aufzuzeigen und dass man
- *unbekannte Objekte* bewertet, um solchen Objekten eine „Chance" zu geben und sie überhaupt in die Bewertung mit aufzunehmen [Ru00].

Als Aufteilung der vorgelegten Objekte schlagen HILL, ROSENSTEIN und FURNAS [Hi95] eine Gewichtung von 50% bekannten zu 50% unbekannten Objekten vor. Es wären jedoch auch denkbar ein anderes Verhältnis zu wählen. Die Wahl der Gewichtung hängt mit dem vorliegenden Sachverhalt zusammen und kann je nach Einsatzgebiet abweichen.

Die gewonnenen Benutzerdaten bilden die Grundlage für die nächsten Schritte, welche nun genauer dargestellt werden.

Berechnung der Proximität

Nachdem die Benutzerdaten vorliegen geht das Verfahren über in die zweite Phase, der *Proximitätsberechnung*. Grundlegend kann das Verfahren als eine Art *Clusteranalyse*[36] verstanden werden, wobei der Unterschied darin besteht, dass die Clusteranalyse die Ähnlichkeiten aller Objekte zu einander untersucht und das CF nur die Ähnlichkeit aller Objekte (hier Benutzer) zum aktiven Objekt (aktiver Benutzer). Im Vergleich zur Clusteranalyse ist es also nicht notwendig eine *Proximitäts-Matrix* zu betrachten, sondern vielmehr einen *Proximitäts-Vektor,* der zur Messung der Ähnlichkeit eines Benutzers mit anderen berechnet wird.

Da im CF nur Ähnlichkeiten betrachtet werden, ist es zudem notwendig, dass Unähnlichkeitsmaße in Ähnlichkeitsmaße transformiert werden [Ru00]. Ein weiteres Problem stellen die Lücken in den Vektoren dar (Missing-Value-Problematik). Die fehlenden Werte müssen in der Bewertung berücksichtigt werden.

---

[36] Clusteranalyse: Verfahren, durch das große Elementmengen durch Bildung homogener Klassen u. Gruppen sinnvoll strukturiert werden sollen. [Wi04] Vgl. auch Bacher [Ba96].

Mentorenauswahl

Zur Mentorenauswahl liegen nun die Ähnlichkeitsmaße der bekannten Benutzer zum aktiven Benutzer vor. Dazu muss die Ähnlichkeit zwischen Benutzer *i* und aktiven Benutzer *a* berechenbar sein und der Benutzer *i* sollte mindestens ein Objekt in seiner Bewertung haben, dass Benutzer *a* nicht bewertet hat, um ihm eine Empfehlung geben zu können. Die Überlappungen im Benutzerprofil sollten so groß sein, dass eine Mindestähnlichkeit eingehalten werden kann. [Ru00]

Prognoseberechnung

Nachdem die Mentoren gefunden wurden, wird im nächsten Schritt eine Auswahl von *Objekt-Mentoren* vorgenommen. Da diese Auswahl die Objekte, insbesondere die Lücken im Rating-Vektor des aktiven Benutzers, betrifft, werden die Missing Values sequenziell durchlaufen und für jede Lücke eine Prognose erstellt. Den so vervollständigten Vektor nennt RUNTE *Prognose-Vektor* [Ru00].

3.2.2.2 Ansätze des Memory Based Collaborative Filtering

Die vier vorgestellten Schritte bleiben bei allen speicherbasierten Verfahren erhalten. Jedoch ändert sich je nach Ansatz die Art der Berechnung und Bearbeitung der einzelnen Phasen. Ausgehend von einem naiven Ansatz werden nun zwei weitere Ansätze vorgestellt.

Naiver Ansatz

Ein naiver Ansatz zur Ermittlung einer Prognose für den aktiven Benutzer ist beispielsweise die *Mittelwertberechnung*. Bildet man über alle verfügbaren Bewertungen das *arithmetische Mittel*, so liefert das Verfahren jedoch auch für alle Benutzer die identischen Werte zurück:

$$f_{ai} = \frac{\sum_{i \in I} v_{ij} u_{ij}}{\sum_{i \in I} v_{ij}} := \bar{u}_j$$

Dabei wird die Prognose der Präferenz f_{ai} des aktiven Benutzers a für ein Objekt $j \in J$ auf Basis des arithmetischen Rating-Mittelwerts $\bar{u}_j$ für dieses Objekt über alle Benutzer $i \in I$ berechnet [Ru00]. RUNTE führt die Indikatorvariable v_{ij} ein, für die gilt: $v_{ij} = 1$, wenn Benutzer i Objekt j bewertet hat und damit ein Wert für u_{ij} existiert, sonst gilt $v_{ij} = 0$.

Erweiterung des naiven Ansatzes

Da die speicherbasierten Systeme im Vergleich zu den modellbasierten Systemen jede Empfehlung neu berechnen sind diese Verfahren sehr rechenintensiv. Daher sollte vor dem Einsatz dieser Systeme genau untersucht werden, wie hoch der Zusatznutzen im Vergleich zu dem weniger aufwendigen Mittelwertverfahren ist. Der naive Ansatz soll daher als Maßstab bei der Betrachtung von speicherbasierten CF-Verfahren dienen, von welchen zwei genauer vorgestellt werden.

Die beiden Ansätze versuchen die Unabhängigkeiten zwischen Benutzer-Ratings und den Objekten fallen zu lassen. Dadurch verspricht man sich genauere Prognosen. Um diese Abhängigkeit zu erreichen ist es nötig, Benutzer (Mentoren) zu finden, die das zu untersuchende Objekt bewertet haben und eine ausreichende Ähnlichkeit zum aktiven Benutzer besitzen.

Ziel ist es eine Prognose abzugeben, die die Ratings „ähnlicherer" Mentoren stärker gewichtet, als Ratings von „unähnlicheren" Mentoren. Dazu wird das Ähnlichkeitsmaß s_{ai} eingeführt, das die Ähnlichkeit von aktiven Benutzer a zu Mentor i zeigt.

Ist das Ähnlichkeitsmaß bekannt, so kann der Prognosewert f_{ai} als gewichteter Mittelwert berechnet werden [Ru00]:

$$f_{ai} = \frac{\sum_{i \in \tilde{M}_{aj}} s_{ai} u_{ij}}{\sum_{i \in \tilde{M}_{aj}} s_{ai}}, j \in J$$

Für die Berechnung sind die *Menge der Mentoren* des aktiven Benutzers *a*

$$M_a = \{i \in I, i \neq a : \sum_{j \in J} v_{aj} v_{ij} \geq 3\}$$

und die Menge der *Objektmentoren* für das Objekt *j* nötig.

$$\tilde{M}_{aj} = \{i \in M_a : s_{ai} v_{ij} > 0\}$$

Die Objektmentoren sind dabei alle Mentoren, die das Objekt j bewertet haben und zum aktiven Benutzer eine positive Ähnlichkeit besitzen [Bo04]. Für die Menge der Mentoren M_a schlägt RUNTE vor, nur Benutzer *i* aufzunehmen, deren Rating-Vektoren mindestens drei Überlappungen zum Rating-Vektor des aktiven Benutzers *a* aufweisen [Ru00].

Falls keine Objektmentoren existieren, also $\tilde{M}_{aj} = \emptyset$ gilt, schlägt RUNTE vor, den naiven Ansatz des arithmetischen Mittels zu benutzen [Ru00]. Wenn für ein Objekt *j* keine einzige Bewertung existiert, so kann f_{ai} nicht berechnet werden.

Um die Objektmentoren zu berechnen ist es also nötig die Ähnlichkeiten s_{ai} zu kennen. Zur Berechnung dieser Ähnlichkeiten werden im Folgenden zwei Verfahren vorgestellt.

Distanzbasierter Ansatz

Die distanzbasierten Ansätze berechnen die Ähnlichkeit s_{ai} über ein *Distanzmaß*, wie beispielsweise der *mittleren euklidischen Distanz* d_{ai}:

$$d_{ai} = \frac{\sqrt{\sum_{j \in J} v_{aj} v_{ij} (u_{aj} - u_{ij})^2}}{\sum_{j \in J} v_{aj} v_{ij}} \text{ für } i \in M_a$$

Da die mittlere euklidische Distanz ein Ausdruck für die Distanz zwischen zwei Benutzern ist, muss sie noch in ein *Ähnlichkeitsmaß transformiert* werden. Dazu schlägt Runte vor, die Ähnlichkeitsmaße auf ein Intervall [0;1] abzubilden [Ru00]:

$$s_{ai} = 1 - \frac{d_{ai}}{t} \text{ mit } t = \max\{d_{ai} \mid i \in M_a\}$$

Der Parameter *t* dient bei der Berechnung als linearer Skalierungsparameter, der den Wertebereich zwischen 0 und 1 erzeugt, falls mehrere d_{ai} mit unterschiedlichen Werten existieren.

Korrelationsbasierter Ansatz

Bei den distanzbasierten Ansätzen wird im ersten Schritt ein Distanzmaß berechnet, welches im zweiten Schritt in ein Ähnlichkeitsmaß transformiert wird. Korrelationsbasierte Ansätze versuchen die Ähnlichkeit direkt über ein Ähnlichkeitsmaß zu berechnen. Eine Möglichkeit dafür ist der *Pearsonsche Korrelationskoeffizient* q_{ai}, der in den Systemen GroupLens[37] [Re94] und Ringo[38] [Sh95] Verwendung findet. Dabei werden bei der Berechnung von q_{ai} alle Bewertungen beachtet, die Benutzer *i* und *a* überlappend bewertet haben. Dazu geht Runte wieder von einer minimalen Überlappung von drei Objekten aus, wie oben bereits beschrieben. Um q_{ai} zu berechnen wird zunächst das arithmetische Mittel

$$\bar{u}_{ai} = \frac{\sum_{j \in J} v_{aj} v_{ij} u_{aj}}{\sum_{j \in J} v_{aj} v_{ij}}$$

berechnet über alle Bewertungen u_{aj} des Benutzers *a*, für die auch bei Benutzer *i* die Bewertungen vorliegen. Danach kann der Korrelationskoeffizient q_{ai} bestimmt werden.

[37] GroupLens: Bewertungssystem für Newsgroup-Beiträge. http://www.grouplens.org/
[38] Ringo: Bewertungssystem für Musik.

$$q_{ai} = \frac{Q_{ai}^{Z}}{Q_{ai}^{N}} = \frac{\sum_{j \in J} v_{aj} v_{ij} (u_{aj} - \bar{u}_{ai})(u_{ij} - \bar{u}_{ia})}{\sqrt{\sum_{j \in J} v_{aj} v_{ij} (u_{aj} - \bar{u}_{ai})^2 \cdot \sum_{j \in J} v_{aj} v_{ij} (u_{ij} - \bar{u}_{ia})^2}}, i \in M_a, Q_{ai}^{N} > 0$$

Um nur positive Korrelationswerte zu berücksichtigen wird folgende Definition eingeführt:

$$s_{ai} = \begin{cases} q_{ai}, q_{ai} > 0 \\ 0, q_{ai} \leq 0 \end{cases}$$

Außerdem können im Nenner Nullwerte auftreten; um dieses Problem zu umgehen wird falls $Q_{ai}^{N} = 0$ der Ähnlichkeitswert $s_{ai} = 0$ gesetzt.

Neben den beiden vorgestellten Basisverfahren werden unter anderem von BRESSE, HECKERMAN und KADIE in [Br98] noch die Möglichkeiten *Vector Similarity, Inverse User Frequency* und *Case Amplification* vorgestellt.

Bewertungsmöglichkeiten der Ansätze

Zur Bewertung der Verfahren gibt es mehrere Möglichkeiten, für die der naive Ansatz als Vergleich zur Verfügung steht. So kann beispielsweise zur Messung der *Prognosegenauigkeit* der *mittlere absolute Prognosefehler* herangezogen werden. Prognosefehler lassen sich über die Differenz zwischen vorhergesagten und tatsächlichen Bewertungen berechnen.

Als weitere Möglichkeit nennt BOHNERT [Bo04] die *Coverage*, welche den Anteil der Objekte angibt, für die überhaupt Prognosen erstellt werden können. Die Coverage vergleicht die Anzahl berechenbarer Prognosen des Algorithmus mit denen des naiven Ansatzes und kann so als Gütemaß herangezogen werden.

RUNTE stellt im Zusammenhang mit der Bewertung von CF-Verfahren noch die Maße *Sensitivität* und *Spezifizität* zur Beschreibung der *Leistungsfähigkeit eines Filters* vor. Wobei die Sensitivität die Wahrscheinlichkeit angibt, mit der ein relevantes Objekt vom Filter akzeptiert wird. Die Spezifizität gibt an, mit welcher Wahrscheinlich ein irrelevantes Objekt gefiltert und somit dem Benutzer nicht angezeigt wird [Ru00].

4 Einsatz von Collaborative Filtering am Beispiel von Amazon.de

Wie bereits erwähnt, werden CF-Verfahren oft von Internet-Shops eingesetzt. Amazon setzt weltweit das NetPerceptions[39]-System ein, das auf GroupLens basiert. Das System von Amazon stellt ein typisches CF-System dar und soll im Folgenden kurz vorgestellt werden.

Betritt man Amazon.de zum ersten Mal, so wirkt das Angebot wenig personalisiert. Allein die Meldung *„Kunden, die dieses Buch gekauft haben, haben auch diese Bücher gekauft“* kann als Art Empfehlungsabgabe gelten. Die volle Leistungsfähigkeit bemerkt der Kunde erst, nachdem er sich als Kunde angemeldet hat, und so dem System „bekannt“ ist. Gleich nach der Anmeldung kann man sich eine Empfehlung des Systems erstellen lassen. Da jedoch noch keine Daten bekannt sind, erscheint folgende Meldung:

Abbildung 4.1: Amazon.de – Empfehlung nach Neu-Anmeldung

An dieser Bildschirmmeldung lassen sich bereits die zwei Verfahren erkennen, mithilfe derer Amazon.de Benutzerdaten erfasst:

- **„Anhand der Seiten, die Sie bei Amazon.de besucht haben, …“:** Hier lässt sich die *implizite* Erhebung von Benutzerdaten erkennen. Das Surfverhalten auf Amazon.de hat also direkten Einfluss auf die Benutzer und deren Bewertung.

- **„Bitte bewerten Sie eine der unten stehenden Artikel oder wählen Sie Ihre bevorzugten Interessensgebiete aus, …“:** Der zweite Hinweis zeigt eine Aufforderung zur *expliziten* Bewertung. Durch das Beantworten eines Fragenkatalogs oder das Bewerten von Artikeln wird der Benutzer „kennen gelernt“.

[39] NetPerceptions: Kommerzielles Collaborative Filtering-System. http://www.netperceptions.com/

Um eine gute Empfehlung zu erhalten ist es also ratsam die explizite Befragung des Systems zu durchlaufen. Dabei werden zuerst die Shops (Bücher, DVD, Software, …) eingegrenzt und im nächsten Schritt soll der Kunde seine bevorzugten Kategorien (Computer & Internet, Fachbücher, …) in jedem ausgewählten Shop angeben. Danach besteht werden noch Fragen nach Lieblingsautor oder der Art der verwendeten Software gestellt, also Fragen, um das Profil des Kunden noch genauer zu ermitteln.

Im letzten Schritt werden dann verschiedene Bücher angezeigt, die der Kunde auf einer Skala von 1 (Mag ich nicht) bis 5 (Finde ich ganz toll!) oder „?“ (Nicht bewertet) bewerten soll. Zusätzlich ist es möglich „Gehört mir“ anzugeben.

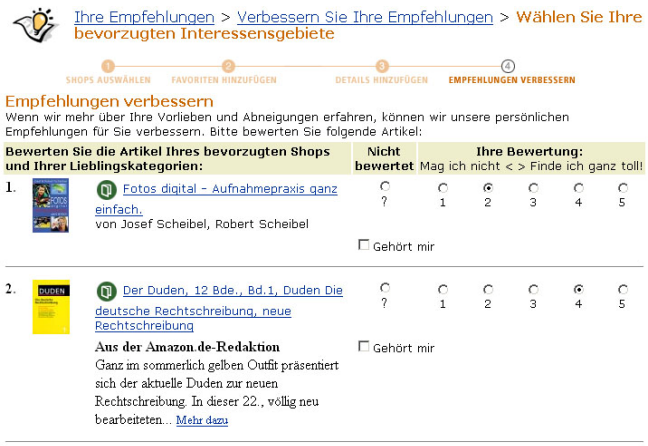


Abbildung 4.2: Buchbewertungskatalog von Amazon.de nach Neu-Anmeldung

Bei RUNTE wurde vorgeschlagen, Objekte bewerten zu lassen, von denen man sich *hohe Bekanntheit* verspricht (Vgl. Kapitel 3.2.2.1). Dies zeigt sich auch in der Befragung durch Amazon.de. Wie Abbildung 4.2 zeigt, wird in diesem Fall eine Bewertung des Dudens vorgenommen.

Nachdem alle Fragen des Systems durch den Benutzer beantwortet wurden, kann Amazon.de Empfehlungen abgeben. Bei jedem Besuch des Benutzers werden anschließend neue Daten erhoben. So besteht beispielsweise jederzeit die Möglichkeit sein persönliches Profil abzuändern oder Artikel explizit zu bewerten. Durch das ständige Protokollieren des Surfverhaltens auf der Seite werden weitere Interessen des Kunden entdeckt. Auch das Einkaufsverhalten geht selbstverständlich in die Kundendaten mit ein. Der Kunde wird also mit jedem Besuch auf der Seite „bekannter“ und die Empfehlungen werden immer genauer auf die Kundenwünsche abgestimmt.

5 Fazit

Diese Arbeit hat gezeigt, dass Collaborative Filtering-Verfahren im Vergleich zu den Feature Based Filtering-Verfahren gewisse Vorteile haben. So werden *Beziehungen zwischen Benutzer und Objekt* aufgedeckt, die *subjektiven Eindrücke* werden beachtet und die *Ermittlung der Objekteigenschaften entfällt*. CF-Verfahren ermöglichen so einen *Erfahrungsaustausch zwischen einer hohen Anzahl an Benutzern*.

Die CF-Verfahren zeigen allerdings auch Schwächen gegenüber den eigenschaftsbasierten Systemen. Das erste Problem, *Kaltstart-Problematik* genannt, zeigt sich in der Beschaffenheit der Systeme. Da sich Empfehlungen auf andere Benutzer stützen, muss ein *Minimum an Benutzerprofilen* bekannt sein, um Empfehlungen abgeben zu können. Auch die *Empfehlung von neuen Objekten* stellt ein Problem dar, da diese neuen Objekte noch keine Bewertung erhalten haben. Das Entfallen der Objekteigenschaften, eben noch als Vorteil gesehen, kann auch ein Problem darstellen. So kann es durchaus zu schlechteren Empfehlungen kommen, weil die *Eigenschaften der Objekte nicht beachtet* werden. Außerdem besteht die Gefahr, dass durch *zufällige Zusammenhänge* schlechte Prognosen abgegeben werden. Dies kann passieren, wenn zwei Benutzer zufällig in mehreren Punkten übereinstimmen und so als sehr ähnlich angesehen werden, in anderen Bereichen jedoch stark voneinander abweichen.

Typische Einsatzgebiete für CF-Verfahren sind alle Gebiete, in denen die subjektiven Eindrücke in der Bewertung und Empfehlung von Objekten einfließen. Zu nennen sind dabei beispielsweise Empfehlungssysteme für *Literatur, Musik, Filme, Webseiten, Restaurants*. In anderen Bereichen, in denen die Objekteigenschaften im Vordergrund stehen, stellen die eigenschaftsbasierten Systeme eine bessere Alternative dar.

Die Schwächen der CF-Verfahren liegen meist dort, wo die eigenschaftsbasierten Systeme ihre Stärken haben und umgekehrt. Um die Schwächen beider Verfahren auszugleichen wird an neuen, kombinierten Verfahren geforscht. So nennt BOHNERT *Feature Based Guided Collaborative Filtering* und *Content Based Collaborative Filtering* [Bo04] als mögliche kombinierte Verfahren.

Literaturverzeichnis

[Ba95] Bankhofer, U.: Unvollständige Daten und Datenmatrizen in der Multivariaten Datenanalyse. Köln, 1995.

[Ba96] Bacher, J.: Clusteranalyse. Oldenbourg, 1996.

[Bo03] Boutilier, C.; Zemel, R.: Online Queries For Collaborative Filtering. In Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics. Toronto, 2003. http://research.microsoft.com/conferences/aistats2003/proceedings/171.pdf (letzter Zugriff am 02.07.04).

[Bo04] Bohnert, F.: Einsatz von Collaborative Filtering zur Datenprognose. http://www.mathematik.uni-ulm.de/sai/ws03/dm/arbeit/bohnert.pdf (letzter Zugriff am 30.06.04).

[Br98] Breese, J.; Heckerman, D.; Kadie, C.: Empirical Analysis Of Predictive Algorithms For Collaborative Filtering. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. 1998.
http://research.microsoft.com/users/breese/cfalgs.html (letzter Zugriff am 30.06.04).

[Gu99] Gupta, D. et al.: Jester 2.0. Evaluation Of A New Linear time Collaborative Filtering Algorithm. In Proceedings of the SIGIR. ACM, 1999.

[He89] Hey, J.B.: U.S. Patent 4,870,579 - System and Method of Predicting Subjective Reactions, Concord, MA, USA, 1989.

[Hi95] Hill, W.; Rosenstein, M.; Furnas, G.: Recommending and Evaluating Choices in a Virtual Community of Use, Human Factors in Computing Systems, CHI '95 Conference Proceedings. Denver, Colorado, USA, 1995.

[Le03] Lehner, F.: MobiLex – Lexikon und Abkürzungsverzeichnis für Mobile Computing und mobile Internetanwendungen, Forschungsbericht Nr. 51, 3. Auflage. Universität Regensburg, 2003.

[ME95] Maltz, D.; Ehrlich, K.: Pointing The Way: Active Collaborative Filtering, Human Factors in Computing Systems, CHI '95 Conference Proceedings. Denver, Colorado, USA, 1995.

[Mi02] Miyahara, K.; Pazzani, M. : Improvement of Collaborative Filtering with the Simple Bayesian Classifier. IPSJ Journal, Vol.43, No.11, Information Processing Society of Japan. 2002.
http://www.ics.uci.edu/~pazzani/Publications/IPSJ.pdf (letzter Zugriff am 01.07.04).

[Re94] Resnick, P. et al.: GroupLens: An Open Architecture For Collaborative Filtering Of Netnews. In Proceedings of the 1994 Computer Supported Collaborative Work Conference. 1994.

[Ru00] Runte, M.: Personalisierung im Internet, Individualisierte Angebote mit Collaborative Filtering. Deutscher Universitätsverlag, 2000.

[Ru03] Russel S.; Norvig P.: Artificial Intelligence: A Modern Approach (Second Edition). Prentice Hall, 2003.

[Sc03] Schlieder, C.: Skript zur Veranstaltung Kulturinformatik I: Semantische Informationsverarbeitung.
http://www.kinf.wiai.uni-bamberg.de/teaching/lehre_ws_2003-2004/KultInfI%20-%20Semantische%20Informationsverarbeitung/semantische_informationsverarbeitung_ws03-04.htm (letzter Zugriff am 01.07.04).

[Un98] Ungar, L.; Foster, D.: Clustering Methods For Collaborative Filtering, AAAI Workshop on Recommendation Systems. Philadelphia, 1998.
http://www.cis.upenn.edu/datamining/Publications/clust.pdf
(letzter Zugriff am 02.07.04).

[Wi04] Wissen.de: Gesellschaft für Onlineinformation.
http://www.wissen.de (letzter Zugriff am 30.06.04).

Integration von kontextbasiertem Information Retrieval in Workflow-Managementsysteme

Daniela Neundorf, Matthias Raps

daniela.neundorf@stud.uni-bamberg.de
matthias.raps@stud.uni-bamberg.de

Abstract: Diese Arbeit befasst sich mit den Möglichkeiten einer Integration von kontextbasiertem Information Retrieval in Workflow-Managementsysteme. Einleitend werden die Hintergründe für die Einführung derartiger Systeme und die mit ihrem Einsatz verfolgten Ziele behandelt. Den Hauptteil der Arbeit bildet die Vorstellung zweier Projekte, die aufzeigen, wie Workflow-Kontext für die Bearbeitung von Geschäftsvorfällen genutzt werden kann. Die Zielsetzung des ersten Projekts KnowMore liegt dabei auf der pro-aktiven Informationsversorgung von Bearbeitern wissensintensiver Aufgaben. Dagegen zielt das VirtualOffice Projekt darauf ab, mit Hilfe von Workflow-Kontext die Posteingangsschnittstelle eines Unternehmens möglichst effizient zu automatisieren.

1. Motivation für den Einsatz von Workflow-Managementsystemen

Die heutige Wirtschaftspraxis ist gekennzeichnet durch kontinuierlichen Druck zu Kosteneinsparungen bei immer kürzer werdenden Produktentwicklungszyklen. Dies resultiert in der Notwendigkeit, betriebliche Abläufe fortwährend zu analysieren und zu optimieren sowie den Informationsfluss effektiv zu gestalten.

Für Unternehmen hat es sich in diesem Zusammenhang als erfolgreich herausgestellt, Geschäftsprozesse[40] explizit zu modellieren und so weit wie möglich computergestützt zu automatisieren, um sie in Workflow-Managementsystemen[40] ausführen zu können. Mit der Einführung derartiger Systeme verfolgen Unternehmen u.a. Ziele wie eine Vereinheitlichung und Verbesserung der Qualität von Prozessen, eine schnellere Bearbeitung von Geschäftsvorfällen bei gleichzeitiger Kostenreduktion sowie eine Erhöhung der Verfügbarkeit von Informationen.

Insbesondere letzterem Ziel widmet sich das nachfolgend vorgestellte Projekt KnowMore.

---

[40] siehe Glossar

2 Der KnowMore - Ansatz (Daniela Neundorf)

2.1 Überblick

Im Projekt KnowMore (Knowledge Management for Learning Organizations) realisiert das DFKI[41] einen Ansatz, der auf die Unterstützung eines Benutzers, welcher innerhalb eines Workflows[42] an einer wissensintensiven Aufgabe (‚Knowledge Intensive Task', kurz: KIT) arbeitet, abzielt. Dieser soll pro-aktiv - d.h. ohne eine explizite, detaillierte Anfrage stellen zu müssen - mit kontext-sensitiven, relevanten Informationen versorgt werden. Um dies zu ermöglichen, wird in KnowMore ein erweitertes Workflow-Modell eingesetzt, welches Informationsagenten unterstützt. Herkömmliche deklarative Beschreibungen einer Workflow-Aktivität werden mit Hilfe so genannter KIT-Beschreibungen um eine Unterstützungsspezifikation erweitert. In dieser werden entsprechende Informationsbedarfe in Form generischer Anfragen zusammen mit den für die spätere Bearbeitung zuständigen Informationsagenten spezifiziert. Zur Workflow-Laufzeit instanziiert und bearbeitet der Agent die ihm zugeordnete generische Anfrage und versorgt so den jeweiligen Benutzer mit relevanten Informationen. [ABM02]

2.2 Akquise und Speicherung von kontext-basierten Informationen

Konventionelle Workflow Management Systeme (WfMS) verfügen laut [Ma01] meist weder über ein umfassendes Repräsentationskonzept für Workflow–Kontext noch über eine komfortable Zugriffsmöglichkeit auf eben diesen zur Laufzeit. Deshalb ist es notwendig, die intelligente Informationsversorgung über die aus dem WfMS zugreifbaren Daten hinaus auf weitere Kontextquellen zu stützen.

Dennoch können [Ma01] zufolge in einem ersten Schritt einige der im WfMS verwalteten Daten in den Workflow-Kontext einfließen, so z.B. Workflow-Kontrolldaten. Diese umfassen Workflow-Instanz-ID, Aktivitäts-ID, aber auch den aktuellen Benutzer. Mit Hilfe dieser Informationen kann ein Informationselement (z.B. ein Dokument) mit der entsprechenden Workflow-Instanz und Aktivität verknüpft werden, innerhalb derer es entstanden ist. Über die Verwaltung des jeweiligen Benutzers kann ein Informationselement ferner im Retrievalprozess dazu herangezogen werden, das Profil eines Workflow-Teilnehmers aufzuspüren. Dadurch kann dem zu unterstützenden Nutzer ggf. ein Experte auf dem betreffenden Gebiet empfohlen werden. Definitionen von Workflows können mit den IDs der von Ihnen umfassten Aktivitäten verknüpft werden, um später allen entsprechenden Workflow- oder Aktivitätsinstanzen relevantes Material vorschlagen zu können.

[41] Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern
[42] siehe Glossar

Im WfMS verfügbare Prüfdaten (audit data) können während des eigentlichen Retrieval-Prozesses zum Auffinden anderer Workflow-Instanzen oder Aktivitäten mit vergleichbaren KIT-Beschreibungen verwendet werden. Zusätzlich können Workflow-Teilnehmern - anhand ihrer spezifischen Kompetenzen und Interessensprofile - aus dem Organigramm einer Organisation benutzerspezifische Sichten auf Workflow-Instanzen abgeleitet werden.

Insgesamt liegt der Nutzen der Speicherung des konkreten Entstehungskontextes eines Informationselements darin, dass die Suche auf diesen Daten in späteren, vergleichbaren Geschäftssituationen ein besseres Retrieval ermöglicht. Darüber hinaus kann über den zugreifbaren Kontext die Qualität der aufgespürten Informationen überprüft werden. So kann im Falle der Einbettung eines Workflows in ein Projekt z.B. über die Workflow-ID herausgefunden werden, ob das entsprechende Projekt erfolgreich war.

Jedoch reichen aufgrund der zu Beginn erwähnten Beschränkung konventioneller WfMS die genannten, vorhandenen Daten für eine pro-aktive Informationsversorgung, wie sie im KnowMore–Ansatz verfolgt wird, nicht aus. Um bereits zur Laufzeit eine intelligente Versorgung mit relevanten Informationen zu gewährleisten, ist es erforderlich, das Datenschema herkömmlicher WfMS um sogenannte KIT-Variablen zu erweitern. Hierdurch wird der konventionelle Datenfluss zu einem Informationsfluss. Die KIT-Variablen beschreiben den Kontext-Informationsfluss zwischen den Aufgaben im Workflow und fungieren als Informationskanal zwischen dem WfMS und den Informationsagenten. Zur Laufzeit beinhalten sie den relevanten Kontext wissensintensiver Aktivitäten, mit Hilfe dessen die generischen Anfragen instanziiert werden. Die KIT-Variablen müssen in die entsprechende Domänenontologie eingebettet sein[43], um die erforderliche Inferenz im Rahmen des intelligenten Retrieval zu ermöglichen. [ABM02], [Ma01]

Die gewünschte intelligente Informationsversorgung erfordert neben der Erweiterung des Datenschemas eine explizite Modellierung wissensintensiver Tätigkeiten. Diese muss nach [ABHKS98] von einem menschlichen Experten während der Geschäftsprozessdefinition vorgenommen werden. Dadurch wird die im Standardmodell der Workflow Management Coalition zur Verfügung gestellte deklarative Aufgabenbeschreibung (welche die Vorgänger und Nachfolger im Prozess, die entsprechende Aktivität mit zugehörigen Werkzeugen und Ressourcen sowie die zugreifbaren Workflow-Variablen umfasst) um die bereits im einleitenden Abschnitt beschriebene Unterstützungsspezifikation erweitert.

[43] d.h. die Typen ihrer Werte müssen gemäß [ABM02], S. 245 als Konzepte der Domänenontologie definiert sein

Die aufgeführten, entweder bereits im WfMS vorhandenen oder mit Hilfe von KIT-Variablen zusätzlich abzugreifenden, Kontextinformationen repräsentieren in ihrer Gesamtheit zur Laufzeit drei verschiedene Arten von Kontextquellen. Die instanziierten KIT-Variablen beherbergen den lokalen Kontext der entsprechenden Aufgabeninstanz, liefern also die erforderliche Information über die Umwelt der aktuellen Aktivität. Das Geschäftsprozessmodell als Ganzes repräsentiert den globalen Kontext der zu unterstützenden wissensintensiven Aufgabe, da ein Geschäftsprozess letztlich immer das Ziel der in ihm zusammengefassten Aufgabensequenz beschreibt. Ergänzend informieren die eingangs aufgeführten Workflow-Kontrolldaten über den Zustand der betrachteten Prozessinstanz. [ABHKS98]

Eine Abspeicherung von Kontextinformationen wird in KnowMore laut [ABMW00] immer dann vollzogen, wenn eine Workflow-Aktivität ein neues relevantes Informationselement erzeugt, das es festzuhalten lohnt. Auch hier wird wieder ein Agent aktiv, der den entsprechenden Kontext aus allen verfügbaren oben genannten Quellen apportiert. Der so gewonnene Entstehungskontext wird - mit dem zugehörigen Informations- bzw. Wissenselement verlinkt - in den Wissensbeschreibungen abgelegt, um so ein späteres kontext-bewusstes Retrieval zu ermöglichen.

2.3 Nutzermodell

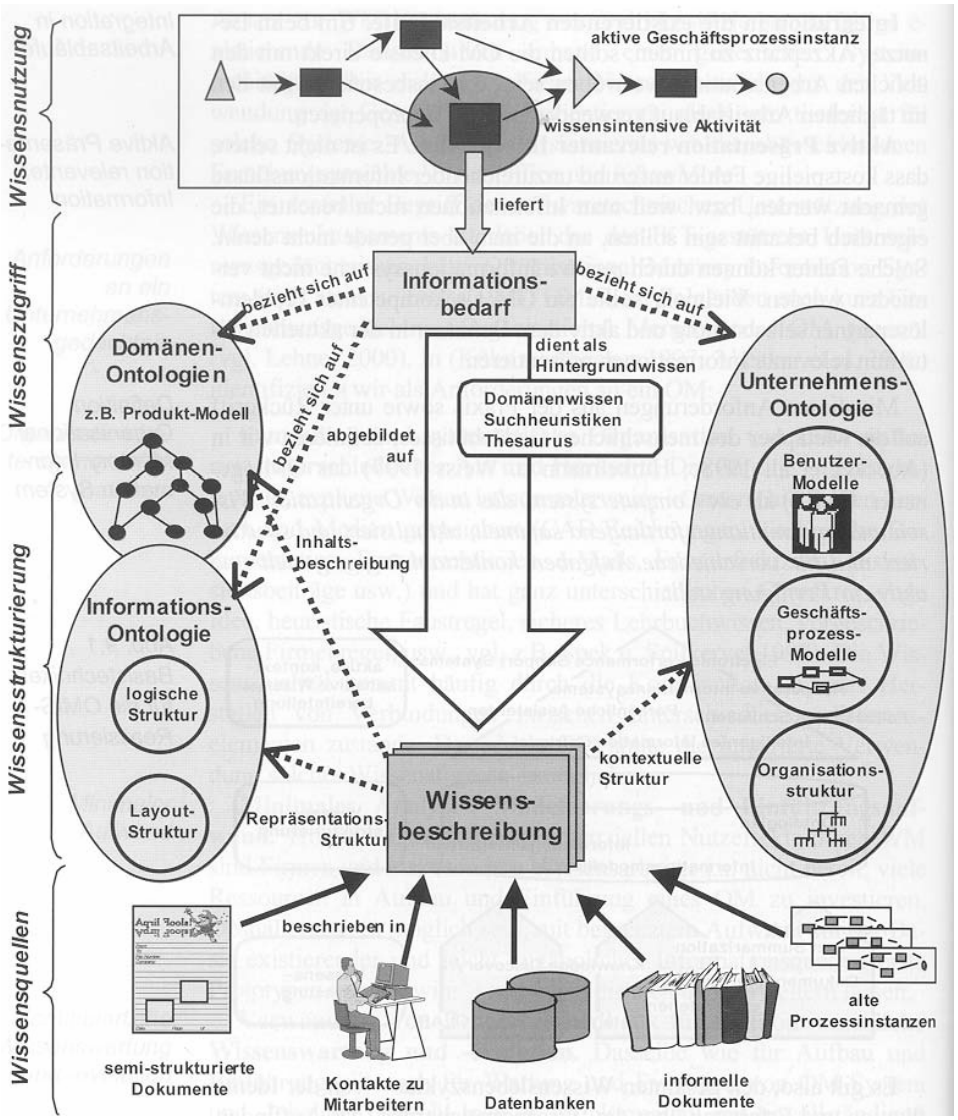


Abbildung 1: Generische Architektur für ein ontologiebasiertes OMIS; Quelle: [ABM02], S. 218

Abbildung 1 zeigt den generischen Architekturrahmen für ein ontologiebasiertes Organizational Memory Information System (OMIS), meist unter dem Begriff Unternehmensgedächtnis verwendet. Darunter verstehen [ABM02] „ein Computersystem, das in der Organisation Wissen und Informationen fortlaufend sammelt, aktualisiert und strukturiert und für verschiedene Aufgaben kontextabhängig, gezielt und aktiv zur Verfügung stellt“. Da ein derartiges Unternehmensgedächtnis typischerweise eine Vielzahl heterogener Wissens- und Informationsquellen enthält (verbunden mit unterschiedlichen Strukturen, Zugriffsmethoden und Inhalten), resultiert dies [Ma01] zufolge in der Notwendigkeit eines Repräsentationsschemas für eine einheitliche Wissensbeschreibung. Ein solches Repräsentationsschema wird durch die im folgenden Abschnitt genauer erläuterten formalen Ontologien zur Verfügung gestellt, auf deren Basis Struktur, Metadaten, Informationsinhalt und -kontext der Wissens- und Informationsquellen modelliert werden.

Die Gesamtheit der in Abbildung 1 dargestellten „karteikartenähnlichen“ Wissensbeschreibungen ersetzt den Index in klassischen IR-Systemen. Durch die ontologiebasierte Beschreibung wird ein präzises Inhaltsretrieval und damit eine kontextsensitive, pro-aktive Belieferung des Benutzers mit relevanten Informationen ermöglicht. [ABM02]

Ergänzend sollen an dieser Stelle nun die in einem OMIS verwendeten Ontologien erläutert werden. Von [ABM02] als „formale, explizite Modellierungen der in einer Gruppe von Akteuren allgemein anerkannten Konzeptualisierungen“ definiert, bilden Ontologien nicht nur die Basis einer gemeinsamen Sprache, sondern fungieren ferner als wichtige Quelle für Hintergrundwissen im Rahmen eines intelligenten Retrieval. Wie in nachfolgender Abbildung ersichtlich, unterscheiden die Autoren hierbei drei Arten von Ontologien – die Informationsontologie, Ontologien des Anwendungsbereichs (sog. Domänenontologien) und schließlich Ontologien der Unternehmung.

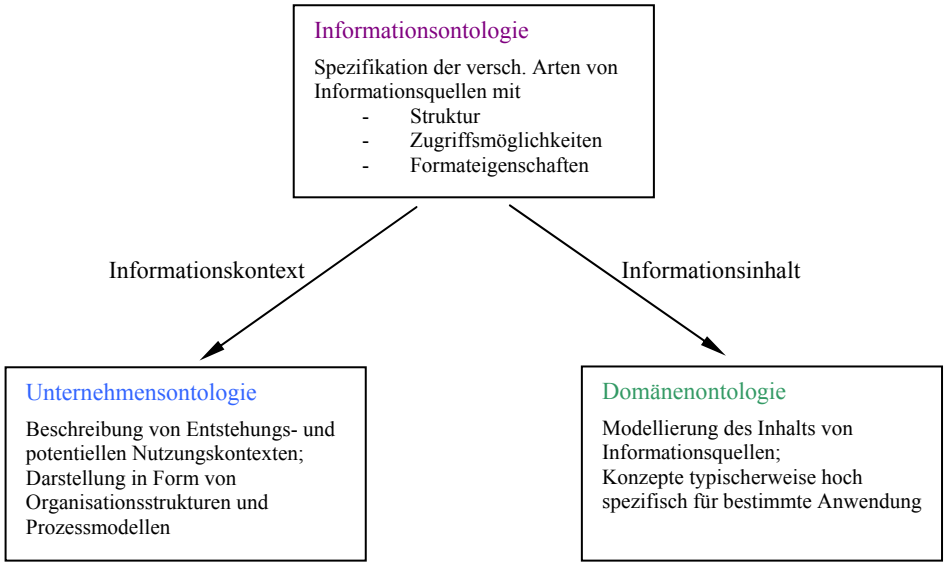


Abbildung 2: Dimensionen der Wissensbeschreibung

Jedes Informations- und Wissenselement wird, wie bereits oben erläutert, in Form von Struktur, Metadaten, Informationsinhalt und –kontext (Entstehungs- und Anwendungskontext) beschrieben. Die hierfür benötigten Konzepte werden von den abgebildeten drei Arten von Ontologien zur Verfügung gestellt. Die resultierenden Beschreibungen werden gemäß [LABHS99] in Attributen des Informations- und Wissenselements „gespeichert".

In Anlehnung an [LABHS99] und [Ma01] werden nun die drei Dimensionen der Wissensbeschreibung näher vorgestellt.

Die Informationsontologie stellt stabile und anwendungsunabhängige Konzepte für die Spezifikation der verschiedenen Arten von Informations- und Wissensquellen mit ihrer jeweiligen Struktur, Möglichkeiten des Zugriffs sowie Formateigenschaften bereit. Sie bietet generische Konzepte und Attribute z.B. für den Autor, die Zuverlässigkeit einer Information etc..
Konkrete Instanzen dieser Ontologie, also Beschreibungen spezieller Informations- und Wissensquellen, greifen auf Konzepte der Unternehmens- bzw. Domänenontologien zurück.

Mit Hilfe der Unternehmensontologie wird der Informationskontext beschrieben, welcher in Form von Organisationsstrukturen und Prozessmodellen dargestellt wird. Für die Bestimmung der Relevanz eines Informations- bzw. Wissenselements in Bezug auf eine bestimmte Aufgabe sind sowohl dessen Entstehungs- als auch potentielle Nutzungskontexte von entscheidender Bedeutung. Da die Konzepte der Unternehmensontologie möglichst unabhängig vom jeweiligen Unternehmen sein sollten, wurden verschiedene Standardontologien entwickelt, die von den meisten Unternehmen eingesetzt werden können.

Domänenontologien werden schließlich benötigt, um den Inhalt von Informations- und Wissensquellen zu modellieren. Typischerweise sind die bereitgestellten Konzepte hoch spezifisch für eine bestimmte Anwendung. Jedoch wurden auch in diesem Bereich Anstrengungen unternommen, um wiederverwendbare, aufgabenunabhängige Domänenontologien anzubieten, z.B. für den Bereich der Chemie [van der Vet und Mars, 1993 nach [LABHS99]] und der Materialwissenschaften [van der Vet, 1995 nach [LABHS99]].

Zusammenfassend wird ein konkretes Informationselement mit Hilfe von Konzepten der Informationsontologie beschrieben, wobei diese wiederum auf Konzepte sowohl der Unternehmens- als auch der Domänenontologie zugreift, um die verschiedenen Attribute des Informationselements zu füllen. Die nachfolgende Abbildung der im OMIS von KnowMore verwendeten Arten von Konzepten und deren Beziehungen untereinander soll diesen Zusammenhang abschließend noch einmal verdeutlichen.

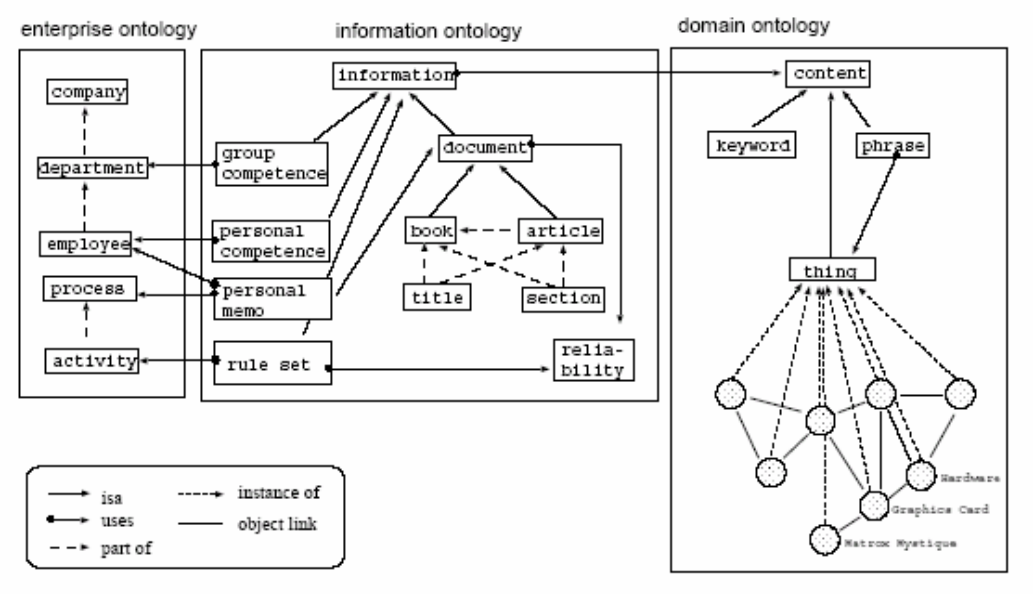


Abbildung 3: Einfache Wissensbeschreibung anhand beispielhafter Ontologien nach [LABHS99]

Hinsichtlich der Repräsentationssprachen für Ontologien zur Wissensbeschreibung konnte bislang noch kein Standard etabliert werden. Die meisten eingesetzten Repräsentationsformate wie Frame-Logic u.a. sind ursprünglich nicht für die Wissensbeschreibung oder das Information Retrieval entwickelt, sondern für diese Einsatzbereiche „zweckentfremdet“ worden. Im Rahmen von KnowMore kommt eine eigens hierfür entwickelte Algebra, die sog. OCRA[44], zum Einsatz, die an dieser Stelle jedoch nicht näher erläutert werden soll. Der interessierte Leser sei auf [LABHS99] verwiesen.

2.4 Durchführung der Suche

Der nachfolgende Abschnitt, der im Gesamtüberblick den Ablauf der Suche und die anschließende Ergebnispräsentation beschreibt, stützt sich auf [Ma01], [ABM02] und [ABMW00]. Auf die zum Einsatz kommende Suchtechnologie und die damit verbundene Suchquelle wird im anschließenden Abschnitt 2.5 detailliert eingegangen.

Wie bereits im einleitenden Abschnitt erläutert, erfolgt die Unterstützung des Benutzers in KnowMore durch eine pro-aktive, kontextsensitive Belieferung mit relevanten Informationen. Der entsprechende Workflow-Teilnehmer wird, ohne eine explizite, detaillierte Anfrage stellen zu müssen, mit Informationen versorgt. Die nachfolgende Abbildung, welche anschließend näher erläutert wird, skizziert die Realisierung dieses Sachverhalts.

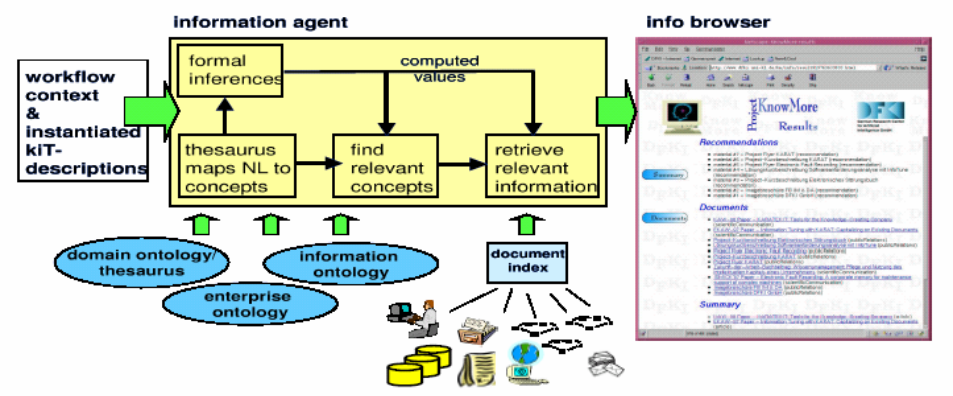


Abbildung 4: Versorgung des Benutzers mit Informationen unter Nutzung von Workflow-Kontext; Quelle: [Ma01]

[44] Object-Centered Relational Algebra

Sobald eine wissensintensive Aktivität innerhalb der Workflow-Ausführung erreicht wird, startet die Workflow Engine[45] neben der eigentlichen Aktivität den in der KIT-Beschreibung angegebenen Informationsagenten als entsprechende Suchmaschine. Der Agent übernimmt nun die Informationsversorgung, indem er die in der KIT-Beschreibung enthaltene generische Anfrage instanziiert, um so relevante Informationen aus den verschiedenen Informations- und Wissensquellen des Unternehmens-gedächtnisses zu apportieren.

Über die Verwendung eines Thesaurus werden im Kontext des Benutzers Konzepte der Domänenontologie identifiziert, die ggf. für eine erweiterte, ontologiebasierte Suche verwendet werden. Zunächst versucht der Informationsagent jedoch solche Informations- und Wissenselemente im Unternehmensgedächtnis aufzuspüren, welche die Suchkonzepte der vom Informationsagenten instanziierten Anfrage in ihren Inhaltsbeschreibungen enthalten. Ist kein derartiges Element verfügbar, so versucht der Agent über die Domänenontologie verwandte Konzepte ausfindig zu machen. Hierzu verfolgt er, ausgehend von den identifizierten Konzepten, gemäß festgelegter ontologiebasierter Suchheuristiken Beziehungskanten in der Domänenontologie. Es kommt also zu einer Anfrageerweiterung. Anhand der zusätzlich identifizierten Konzepte erfolgt wiederum eine Suche in den Inhaltsbeschreibungen der im OMIS gespeicherten Informations- und Wissenselemente. Dabei werden die in den instanziierten KIT-Variablen gehaltenen Kontextinformationen des laufenden Workflows dazu verwendet, die Relevanz gefundener Informationen zu bestimmen.

Die Ergebnisse des Suchprozesses werden dem Nutzer in Form einer Webseite im KnowMore info browser (rechte Seite der nachfolgenden Abbildung) präsentiert. Diese enthält Links zu allen gefundenen Informationen, die wiederum zu Kategorien wie problembezogene Dokumente, best practice reports, Beiträge in Intranet-Diskussionsforen etc. gruppiert sind. Weiterhin werden Referenzen zu vorausgehenden Workflow-Instanzen mit demselben Problem und davon betroffenen Dokumenten und anderen Informationselementen aufgeführt. Ebenfalls referenziert werden die an diesen Workflows beteiligten Nutzer oder aber Nutzer, die aufgrund ihres Profils über entsprechende Erfahrung verfügen.

[45] siehe Glossar

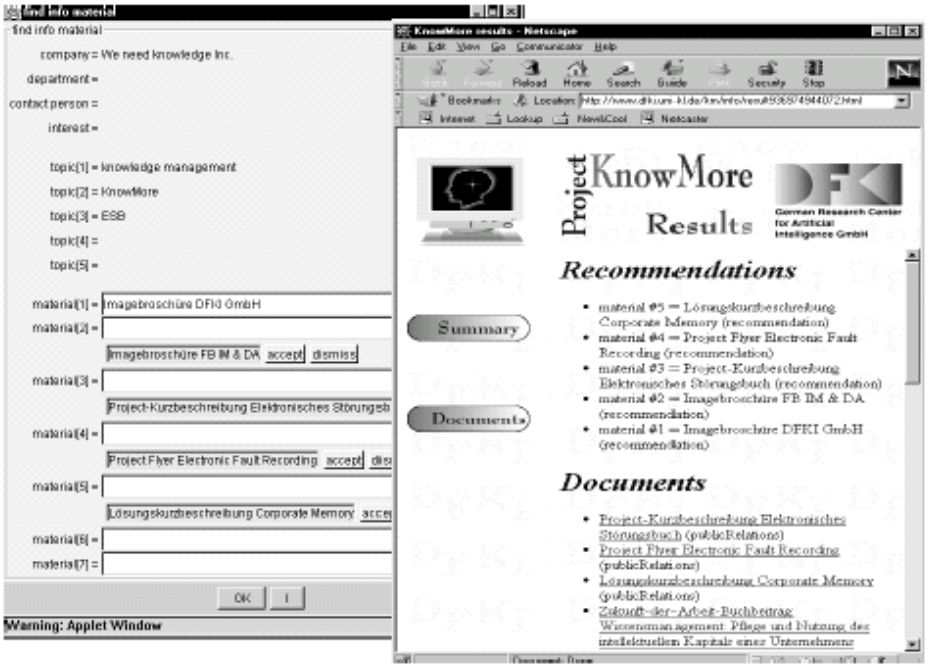


Abbildung 5: Ergebnispräsentation im Projekt KnowMore; Quelle: [ABMW00]

Neben der Präsentation im info browser werden gefundene, relevante Informationselemente zusätzlich in einem Editor-Fenster (linke Seite der Abbildung) hinterlegt. Der jeweilige Nutzer, der pro-aktiv und kontextbasiert mit Informationen - vorerst in Form von Vorschlägen - beliefert wird, kann diese über das Editor-Fenster annehmen bzw. ablehnen oder aber anderes Material gemäß seiner persönlichen Einschätzung auswählen. Letzteres erfolgt durch Eingabe in ggf. mehrere leere material[]-Felder. Folglich erhält der Benutzer im Editor-Fenster die Möglichkeit zum Relevance Feedback.

Unabhängig davon, wie sich der Nutzer entscheidet, verfolgt das System den Zustand des Workflows samt erzeugten möglicherweise zukünftig relevanten Informationselementen und speichert die Beurteilungsergebnisse automatisch zusammen mit der relevanten Kontextinformation ab.
Wird im Rahmen der weiteren Workflow-Ausführung erneut ein Informationsagent aktiv, um weitere potentiell relevante Informationen zur Verfügung zu stellen, so erinnert sich das System an die abgespeicherten Ergebnisse und verändert letztlich die Vorschläge entsprechend. Somit wird der Nutzer fortwährend mit auf seinen Kontext basierenden Informationen versorgt.

2.5 Suche

Quelle für die Suche ist zunächst die im Abschnitt ‚Nutzermodell' eingeführte Menge „karteikartenähnlicher" Wissensbeschreibungen, die den Index klassischer IR-Systeme ersetzt. Dabei enthält jede dieser „Karteikarten" die Gesamtheit der zu dem entsprechenden Informations- bzw. Wissenselement gespeicherten Attribute. Gegebenenfalls wird im Rahmen der Suche zusätzlich auf die Domänenontologie zurück gegriffen.

2.5.1 Verwendete Technologie

Bereits im Abschnitt 2.4 wurden ontologiebasierte Suchheuristiken kurz aufgegriffen. Da diese die im Rahmen von KnowMore zum Einsatz kommende „Suchtechnologie" darstellen, sollen sie im folgenden Abschnitt näher erläutert werden. Die Ausführungen orientieren sich dabei an [LHAS99] und [LABHS99].

Wie bereits im eben aufgeführten Abschnitt ‚Durchführung der Suche' erläutert, kommen ontologiebasierte Suchheuristiken dann zum Einsatz, wenn im Unternehmensgedächtnis keine Informations- und Wissenselemente vorhanden sind, welche die ursprünglichen Suchkonzepte in ihren Inhaltsbeschreibungen enthalten. Über den Einsatz festgelegter Suchheuristiken, welche in generische und domänenspezifische unterschieden werden können, sollen in diesem Fall verwandte Konzepte ausfindig gemacht werden. Hierzu setzt der Informationsagent die Suchheuristiken in intelligente Graphtraversierungsstrategien um, um so über das Verfolgen von Beziehungskanten in der Domänenontologie weitere Suchkonzepte zu identifizieren.

Anhand des nachfolgenden Ausschnitts einer Domänenontologie, der [LABHS99] entnommen ist, wird das Konzept der generischen Suchheuristik vorgestellt. Daran schließt sich eine zunächst theoretische, dann wiederum beispielhaft verdeutlichende, Erläuterung domänenspezifischer Suchheuristiken an.

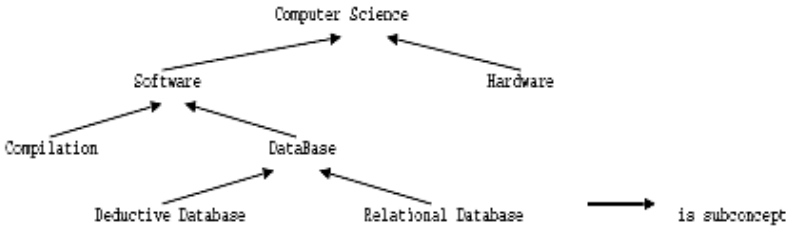


Abbildung 6: Ausschnitt einer Domänenontologie; Quelle: [LABHS99]

Im abgebildeten Sachverhalt sollen Informationen zum Konzept „Database“ geliefert werden. Sind diese nicht verfügbar, so wäre eine generische Suchheuristik, nach Informationen zu „Deductive Database“ oder „Relational Database“ zu suchen. Bei diesen handelt es sich, wie abgebildet, um Subkonzepte des eigentlichen Suchbegriffs. Folglich ist es durchaus möglich, dass über sie generische Informationen zum Suchkonzept „Database“ erschlossen werden können, da bspw. Literatur zu Themenspezialisierungen in der Regel ebenfalls Informationen zum übergeordneten Thema enthält. Kann auch zu den im sublink von „Database“ enthaltenen Konzepten „Deductive Database“ und „Relational Database“ keine Information gefunden werden, so ist entsprechend der uplink von „Database“, hier „Software“, in gleicher Weise zu durchsuchen.

Da in einem konkreten Anwendungsszenario neben diesen generischen Suchheuristiken vielfältige domänenspezifische Suchheuristiken angewandt werden können, geben Liao et al. in [LABHS99] dem Nutzer das Konstrukt der ‚heuristics expression’ an die Hand. Mit Hilfe einer Formelsequenz der nachfolgenden Form können anwendungsspezifische Suchheuristiken angegeben und so die kommentierten Beziehungen (annotated relationships) zwischen den Konzepten der Domänenontologie für eine erweiterte, ontologiebasierte Suche genutzt werden. Bezogen auf den Kontext von KnowMore müsste eine solche Formelsequenz zur Prozessdefinition von einem menschlichen Experten erstellt und im System für alle Informationsagenten zugänglich hinterlegt werden.

$$f_1 \circ f_2 \circ \ldots \circ f_n \quad \text{wobei} \quad f_i \equiv (\lambda)^{\gamma}$$

In einer solchen Formelsequenz steht λ für einen link oder inversen link ($link^{-1}$).
Bei γ handelt es sich um eine sog. ‚partial closure specification’, die ein effizientes Verfolgen einer vorgegebenen Anzahl von links zwischen Konzepten ermöglicht. In der in Abschnitt 2.3 erwähnten OCRA ist dieses Konstrukt als spezieller Anfrageoperator direkt implementiert.
Der Aufbau der ‚heuristics expression’ folgt einer kaskadischen Strategie mit Abarbeitung von rechts nach links. Dies bedeutet konkret, dass diejenigen Heuristiken, die potentiell unsicherere Erkenntnisse liefern und somit weiter links stehen, zuletzt angewandt werden sollen.
Im Normalfall wird die Formelsequenz nur soweit abgearbeitet, bis eine Ergebnisknotenmenge generiert werden kann. Allerdings könnte man dem Benutzer meiner Meinung nach optional eine Anfrageerweiterung anbieten. So könnte dieser schrittweise die Suche ausweiten, sofern ihm der Umfang der bisherigen Ergebnismenge nicht ausreicht. Dabei ist dem Benutzer gegenüber jedoch klar herauszustellen, dass die Zuverlässigkeit der Ergebnisse im Regelfall abnehmen wird.
Als Input erhält eine ‚heuristics expression’ eine Menge von Knoten, nämlich die Suchkonzepte. Die in der Formel angegebenen Traversierungsanweisungen werden nun von jedem dieser Knoten ausgehend von rechts nach links ausgeführt, wobei jeder Schritt eine Zwischenmenge von Knoten als Ausgangspunkt für den nächsten Schritt liefert. Die ‚partial closure specification’ gibt hierbei die Anzahl der zu verfolgenden Kanten des gleichen Typs an.

Zur Veranschaulichung des bislang nur theoretisch vorgestellten Konstrukts der ‚heuristics expression' wird nachfolgend ein Beispiel herangezogen, das [LHAS99] entnommen wurde.

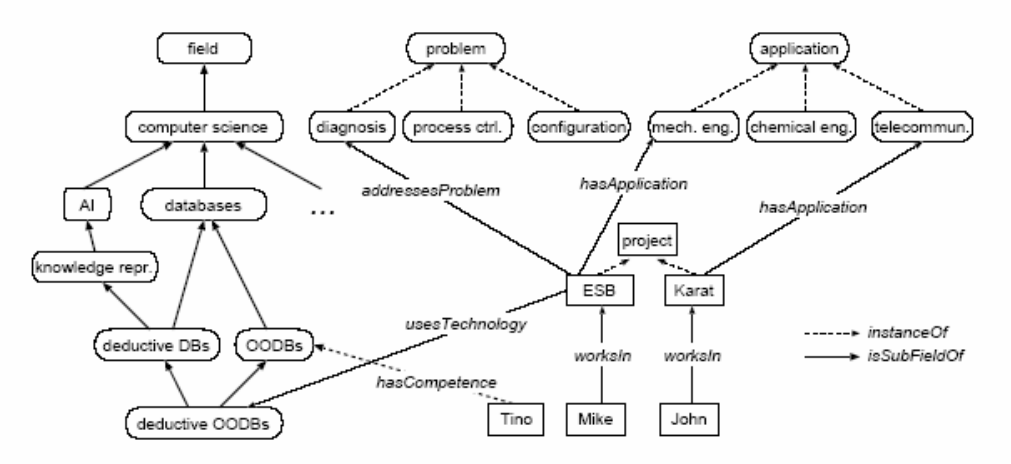


Abbildung 7: Ausschnitt einer erweiterten Ontologie aus dem Bereich Informatik; Quelle: [LHAS99]

Gesucht werden soll in diesem Beispiel nach Experten auf dem Gebiet „deductive OODBs".
Die wohl zuverlässigsten Ergebnisse liefert eine Suche nach Personen, die direkt mit diesem Suchkonzept über eine „hasCompetence"-Kante verbunden sind. In obiger Notation ausgedrückt entspräche dies folgender Formulierung: $(\text{hasCompetence}^{-1})^1$. Diese bietet sich als Einstieg in den Suchprozess an und steht folglich am rechten Ende der Formelsequenz.
Da jedoch keine „hasCompetence"-Kante auf das Konzept „deductive OODBs" verweist, liefert diese Heuristik eine leere Knotenmenge.
Als nächstes sollte daher nach Personen gesucht werden, die in einem Projekt gearbeitet haben, in welchem die gesuchte Technology eingesetzt wurde. Die entsprechende Anweisung lautet $(\text{worksIn}^{-1})^1 \circ (\text{usesTechnology}^{-1})^1$ und stellt in Bezug auf die Abarbeitungsreihenfolge die 2. Teilformel von rechts dar. Im obigen Beispiel gelangt man über das Entlanglaufen der „usesTechnology"-Kante von „deductiveOODBs" aus entgegen der Pfeilrichtung zum ESB-Projekt. Über das Verfolgen der „worksIn"-Kante - wiederum entgegen der Pfeilrichtung – gelangt man zu Mike. Man kann annehmen, dass Mike über Kenntnisse auf dem Gebiet „deductiveOODBs" verfügt, da er in einem Projekt (dem „ESB project") beschäftigt war, in dem die entsprechende Technologie eingesetzt wurde.

Allerdings ist hier schon feststellbar, dass ein MA, der direkt mit einem Fachgebiet über eine „hasCompetence“-Kante verbunden ist, der sicherere Experte auf diesem Gebiet sein wird.

Hätte man auch über diese Teilformel noch keine Ergebnisknotenmenge erlangt, könnte man nach solchen Personen suchen, die in einem unmittelbaren Oberbereich des gesuchten Themas erfahren sind, hier also „deductiveDBs“ bzw. „OODBs“. Die zugehörige Formalisierung lautet: $(\text{hasCompetence}^{-1})^1 \circ (\text{isSubFieldOf})^1$. Auf diesem Wege würde man hier Tino finden, der Kompetenzen im Bereich „OODBs“ vorweisen kann.

Folglich lautet die komplette ‚heuristics expression’ für den beschriebenen Sachverhalt:

$$(\text{hasCompetence}^{-1})^1 \circ (\text{isSubFieldOf})^1 \circ (\text{worksIn}^{-1})^1 \circ (\text{usesTechnology}^{-1})^1 \circ (\text{hasCompetence}^{-1})^1$$

Abschließend soll noch erwähnt werden, dass eine Heuristik-Sprache, wie die hier verwendete, in Szenarien, in denen Anfragen manuell gestellt werden, in die Anfragesprache eingebunden werden sollte. Die Begründung dafür liegt in der Erkenntnis, dass eine Einbettung in die Anfragesprache eine größere Flexibilität gewährleistet als eine unmittelbare Verankerung derartiger Suchheuristiken in der Implementierung der Retrieval-Komponente. Da Anfragen in KnowMore jedoch durch die zuständigen Informationsagenten automatisiert gestellt werden, scheidet diese flexiblere Möglichkeit hier aus.

2.5.2 Suchmaschine

Im Rahmen von KnowMore sind zwei Arten von „Suchmaschinen“ zu unterscheiden: zum einen die in den KIT-Beschreibungen spezifizierten und dann im Rahmen der pro-aktiven Suche eingesetzten Informationsagenten, zum anderen diejenigen Agenten, die die kontextbewusste Speicherung neu generierter Informations- und Wissenselemente übernehmen. Letztere sind deshalb als „Suchmaschine“ einzustufen, weil sie den für die Abspeicherung benötigen Entstehungskontext eines Informations- bzw. Wissens-elements aus den im Abschnitt 2.2 genannten Kontextquellen des Workflows apportieren.
Über die Implementierung beider Agenten kann an dieser Stelle keine Aussage getroffen werden, da hierzu die einschlägige Literatur keine Informationen bereitstellt. Im Hinblick auf den Funktionsumfang der Informationsagenten sei auf Abbildung 4 verwiesen, zugehörige Erläuterungen befinden sich in Abschnitt 2.4. Für das Ziehen formaler Inferenzen werden die im vorhergehenden Abschnitt erläuterten ontologiebasierten Suchheuristiken angewandt.

2.5.3 Betrachtete Features

Bei den von den beiden Arten von Suchmaschinen extrahierten Daten handelt es sich um Metainformationen, die als Attribute zusammen mit den zugehörigen Informations- und Wissenselementen in der dem Unternehmensgedächtnis zugrunde liegenden Datenbank abgelegt sind. Die Gesamtheit der zu einem Informations- bzw. Wissenselement gespeicherten Attribute bildet die bereits zu Beginn erläuterte "karteikartenähnliche" Wissensbeschreibung dieses Elements. Die möglichen Attributwerte werden dabei durch die drei beschriebenen Arten von Ontologien vorgegeben.

Konkret werden folgende features im Rahmen von KnowMore betrachtet und auch verwaltet: Metainformationen sowohl über den Workflow-Kontext als auch über den relevanten Kontext wissensintensiver Aktivitäten; die in den Inhaltsbeschreibungen eines Informations- bzw. Wissenselements verwendeten Konzepte der Domänenontologie und schließlich Merkmale des erzeugten Informations- bzw. Wissenselements selbst. Bei letzteren handelt es sich um Informationen wie den Dateinamen, den Speicherort etc., die vermutlich aus den Dokumenteneigenschaften extrahiert werden oder aber über die für die Erstellung verwendete Applikation zugreifbar sind. Die Metainformationen über den Workflow-Kontext können über im Workflow Management System zugreifbare Workflow-Kontrolldaten wie Workflow-Instanz-ID, Aktivitäts-ID, aktueller Benutzer etc. apportiert werden. Die Metainformationen über den relevanten Kontext wissensintensiver Aktivitäten, innerhalb derer Informations- bzw. Wissenselemente erzeugt wurden, können zum Zeitpunkt des Abspeicherns aus den KIT-Variablen extrahiert werden. Bei den in diesen Variablen gehaltenen Informationen handelt es sich um Instanziierungen der generischen Anfragekriterien der zugehörigen KIT-Beschreibung. Die in der Inhaltsbeschreibung eines Informations- bzw. Wissenselements verwendeten Konzepte der Domänenontologie werden in Form von Verweisen auf eben diese Konzepte indexiert.

Zumindest ein Teil dieser Features – auf jeden Fall die in den Inhaltsbeschreibungen referenzierten Konzepte der Domänenontologie sowie die Merkmale des Informations- bzw. Wissenselements selbst – kann im nachfolgend abgebildeten Editor eingesehen und ggf. geändert werden.

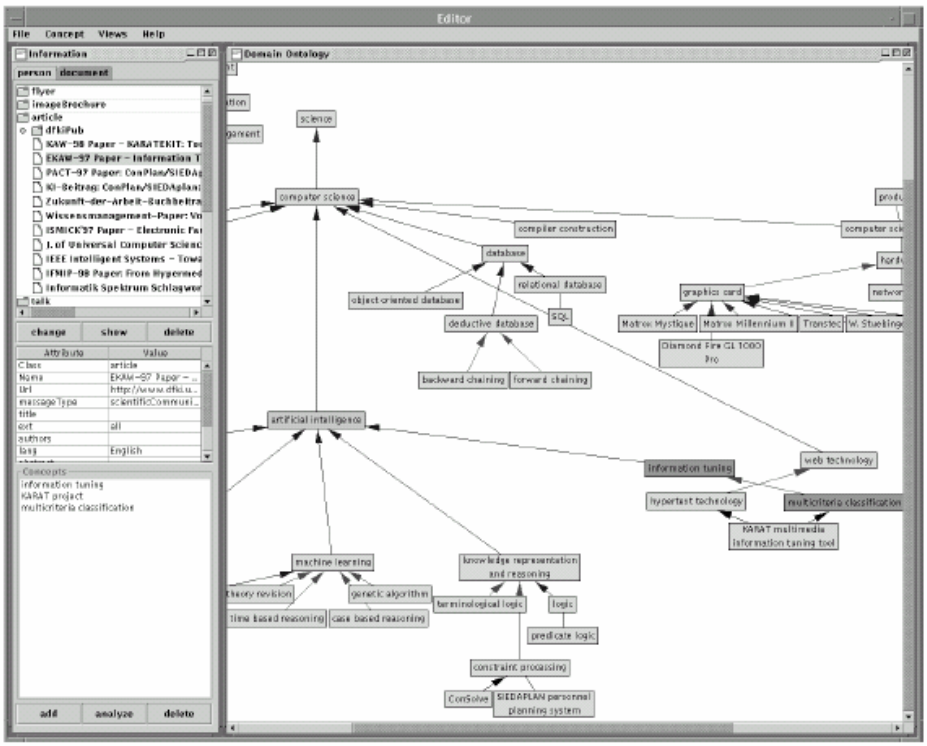


Abbildung 8: Im Projekt KnowMore eingesetzter Editor für die Beschreibung von Dokumenten und Ontologien; Quelle[LABHS99]

2.5.4 Indexierung der erfassten Daten

Bei der Indexierung der erfassten Daten ist meiner Meinung nach neben dem Standardfall der automatischen Indexierung durch den jeweiligen für die kontext-bewusste Speicherung zuständigen Agenten die Indexierung zum Start des KnowMore-Systems gesondert zu betrachten. Hintergrund ist die notwendige Erschließung eines anfänglichen Wissenspools, um Nutzer des Systems von Beginn an pro-aktiv und kontext-sensitiv mit Informationen versorgen zu können.

Die Indexierung zum Systemstart ist meinem Verständnis nach - die einschlägige Literatur hält hierzu keine Informationen bereit – höchstens teilautomatisiert durchführbar. Um in den zu indexierenden Dokumenten Konzepte der Domänenontologie zu identifizieren, könnte ich mir den Einsatz einer in Microsoft VBA implementierten erweiterten Suchfunktion vorstellen. Hierzu würden die zu indexierenden, in einem gemeinsamen Verzeichnis abgelegten Dokumente nacheinander in Microsoft Word geladen und ggf. manuell nachformatiert werden. Die Suchfunktion würde nun sequentiell die in einer gesonderten Datei abgelegten Begriffe der Domänenontologie einlesen und nach diesen im aktuellen Dokument suchen. Am Ende der Durchsuchung eines Dokuments könnten gefundene Begriffe zusammen mit Vorkommenshäufigkeit und –ort dem die Indexierung betreuenden menschlichen Experten präsentiert werden. Dieser entscheidet daraufhin, welche Konzepte der Domänenontologie mit dem entsprechenden Dokument verlinkt werden sollen und nimmt die Verlinkung z.B. in dem in Abbildung 8 abgebildeten Editor vor.

Der Standardfall der automatischen Indexierung durch einen entsprechenden Agenten kommt, wie bereits erläutert, immer dann zur Anwendung, wenn im Rahmen der Ausführung einer wissensintensiven Tätigkeit ein neues Informations- bzw. Wissenselement erzeugt wird, das es festzuhalten lohnt. Hier übernimmt der jeweilige Agent die Extraktion der für die Ablage benötigten Metainformationen. Diese werden aus dem Workflow Management System bzw. aus zur Aufgabenbearbeitung verwendeten Applikationen abgegriffen. Ferner kommen hinsichtlich der Identifizierung der im Informations- bzw. Wissenselement enthaltenen Konzepte der Domänenontologie die im Abschnitt 2.5.1 beschriebenen generischen und domänen-spezifischen Suchheuristiken zum Einsatz.

2.6 Fazit

Das in diesem Abschnitt vorgestellte Projekt KnowMore zielt auf eine Unterstützung wissensintensiver Kernprozesse und der in diesen vorkommenden wissensintensiven Tätigkeiten, den sogenannten KITs, ab. Dabei geht es nicht um eine Optimierung der Prozesse selbst. Statt dessen liegt der Fokus auf der Installation eines zusätzlichen ‚knowledge-service process', der dem jeweiligen Benutzer die Erledigung entsprechender Aufgaben durch relevante Zusatzinformationen erleichtern soll. Da jedoch keinerlei Informationen zur Ergebnisqualität des KnowMore-Systems verfügbar sind, kann nicht beurteilt werden, in welchem Umfang die Zielsetzung dieses Ansatzes im praktischen Einsatz erreicht wird.

3 VirtualOffice (Matthias Raps)

Im Folgenden soll nun ein weiteres Projekt des DFKI vorgestellt werden, in dem ebenfalls Kontextinformation aus WfMS genutzt werden. Es handelt sich dabei um das VirtualOffice Projekt oder – um eine einfache Umschreibung zu geben - ‚Dokumentanalyse meets Workflowmanagement'.

3.1 Motivation und Zielsetzung

Ausgangspunkt für dieses Projekt war die Überlegung, dass bis heute zum einen papierbasierte Dokumente oftmals die dominierenden Medien zur Kommunikation im und mit Unternehmen sind und zum anderen aber WfMS (Workflow Management Systeme) und DMS (Dokumentmanagementsysteme) verstärkt zum Einsatz kommen. Von diesen Systemen werden alle Dokumente und damit deren Inhalte für deren Verarbeitung in elektronischer Form benötigt, die mit einem bestimmten Workflow in Bezug stehen. Die Beseitigung dieses Medienbruches stellte sich bis dato als sehr zeit- und kostenintensiv dar, da alle papierbasierten Dokumente von Mitarbeitern sortiert, verteilt und die Informationen in verschiedene Systeme manuell eingegeben werden müssen. Mit VirtualOffice wird nun versucht, diesen Medienbruch möglichst effizient in den Griff zu bekommen und somit - als Hauptziel - den Posteingang eines Unternehmens zu automatisieren.

Dazu soll eine automatische Analyse von gedruckten und von elektronischen Dokumenten ermöglicht werden, sowie eine gezielte Extraktion relevanter Informationen aus den Dokumenten erfolgen. Schließlich sollen die Dokumente bzw. die daraus gewonnenen Informationen der korrekten Workflow-Instanz zugeordnet werden. Üblicherweise wird dabei die Workflow-Instanz ein bestimmtes Dokument bzw. eine Information daraus angefordert haben.

Um die gesteckten Ziele zu erreichen, wird ein sog. DAU–System (document analysis and understanding) generisch in ein übergeordnete WfMS integriert. Durch diese Integration kann das DAU-System den Workflow-Kontext für seine komplexen Aufgaben nutzen, was zu einer verbesserten Qualität der Analyseergebnisse führt (vgl. [BMHW97]). Der Unterschied zu üblichen DAU–Systemen kann also darin gesehen werden, dass in dem im VirtualOffice genutzten DAU-System das komplette Hintergrundwissen über ein Dokument für die Analyseaufgabe genutzt wird. Dieses Wissen wird unter anderem aus den Geschäftsprozessen eines Unternehmens gewonnen.

Im Folgenden sollen nun zunächst Aufbau und Ablauf des VirtualOffice Szenarios ausführlicher erläutert werden. Als durchgängiges Workflow-Beispiel soll der Beschaffungsprozess eines bestimmten Artikels ‚xyz' in einem Unternehmen dienen.

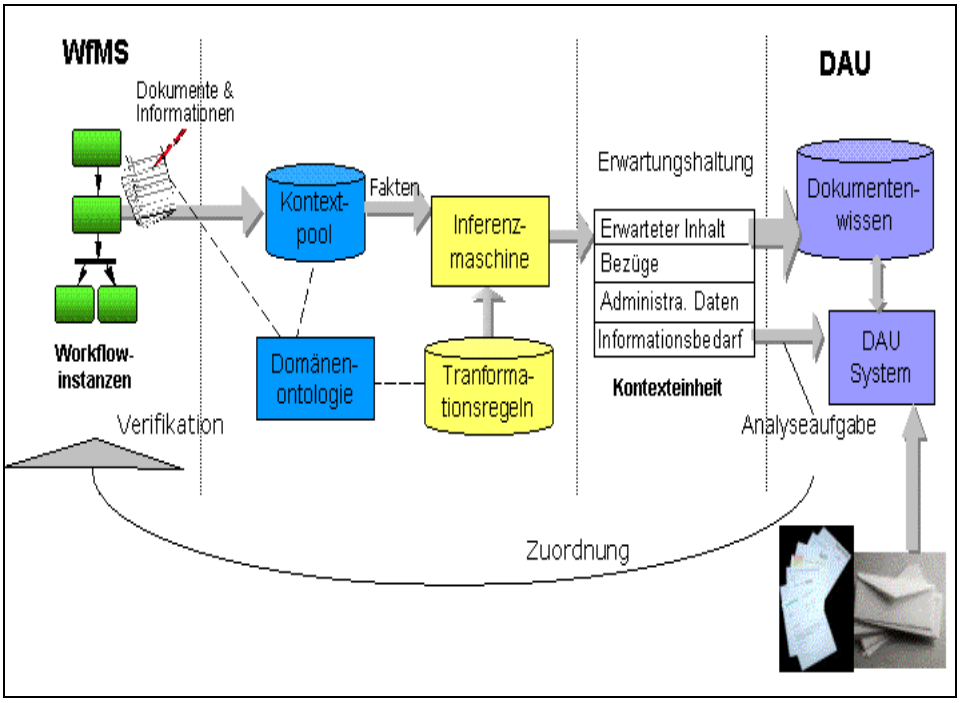


Abbildung 9: Das VirtualOffice Szenario; in Anlehnung an [ABM02]

3.2 Die Kontextgewinnung aus Workflows

Workflow Management Systeme sind zwar als Quelle für Workflow-Kontext durchaus geeignet, aber – wie bereits beim KnowMore Projekt erwähnt - verfügen konventionelle WfMS weder über ein umfassendes Repräsentationskonzept für Workflow–Kontext noch über eine komfortable Zugriffsmöglichkeit auf diesen zur Laufzeit. Aus diesem Grunde wurde im Rahmen des VirtualOffice Projektes der sogenannte Kontext-Pool zur Speicherung von Workflow-Kontext eingeführt. Bevor dieser genauer erläutert wird, soll aber zunächst eine hier geeignete Definition von Workflow-Kontext gegeben werden.

Das DAU benötigt zum einen eine möglichst genaue Beschreibung des erwarteten Dokuments, aber auch Informationen über den ursprünglichen Workflow, der das

Dokument bzw. spezielle Informationen daraus benötigt. Dies ist beispielsweise dessen Workflow-Instanz-ID oder dessen Informationsbedarf über das zu analysierende Dokument. [Ma99] gibt deshalb eine Definition, die Workflow-Kontext aus der Sicht des DAU beschreibt: „Workflow context includes all data related to a document with relevance to DAU and data required for the integration into the WfMS.” Diese

Definition soll vorerst genügen. Eine genauere Erläuterung der einzelnen Workflow-Kontext-Quellen, die VirtualOffice nutzt, erfolgt in 3.6.

Beim Kontext-Pool handelt es sich nun um eine externe Datenbank, in der alle Informationen oder Referenzen auf diese Informationen gespeichert werden, die während der Ausführung des Workflows als nützlich für die späteren Schritte angesehen werden. Im Ablauf des Beschaffungsprozesses können solche Informationen beispielsweise innerhalb der Aktivität Auftragserstellung der Nachrichtentyp ‚Auftrag', der Angestellte ‚Meier', der Empfänger ‚Müller GmbH' und dessen Kundennummer sein. Weitere Kontextinformationen sind die Referenz auf die akt. Workflow-Instanz-ID oder Referenzen auf vorhergehende Korrespondenzdokumente, wie eine Anfrage an die Firma ‚Müller GmbH', ob die gewünschten Produkte geliefert werden können, oder die zugehörige Bestätigung mit einer Preisliste. Damit eine Speicherung dieser Informationen möglich ist, muss diese innerhalb der Workflow-Definition modelliert werden und wird dann z.B. durch eine zusätzliche Applikation während des Workflows ausgeführt. Anstelle einer Hilfsapplikation bieten manche kommerzielle WfMS aber auch Skriptsprachen an, mit denen eine direkte Ausgabe von Workflow-Attributen in Dateien ermöglicht wird.
Für den Fall, dass das zugrundeliegende WfMS es nicht zulässt, dass alle relevanten Informationen zugreifbar sind, wenn diese benötigt werden, ist es außerdem notwendig, spezielle Aktivitäten innerhalb der Workflow-Definition zu modellieren, damit diese Informationen direkt - statt nur der Referenzen - im Kontext-Pool gespeichert werden.
Insgesamt kann man folglich feststellen, dass der Kontext-Pool als ein Archiv dient, welches die Verfügbarkeit von benötigtem Workflow-Kontext garantiert.

3.3 Die Erwartungen und die Kontexteinheit

Tritt nun innerhalb eines Workflows ein Ereignis auf, welches als Antwort ein bestimmtes Dokument oder eine Information daraus benötigt, so muss das DAU darüber in Kenntnis gesetzt werden. Im vorliegenden Beispiel benötigt der aktuelle Beschaffungs-Workflow, nachdem der Auftrag an die Firma Müller GmbH gesendet wurde, eine Auftragsbestätigung von dieser. Diese Erwartung der Workflow-Instanz wird in einer Kontexteinheit festgehalten, die aus folgenden Bestandteilen besteht:

- dem erwartetem Inhalt (content data): Dieser umfasst alle Informationen, die über das erwartete Dokument bekannt sind. In dem gegebenen Beispiel ist das z.B., dass auf eine Auftragsbestätigung der Firma ‚Müller GmbH' über eine bestimmte Menge des Artikels ‚xyz' gewartet wird.
- Bezüge (reference data): Diese beinhalten alle Daten, auf die innerhalb des erwarteten Dokumentes Bezug genommen werden könnte, wie etwa das Auftragsdatum, die Auftragsnummer oder der verantwortliche Mitarbeiter.

Neben der Erwartung enthält eine Kontexteinheit zusätzlich:

- den Informationsbedarf (information need): Dies sind alle Informationen, die eine Workflow-Instanz für ihren weiteren Ablauf benötigt. Dieser Bedarf wird üblicherweise durch eine Liste von benötigten Informationsitems ausgedrückt, die das DAU aus dem zugehörigen Dokument extrahieren soll. Dies kann hier z.B. die Liefermenge, das Lieferdatum etc. sein.
- administrative Daten: Diese werden für die Integration in das WfMS benötigt. Beispiel hierfür ist Workflow-Instanz-ID oder das Ereignis, welches angestoßen werden soll, wenn das erwartete Dokument eingetroffen ist.

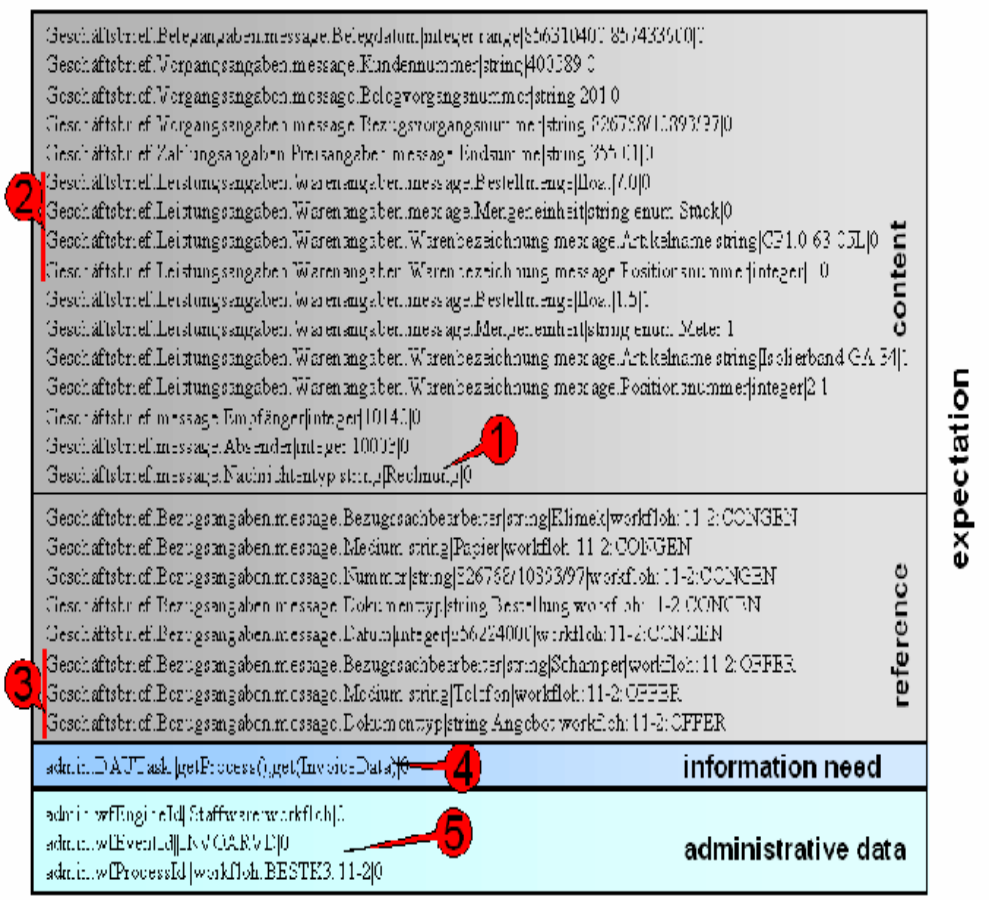


Abbildung 10: eine exemplarische Kontexteinheit; Quelle: [WM01]

Eine exemplarische Kontexteinheit ist in der Abbildung 10 zu sehen. Dort wird eine Rechnung (1) mit einer bestimmten Produktliste (2) erwartet. Referenziert wird auf das

zugehörige Angebot (3). Der Informationsbedarf sind die Rechnungsdaten (4), und als administrative Daten ist das Event ‚INVOARDV' angegeben, das bei Eintreffen des zugehörigen Dokuments ausgelöst werden soll.

Die Kontexteinheit selbst wird durch eine Inferenzmaschine gebildet, welche eine Menge von Transformationsregeln sowie den Kontextpool als Faktenbasis nutzt.

Diese Regeln besitzen zwei Aufgaben. Einerseits werden die Informationen, die in der Domänen-Ontologie des Workflows und dessen Applikationen genutzt wurden, in die DAU-Ontologie transformiert, wo diese durch das Dokumentenwissen repräsentiert werden. Die zweite Aufgabe ist die Ableitung einer Beschreibung des erwarteten Dokuments aus dem zur Verfügung stehenden Workflow-Kontext.

Mögliche Ausprägungen von solchen Produktions- oder Transformationsregeln können in Bezug auf den Beschaffungsprozess sein:

- Der Empfänger des Auftrages wird der Absender der Auftragbestätigung sein.
- Der für den Auftrag verantwortliche Mitarbeiter wird möglicherweise in der Anrede der Auftragsbestätigung zu finden sein.
- Die Auftragsnummer wird möglicherweise im Betreff aufgeführt sein.
- usw.

Die erstellte Kontexteinheit wird in der Erwartungshaltung gespeichert. Diese Erwartungshaltung wird dann mit allen enthaltenen Kontexteinheiten dem DAU-System durch das Dokumentenwissen zur Verfügung gestellt und ist somit der benötigte Workflow-Kontext. Der in der Erwartungshaltung enthaltene Kontext ist somit nach [Ma01] nicht nur eine Ansammlung verfügbarer Informationen, sondern ist vielmehr maßgeschneidert für die Aufgaben des DAU-Systems.

3.4 Nutzung der Kontexteinheiten zur Prozessidentifikation

Trifft nun ein Dokument mit der Eingangspost eines Unternehmens ein, so wird versucht, eine Prozessidentifikation durchzuführen. D.h., es wird versucht, für das Dokument die korrekte zugehörige Workflow-Instanz zu finden. Hierfür nutzt das DAU-System die Erwartungshaltung – und somit alle Kontexteinheiten - um die Analyse und die Zuordnung möglichst effizient durchzuführen.
Sobald eine Übereinstimmung mit einer Erwartung gefunden wurde, werden alle angeforderten Informationen extrahiert und das Dokument mit diesen Informationen an die Workflow-Instanz übergeben. Dort wird dann manuell verifiziert, ob die erfolgte Zuordnung und die Extraktion korrekt sind, und die Bearbeitung des Workflows wird fortgesetzt.

Bei dem bis hier angewandten Workflow-Kontext handelt es sich um dynamischen Kontext, da aktuelle Prozessinstanzen genutzt wurden, um die Informationsrecherche durch das DAU durchführen zu lassen. In VirtualOffice wird neben diesem aber auch

innerhalb von sog. Standarderwartungen statischer Kontext genutzt. Diese dienen für zwei Bereiche der Ausnahmebehandlung. Zum einen kann es sein, dass „nicht erwartete“ Post eintrifft, die von keiner Instanz benötigt wird. Hierfür muss kein Abgleich mit den einzelnen Kontexteinheiten erfolgen. Ein klassisches Beispiel hierfür ist der Eingang von Werbebriefen. Hierfür kann dann beispielsweise ein eigener Workflow gestartet werden. Zusätzlich ist es bei unerwarteter Post auch möglich, bereits befriedigte Erwartungen in die Analyse mit einzubeziehen. Dies ist beispielsweise bei Mahnungen sinnvoll, da hier ein Bezug zu der zugehörigen Rechnung und somit zum entsprechenden Workflow hergestellt werden kann.

Ein weiterer Bereich, der über Standarderwartungen abgebildet wird, ist die Zuordnung von Dokumenten, für die prinzipiell zwar eine Erwartung vorhanden sein müsste (z.B. Rechnungen), aber vom DAU–System trotzdem keine passende Zuordnung durchgeführt werden kann. Auch hierfür gibt es in der Regel spezifische Workflow-Definitionen, die instanziiert werden.

3.5 Das Dokumentenwissen und das DAU-System

Das DAU-System nutzt für seine Analysetätigkeiten das Dokumentenwissen. Dieses zentrale Repository enthält – wie bereits erwähnt- die Erwartungen der einzelnen Workflow-Instanzen. Außerdem ist dort generisches Dokumentenwissen abgelegt, d.h. beispielsweise der prinzipielle Aufbau einer Rechnung oder eines Geschäftsbriefes. Schließlich wird das Dokumentenwissen dazu genutzt, die Analyseergebnisse zu speichern. Um eine einheitliche Repräsentation innerhalb des Dokumentenwissens zu erreichen, wurde für den Prototypen des VirtualOffice Ansatzes eine frame-basierte Notation gewählt. Dort repräsentiert ein Frame ein Dokument oder einen Teil eines Dokuments. Die einzelnen Frames stehen dabei durch Spezialisierungen (is-a) und Aggregationen (is-part-of) zueinander in Beziehung. Für genauere Informationen über die Repräsentation des Dokumentenwissens sei auf [WM01] verwiesen.

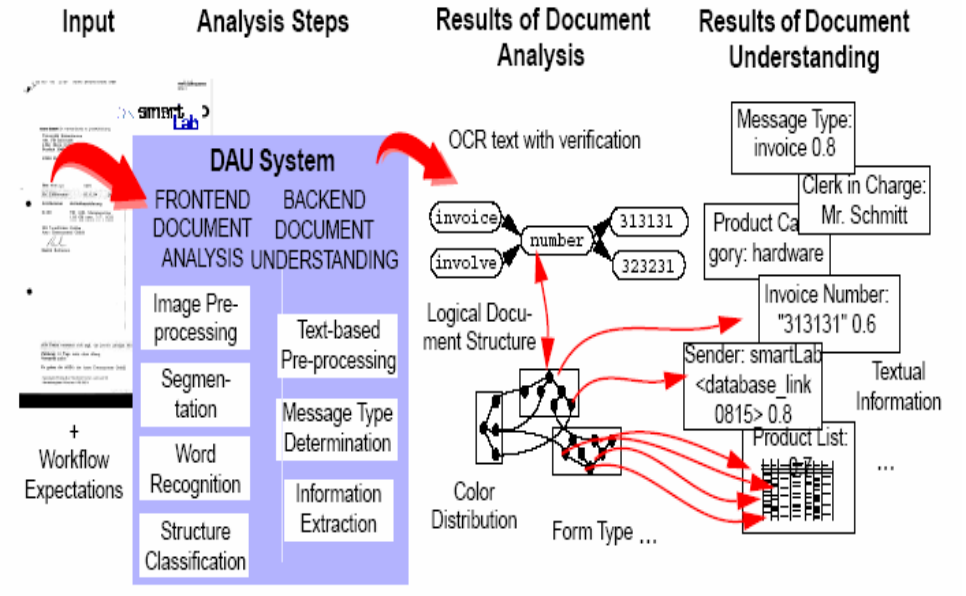


Abbildung 11: Schritte und Ergebnisse eines DAU-Systems; Quelle: [ABMW00]

Das DAU–System selbst besteht aus den beiden Hauptschritten der Dokumentenanalyse (Frontend) und - darauf aufbauend - des Dokumentenverstehens (Backend) (vgl. auch Abbildung 11). Der Input für die Dokumentenanalyse sind eingescannte Dokumente; das Ergebnis ist eine einheitliche Repräsentation der Informationen zusammen mit beschreibenden Attributen. Die Dokumentenanalyse besteht dabei aus verschiedenen Komponenten. Üblicherweise wird nach einer ersten einfachen Bildverarbeitung das Dokument in geometrisch zusammengehörige Bestandteile zerlegt, um dann Segmente von einzelnen Buchstaben, Wörtern, Zeilen und Blöcken aufzufinden. Eine weitere Komponente der Dokumentenanalyse ist die Optical Character Recognition (OCR). Hier erfolgt die Transformation des erkannten Dokumentenbildes in weiterverarbeitbaren ASCII-Text. Des weiteren erfolgt die sog. Strukturklassifikation. Hier werden spezielle Dokumentteile erkannt, wie etwa die Absenderadresse, der Betreff oder andere markante Merkmale.

Die aus der Dokumentenanalyse gewonnen Informationen werden dann als Input für das Dokumentenverstehen genutzt. Hier wird versucht, den Inhalt des Dokuments korrekt zu erschließen. Eine mögliche Komponente hierfür ist die Dokumentklassifikation, die beispielsweise den Nachrichtentyp wie Rechnung, Auftrag oder Mahnung erkennt. Auf diesem Wissen aufsetzend kann dann eine tiefergehende Extraktion von Information erfolgen. Beispielsweise können dann unter Zuhilfenahme des generischen Dokumentwissens über den prinzipiellen Aufbau einer Rechnung der referenzierte Auftrag, das Datum, die einzelnen Positionen, der Rechnungsbetrag usw. gewonnen werden.

Neben generischem Dokumentwissen trägt auch der Workflow-Kontext, in Form der Erwartungen, zu einer Verbesserung von Recall und Precision des DAU bei. Dies soll anhand einiger Beispiele verdeutlicht werden.
So kann beispielsweise die Komponente, die für die Logoerkennung zuständig ist, ihre Suche auf diejenigen Firmenlogos beschränken, die im Kontext erwähnt wurden. Pattern Matching Komponenten erkennen Ausdrücke und Formulierungen, die typisch für einen speziellen Nachrichtentyp sind. Ist z.B. auch der Mitarbeitername bekannt, der üblicherweise in dem bestimmten Nachrichtentyp vorkommt, kann die Precision weiter gesteigert werden .

Bei der Umsetzung des DAU–Ansatzes ist zusätzlich zu den einzelnen Komponenten des DAU–Systems eine zentrale Steuerung notwendig. Diese übernimmt und koordiniert die Kommunikation mit dem Posteingangsserver, dem WfMS und den einzelnen Komponenten. Für die Analyse existieren eine Reihe von Strategien, die festlegen, wann was durch welche Komponente analysiert werden soll. Schließlich werden durch einen Ergebnisgenerator die einzelnen Hypothesen der Komponenten zu einem Ergebnis zusammenfasst, welches dann an das WfMS übergeben wird. Die komplette Architektur des Prototypen zu VirtualOffice ist in Abbildung 12 ersichtlich.

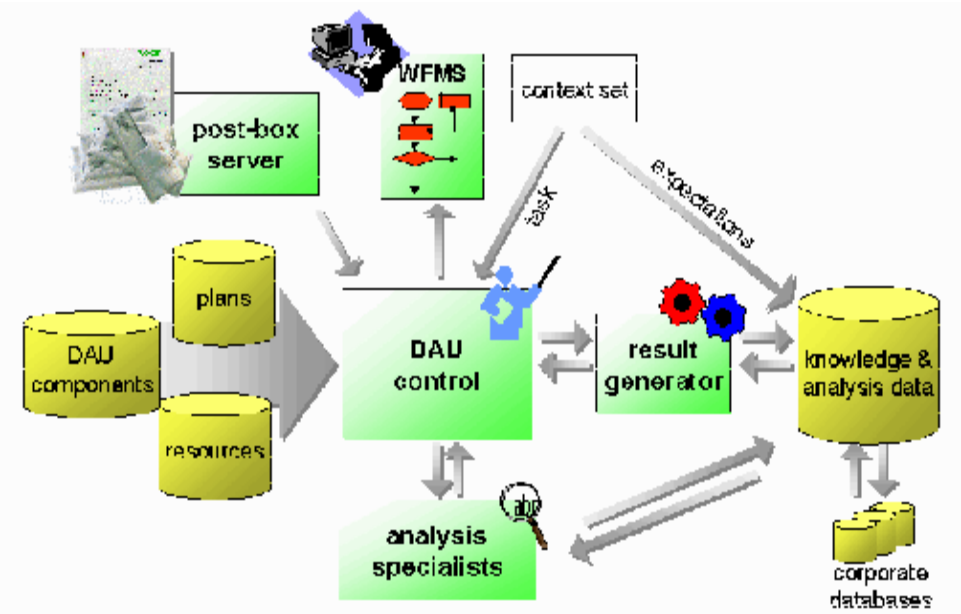


Abbildung 12: Die Systemarchitektur des VirtualOffice Prototypen; Quelle: [WM01]

3.6. Der genutzte Workflow-Kontext in VirtualOffice

Wie bei der Definition von Workflow-Kontext bereits angesprochen, nutzt VirtualOffice verschiedene Arten von Quellen für die Erstellung der Kontexteinheiten. Diese sollen hier nochmals kurz in Anlehnung an [Ma01] zusammengefasst und erläutert werden.

Die WfMC[46] unterscheidet folgende drei Arten von Daten, die von VirtualOffice genutzt werden:

- Workflow-Kontrolldaten (workflow control data): interne Daten, die durch das WfMS verwaltet werden und üblicherweise für Anwendungen nicht zugänglich sind; beispielsweise die Workflow-Instanz-ID bzw. Aktivitäts-ID. Diese wird benötigt, um eine spezielle Kontexteinheit mit der diese erzeugenden Instanz bzw. Aktivität zu verknüpfen.
- Workflow-relevante Daten (workflow relevant data): Daten, die vom WfMS genutzt werden, um Zustandsübergänge, Übergangsbedingungen und die Zuordung von Workflows zu Bearbeitern festzulegen. Diese werden bei VirtualOffice zur Befüllung des Kontextpools genutzt.
- Anwendungsdaten (application data): Daten, die spezifisch von bestimmten Anwendungen genutzt werden und nicht zugänglich für das WfMS sind. Diese Daten werden ebenfalls im Kontextpool gespeichert. Beispiel: Die in ein Textverarbeitungsprogramm eingegebene Adresse des Empfängers.

Zusätzlich wird Workflow-Kontext aus folgenden Quellen gewonnen:

- Workflow-Definition (workflow definition oder auch process model): Spezifiziert die einzelnen Bestandteile eines Workflows. Das Workflow engine führt anhand dieser Spezifikation die Workflow-Instanz aus. Die Workflow-Definition legt also fest, was wann, wie, von wem und mit welchen Daten durchgeführt wird. Bei VirtualOffice wird beispielsweise der Name der Workflow-Definition benötigt, die dann auszuführen ist, wenn eine Standarderwartung zum Tragen kommt. Außerdem wird der Name von den Aktivitäten benötigt, die nach einer erfolgreichen Zuordnung auszuführen sind.
- Organisationsmodell (organizational model): Dies repräsentiert die Struktur einer Organisation, aus der hervorgeht, wer für welche Aufgabe zuständig ist. Dieses Modell wird bei VirtualOffice dazu genutzt, nutzerspezifische Informationen zu gewinnen (beispielsweise die Telefonnummer des verantwortlichen Mitarbeiters, auf die in der Eingangspost referenziert werden könnte). Außerdem können Standardkontexteinheiten dazu genutzt werden, Dokumente direkt Nutzern, Rollen oder Abteilungen zuzuweisen, anstatt eine eigene Workflow-Definition zu instanziieren.

[46] WfMC: Workflow Management Coalition; www.wfmc.org

- Informationsbedarf (information need): Auch der Informationsbedarf einer Aktivität wird als Workflow-Kontext angesehen und wie bereits beschrieben als eine Liste von benötigten Informationsitems dargestellt, die mit der Domänenontologie korrespondieren.

3.7 Die Ergebnisqualität von VirtualOffice

Für den VirtualOffice Prototypen wurden experimentell die in Abbildung 13 dargestellten Ergebnisse ermittelt. Das System als Ganzes wurde getestet mit 18 Erwartungen und 16 Informationsitems, wie Absender, Nachrichtentyp, Rechnungssumme etc.. Bei einer Simulation der Originalprozesse wurde dabei festgestellt, das ca. 10 Informationsitems pro Prozess erscheinen und somit in den Erwartungen wiederzufinden sind. Auf der Seite der Eingangspost wurde das System mit 12 Dokumenten überprüft, die durchschnittlich 6 der relevanten Informationsitems enthielten. In der Ergebnistabelle ist jeweils ein Lauf mit Erwartungen einem Lauf ohne Erwartungen gegenüber gestellt.
Es ist ersichtlich, dass sowohl recall als auch precision der einzelnen DAU-Komponenten sich bei der Hinzuziehung von Erwartungen verbessern und vor allem die entscheidende Genauigkeit der Identifikation des korrekten Prozesses bei 100 % liegt.

	Logo-erkennung *(precision)*	Logo-erkennung *(recall)*	Nachrichten-typ *(precision)*	Nachrichten-typ *(recall)*	Referenzen *(precision)*	Referenzen *(recall)*	Genauigkeit der Prozess-identifikation
Mit Erwar-tungen	100%	66,6%	90%	75%	94%	80%	100%
Ohne Erwar-tungen	100%	58,3%	80%	66,6%	73,3%	55%	66,6%

Abbildung 13: Ergebnisse der Prozessidentifikation und einiger Informationsitems; in Anlehnung an [WM01]

Allerdings sind diese Ergebnisse aus meiner Sicht etwas mit Vorsicht zu genießen, da die Anzahl der Dokumente lediglich bei 12 lag und somit kaum von einer realistischen, praxisnahen Simulation gesprochen werden kann.

3.8. Mögliche Effizienzsteigerungen durch den VirtualOffice Ansatz

Zum Abschluss der Vorstellung von VirtualOffice sollen in Anlehnung an [ABM02] mögliche Effizienzsteigerungen, die sich durch den Einsatz von VirtualOffice ergeben, zusammengefasst und zum Teil kurz erläutert werden.

Auf der Seite des Workflow-Management-Systems kann man folgende Vorteile erzielen:

- Die Automatisierung der Posteingangsschnittstelle eines Unternehmens. Es ist nicht mehr notwendig, die gesamte Eingangspost von Mitarbeitern manuell zuordnen und bearbeiten zu lassen. Man hat somit

- einen verbesserten Übergang zwischen papierbasierter und elektronischer Bearbeitung, da eine manuelle Eingabe der Dokumente und der darin enthaltenen Informationen nicht mehr notwendig ist.

- Die Zuordnung erfolgt zu der zugehörigen Workflow-Instanz, statt dass nur eine Zuordnung zu einem Mitarbeiter oder einer Abteilung erfolgt.

- Es ergibt sich eine z.T. erhebliche Reduzierung von Transport- und Liegezeiten und somit die Möglichkeit einer zeitnahen Weiterverarbeitung der Eingangspost.

- Insgesamt ist also eine deutliche Reduzierung des Arbeitsaufwandes und somit der entstehenden Kosten gegenüber der manuellen Zuordnung von Dokumenten vorhanden und

- man besitzt dadurch jetzt ein Kontext-Konzept aus dem Bereich des Workflow-Managements.

Neben diesen Vorteilen, die aus der Sicht des WfMS beschrieben sind, können auch Vorteile auf der Seite des DAU-Systems festgestellt werden:

- Die Analyse der Dokumente erfolgt jetzt im Unternehmenskontext und nicht nur statisch auf Basis allgemeiner Vorgaben.

- Die Laufzeiten des DAU-Systems verkürzen sich durch die mögliche Suchraumreduktion. Beispielsweise kann das DAU-System seine Aufgabe beenden, wenn der komplette Informationsbedarf einer Kontexteinheit extrahiert wurde. Es sind dann keine weiteren Analyseschritte notwendig. Somit erfolgt also eine

- zielgerichtete Informationsextraktion.

- Es kann der Einsatz verschiedenster Analysestrategien erfolgen, abhängig davon, welche Erwartungen momentan in der Erwartungshaltung vorhanden sind.

- Insgesamt führt dies zu einer erhöhten Analysequalität und einer besseren Trefferquote des Gesamtsystems.

3.9 Fazit

Betrachtet man die dargestellten Effizienzsteigerungen, die durch den VirtualOffice Ansatz zu erreichen sind, so erscheint das System für den Praxiseinsatz durchaus interessant zu sein. Allerdings stellt sich die Frage, ob sich die erzielten Ergebnisverbesserungen auch dort in dieser deutlichen Form herauskristallisieren. Ein weiterer Punkt ist die durchaus komplexe Systemstruktur, was die Frage nach Implementierungs-, Wartungs- und Betriebskosten aufkommen lässt, die für Unternehmen von entscheidender Bedeutung sind. Nichtsdestotrotz bleibt die Feststellung, dass man es mit dem VirtualOffice Ansatz durchaus geschafft hat, einen Teil der Lücke zwischen papierbasierten Teilen der Kommunikation und automatisierten Geschäftsprozessen zu schließen.

4 Schlussbemerkung

Die vorgestellten Projekte KnowMore und VirtualOffice setzen auf am Markt verfügbare Workflow-Managementsysteme auf. Da derartige Systeme jedoch nicht über ein umfassendes Konzept für Workflow-Kontext verfügen, mussten in beiden Projekten eigene Lösungen für das Retrieval der gewünschten Kontextinformationen konzipiert werden. Daher hat es sich das DFKI u.a. zum Ziel gesetzt, ein optimiertes Workflow-Managementsystem zu entwickeln, welches derartige Zusatzlösungen überflüssig macht.

Auch wenn existierende Systeme mit oben genanntem ‚Defizit' behaftet sind, wird dies kein größeres Hemmnis für ihre weitere Nutzung sein. In Zeiten, in denen insbesondere mittelgroße und große Unternehmen ihre Ressourcen zunehmend über den gesamten Globus verteilen, ist die effektive Unterstützung verteilten Arbeitens von entscheidender Bedeutung.

Literaturverzeichnis

[ABHKS98] Abecker, A.; Bernardi, A.; Hinkelmann, K.; Kühn, O.; Sintek, M.: Toward a Technology for Organizational Memories. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 1998

[ABM02] Abecker, A.; Bernardi, A.; Maus, H.: Potenziale der Geschäftsprozessorientierung für das Unternehmensgedächtnis. In (Abecker, A.; Hinkelmann, K.; Maus, H.; Müller, H.J., Hrsg.): Geschäftsprozessorientiertes Wissensmanagement – Effektive Wissensnutzung bei der Planung und Umsetzung von Geschäftsprozessen. Springer-Verlag, Berlin, Heidelberg, 2002; S. 215-248

[ABMW00] Abecker, A.; Bernardi, A.; Maus, H.; Wenzel, C: Information support for knowledge-intensive business processes – combining workflows with document analysis and information retrieval. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 2000

[BMHW97] Baumann, S.; Marlburg, M.; auf´m Hofe, H. M.; Wenzel, C.: From paper to a corporate memory: A first step. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 1997

[HKT02] Hinkelmann, K.; Karagiannis, D.; Telesko, R.: PROMOTE – Methodologie und Werkzeug für geschäftsprozessorientiertes Wissensmanagement. In (Abecker, A.; Hinkelmann, K.; Maus, H.; Müller, H.J., Hrsg.): Geschäftsprozessorientiertes Wissensmanagement – Effektive Wissensnutzung bei der Planung und Umsetzung von Geschäftsprozessen. Springer-Verlag, Berlin, Heidelberg, 2002; S. 65-90

[LABHS99] Liao, M.; Abecker, A.; Bernardi, A.; Hinkelmann, K.; Sintek, M.: Ontologies for Knowledge Retrieval in Organizational Memories. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 1999

[LHAS99] Liao, M.; Hinkelmann, K.; Abecker, A.; Sintek, M.: A Competence Knowledge Base System as Part of the Organizational Memory. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 1999

[Ma99] Maus, H.: Towards a Functional Integration of Document Analysis and Understanding in Workflow Management Systems. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 1999

[Ma01] Maus, H.; Workflow Context as a Means for Intelligent Information Support. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 2001

[Oe01] Oestereich, B.: Objektorientierte Softwareentwicklung – Analyse und Design mit der UML. Verlag R. Oldenbourg, München, 2001

[WfMC99] Workflow Management Coalition: Terminology and Glossary. WFMC-TC-1011, Version 3.0, 1999

[WM01] Wenzel, C.; Maus, H.: Leveraging corporate context within knowledge-based document analysis and understanding. German Research Center for Artificial Intelligence (DFKI) GmbH Kaiserslautern, 2001

Glossar (Daniela Neundorf)

Geschäftsprozess: „Zusammenfassung von organisatorisch eventuell verteilten, fachlich jedoch zusammenhängenden Aktivitäten, die notwendig sind, um einen Geschäftsvorfall [...] ergebnisorientiert zu bearbeiten. Die Aktivitäten eines Geschäftsprozesses stehen gewöhnlich in zeitlichen und logischen Abhängigkeiten zueinander.“ [Oe01], S.325

Knowledge Intensive Tasks (KIT): „[...] Prozessaktivitäten, die auf Kernkompetenzen basieren und dem Bearbeiter ein hohes Maß an Gestaltungsmöglichkeiten und Entscheidungsspielräumen geben.“ [HKT02], S. 69

Workflow: Computergestützte Automatisierung eines Geschäftsprozesses oder eines Teiles davon. Während des automatisierten Ablaufs werden Dokumente, Informationen und Aufgaben gemäß einer Reihe von prozeduralen Regeln zwischen den einzelnen Workflow – Teilnehmern zur Bearbeitung ausgetauscht. [WfMC99], S.7

Workflow Engine: stellt Betriebsfunktionen bereit, um die Ausführung von Geschäftsprozess(instanzen) basierend auf den Prozessdefinitionen zu unterstützen. [WfMC99], S.50

Workflow Management System (WfMS): System, welches die Ausführung von Workflows unter Zuhilfenahme von Software definiert, anstößt und koordiniert. Das System ist in der Lage, Prozessdefinitionen zu speichern und zu interpretieren, mit Workflow - Teilnehmern zu interagieren und, wo erforderlich, die Nutzung von IT Werkzeugen und Applikationen anzustoßen. [WfMC99], S. 8

Integration von Kontextbasiertem Information Retrieval in Portalsysteme

Antje Konrad, Lutz Lindner
antje.konrad@stud.uni-bamberg.de
lutz.lindner@stud.uni-bamberg.de

Abstract: Portalsysteme bieten eine Lösung zu verschiedenen Problemen, die mit der stetig steigenden Flut an Informationen, die auf Unternehmen einwirken, zusammenhängen. Zum einen bieten sie eine strukturelle Unterstützung des gesamten unternehmerischen Workflows, da Informationen und Arbeitsprozesse auf einer gemeinsamen Oberfläche integriert werden können und zum anderen bieten Portalsysteme zusätzlich die Möglichkeit weitere Funktionalität einzubinden. Besonders im Hinblick auf die Suche nach, in Netzen verfügbaren, Informationen eröffnen Portalsysteme vielfältige Chancen. Ziel des präsentierten Ansatzes zur Integration von Kontextbasiertem Information Retrieval ist die Unterstützung des Benutzers bei der Suche von relevanten Informationen. Dabei wird versucht, durch Kommunikation des Benutzerkontexts, der sich durch die Interaktion mit dem System durch den Benutzer ergibt, proaktiv auf den potentiellen Informationswunsch des Nutzers zu reagieren. Um diese Kommunikation zu erreichen wird, aufgrund der Heterogenität der Ressourcen, eine gemeinsame „Sprache“ benötigt. Der hier vorgestellte Ansatz liefert hierzu Lösungen und beschreibt anhand eines Prototyps in welcher Weise dieses Konzept realisierbar ist.

1 Motivation

Unsere heutige Gesellschaft entwickelt sich immer mehr zu einer Dienstleistungsgesellschaft bei der Wissen der Garant für den unternehmerischen Erfolg ist. Durch die rasanten Veränderungen, besonders im Bereich der Informations- und Telekommunikationstechnik, ist man heutzutage in der Lage, Wissen raumübergreifend über das Internet verfügbar zu machen. Jedoch zeigt die Tatsache, dass heute verfügbare Wissensbasen wie z.B. das Internet „chaotisch“ (un-)strukturiert sind, dass meistens nicht die Existenz der gesuchten Informationen das Problem darstellt, sondern die Suche danach. Durch die ständige Erweiterung des Internet (täglich kommen ca. eine Mio. neue Internetseiten dazu) entsteht eine wahre Flut an Informationen, der nicht effizient entgegen gewirkt werden kann. Erschwerend kommt hinzu, dass die Informationsquellen sehr verschieden sein können.

So genannte Portalsysteme sind der Versuch, dem Informationssuchenden eine personalisierte Umgebung zur Verfügung zu stellen, um verschiedene Informationsquellen zu integrieren und gemeinsam für den Nutzer zu erschließen. Die verschiedenen Informationsquellen werden in so genannten Portlets dargestellt. Im zweiten Kapitel dieser Ausarbeitung wird näher auf diese Portalsysteme, deren grundsätzlichen Bedeutung sowie deren Aufbau eingegangen. Exemplarisch ist hier das Open-Source-Portalsystem *Jetspeed* herausgegriffen um näher auf die Besonderheiten dieser Systeme eingehen zu können.

Die Stärke der Portalsysteme liegt in der Integration unterschiedlicher Informationsquellen. Doch im Hinblick auf die Suche nach relevanten Informationen bieten sie alleine kaum eine Verbesserung. Dieses Defizit soll mit Hilfe des klassischen Information Retrieval gelöst werden. Ziel ist es, innerhalb des Portalsystems, unter den verschiedenen Portlets (Informationsquellen), Kontextinformation zu kommunizieren, um auf das Informationsbedürfnis des Nutzers reagieren zu können und die Suche nach relevanter Information zu unterstützen. Diese Kontextinformation generiert sich aus der Interaktion des Nutzers innerhalb der einzelnen Portlets. Als Hauptproblem dieser Kommunikation innerhalb des Portalsystems stellt sich die Vielfältigkeit der Informationsarten heraus, die in dem Portalsystem integriert werden können. Das heißt, egal was für Informationen (z.B. OLAP-Abfragen, SQL-Abfragen, Ergebnisse von Internetsuchmaschinen) - der Kontext muss kommunizierbar und für alle anderen Portlets „verständlich“ sein. Ein Lösungsansatz bietet das *Resource Description Framework (RDF)* im Zusammenspiel mit der Theorie von *Ontologien* und des Semantic Web. Diese Konzepte ermöglichen die Definition eines gemeinsamen „Sprach“-Standards, um die Kommunikation innerhalb des Portalsystems zu ermöglichen. Hierauf wird speziell im dritten Kapitel eingegangen.

Hauptteil der Ausführungen wird im Kapitel 4 aber der Ansatz, zur Integration von kontextbasiertem Information Retrieval in Portalsysteme, von Priebe und Pernul sein. Dieser Ansatz verbindet das Konzept der Portalsysteme mit dem konventionellen Information Retrieval, unter Verwendung von RDF/Ontologien. Priebe/Pernul zeigen mit ihrem Prototyp wie die Integration von kontextbasiertem Information Retrieval umgesetzt werden kann. Zuerst wird aber das theoretische Fundament, auf dem das Konzept von Priebe/Pernul aufbaut, näher betrachtet, um danach speziell auf das verwendete Information Retrieval Modell eingehen zu können. Schließlich werden der, bis dato, verfügbare Prototyp und die Forschungsergebnisse analysiert und einer Evaluation unterworfen.

2. Portalsysteme

Dieser Abschnitt soll eine kurze Einführung in das Konzept der Portalsysteme geben. Zunächst wird der Portalbegriff definiert, bevor dann die Möglichkeiten der Klassifizierung von Portalsystemen beschrieben werden. Im Anschluss wird auf die Struktur und den Aufbau dieser speziellen Systeme eingegangen und ein detaillierter Einblick in die Funktionsweise gegeben. Zum besseren Verständnis wird das Open-Source-Portalsystem *Jetspeed* als Beispiel für eine Realisierung herangezogen.

2.1 Portal-Begriff

Zu allererst stellt sich die Frage, was Portal eigentlich bedeutet. Das Wort Portal kommt ursprünglich aus dem lateinischen und bedeutet wörtlich übersetzt soviel wie „Tor zum Palast“. In Analogie dazu beschreibt ein Portalsystem einen Einstiegspunkt zu mehreren verschiedenartigen Informationen bzw. Informationsquellen.

Portalsysteme bieten in diesem Rahmen eine Umgebung zur Integration der verfügbaren Informationsquellen und stellen sie dem Nutzer personalisiert zur Verfügung. Das bedeutet, dass jeder Nutzer des Portalsystems die Oberfläche auf seine Anforderungen hin strukturell und inhaltlich gestalten kann. Dadurch kann ein Portalsystem für verschiedenste Nutzergruppen, mit den unterschiedlichsten Interessen, eingesetzt werden.

Die unterschiedlichen Informationsquellen werden in so genannten Portlets dargestellt. Diese Portlets verhalten sich ähnlich wie Fenster innerhalb des X-Window-Desktops. Sie bieten dem Nutzer Funktionalität im Hinblick auf die Darstellung des Portlets (z.B. Größe des Fensters) sowie im Hinblick auf den Inhalt. Als Portlet-Content kann man sich grundsätzlich jegliche Art von Informationen (z.B. Hypertextdokumente), Web-Services und Anwendungen vorstellen.

Der Hauptvorteil von Portalsystemen liegt in der Tatsache, dass sie web-fähig sind. Durch die Plattformunabhängigkeit der zugrunde liegenden Web-Infrastruktur, ermöglichen Portalsysteme dem Nutzer die Möglichkeit von jedem Ort aus, auf das Portalsystem und somit auf seine gewohnte (Arbeits-) Umgebung zuzugreifen. Besonders im Bereich B2E (business-to-employee) ist dieser Aspekt von großem Nutzen. [JR01]

2.2 Klassifikation von Portalsystemen

Mittlerweile gibt es viele verschiedene Portalseiten im Internet und in Unternehmen, die ihre Dienste bereitstellen. Unterscheidungsmerkmale sind die, vom Portal bereitgestellten Dienste sowie Art und der Umfang des Informationsangebotes oder aber auch der Nutzerkreis. Das zentrale Klassifikationsmerkmal von Portalen ist aber die inhaltliche Ausrichtung des Portals.

Heutige Portalsysteme lassen sich am besten in zwei unterschiedlichen Formen klassifizieren. Zum einen unterscheidet man zwischen vertikalen und horizontalen Portalsystemen, und zum anderen gemäß deren Nutzerkreis in Unternehmensweite Portale (Enterprise Portals). Dabei In der folgenden Abbildung 1 ist die erste Unterscheidung bildhaft dargestellt.

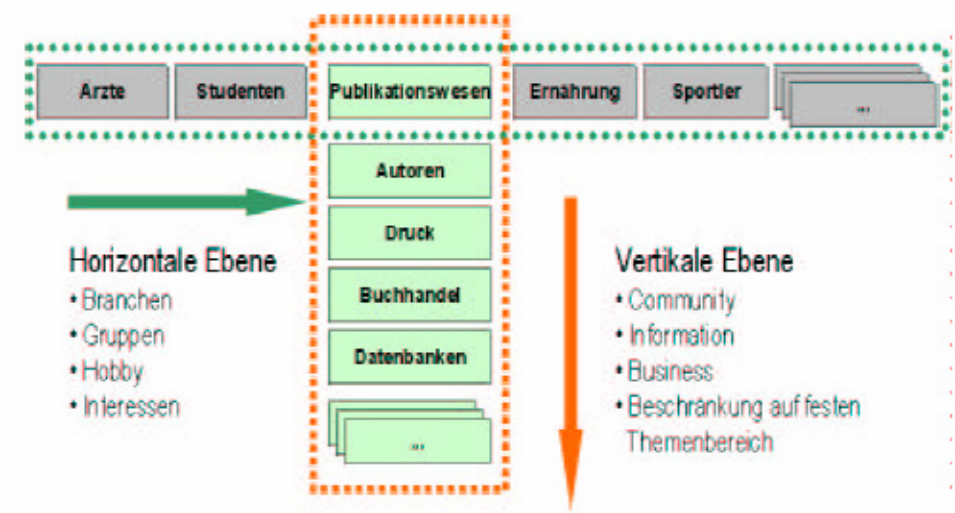


Abbildung 15: Horizontale und vertikale Ausrichtung von Portalen [OF01]

Dabei erkennt man sofort, das horizontale Portale nahezu alle denkbaren Themenbereiche abdecken soll. Sie eigenen sich daher sehr gut als Einstiegspunkt zu jeglicher Art von Information. Aufgrund ihrer „allrounder"-Qualitäten werden sie von der breiten Masse an Nutzern benutzt. Man nennt sie daher auch Verbraucher-Portale. Populäre Beispiele für solche Portale sind zum Beispiel *Yahoo* oder *Web.de*.

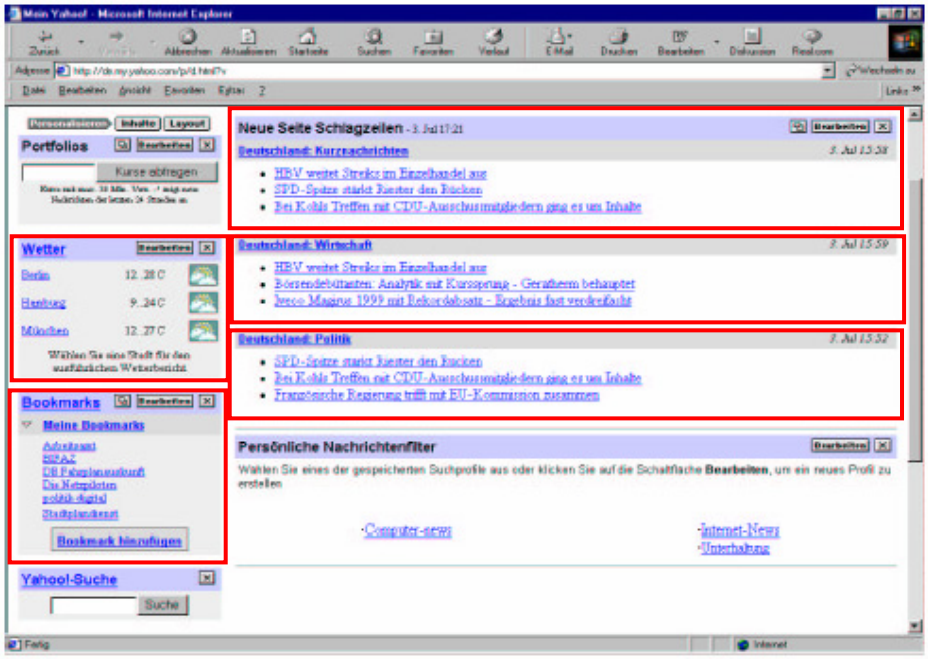


Abbildung 2: Personalisierte Seite des Internetportals Yahoo [OF01]

Im Rahmen der Personalisierung auf die unterschiedlichen Bedürfnisse der Benutzer stellt Yahoo Dienste wie z.B. regionen-genaue Wettervorhersage, herausgefilterte Nachrichten oder auch aktuelle Wertpapierkurse zur Verfügung. Wie in Abbildung 2 erkennbar, kann sich der Benutzer selbst die unterschiedlichen Portlets einbinden, die Anordnung bestimmen oder aber auch ihre Größe verändern.

Vertikale Portale unterscheiden sich von den horizontalen Vertretern nicht im Erscheinungsbild sondern nur anhand der bereitgestellten Dienste und Informationen, die speziell für bestimmte Themenbereiche bestimmt sind. Der Nutzerkreis beschränkt sich auf die, an dem behandelten Themenkomplex, interessierten Benutzer. Man nennt daher vertikale Portale auch Geschäftsportale, weil sie nur für geschäftliche Benutzer (Lieferant, Produzent, aber auch Konsument) interessant sind.

Eine weitere Kategorie bilden die Unternehmensportale. Unternehmenportale lassen sich gemäß ihrem Content in Web-Portale, Communitity-Portale, unternehmensweite Portale und andere einteilen. Die beiden Unterscheidungskategorien sind orthogonal zu einander, das heißt z.B. ein Webportal kann entweder horizontal als auch vertikal sein. Beispiel für ein horizontales Webportal wäre Yahoo. Yahoo bietet allgemeine Informationen zu allen erdenklichen Themen auf einer Weboberfläche. Im Bereich der unternehmensweiten Portale (Enterprise Portals) gibt es eine noch detailliertere Abstufung, die sich nach der Ausrchtung gemäß des betriebswirtschaftlichen Kontext richtet. So unterscheidet man hierbei in B2B(business-to-business)-, B2C(business-to-consumer)- und B2E(business-to-employee)-Portale. Während B2B-Portale hauptsächlich für den geschäftlichen Datenverkehr zwischen mehreren Unternehmen (z.B. Lieferant und Produzent) eingesetzt werden, stellen B2C-Portale den direkten Kontakt zum Kunden her (z.B. Kaufgeschäfte abwickeln). Wichtigste Kategorie hierbei sind aber die B2E-Varianten. Hierbei versucht man dem Mitarbeiter eines Unternehmens einen einfachen und raum-unabhängigen Zugriff auf die Daten des Unternehmens zu ermöglichen. [JR01]

Auf der Oberfläche dieser Enterprise Portals lassen sich interne sowie externe Informationen integrieren. Die unterschiedlichen Daten der einzelnen Informationsquellen lassen sich somit auch einheitlich, mit Hilfe des Portalsystems, auf einer Oberfläche präsentieren. Unternehmensportale können aber nicht nur Informationen darstellen, sondern haben zusätzlich die Fähigkeit Workflowkomponenten, wie z.B. E-Mail oder Kalender einzubinden. Zur Veranschaulichung ist dieser Sachverhalt auf der folgenden Abbildung 3 visualisiert.

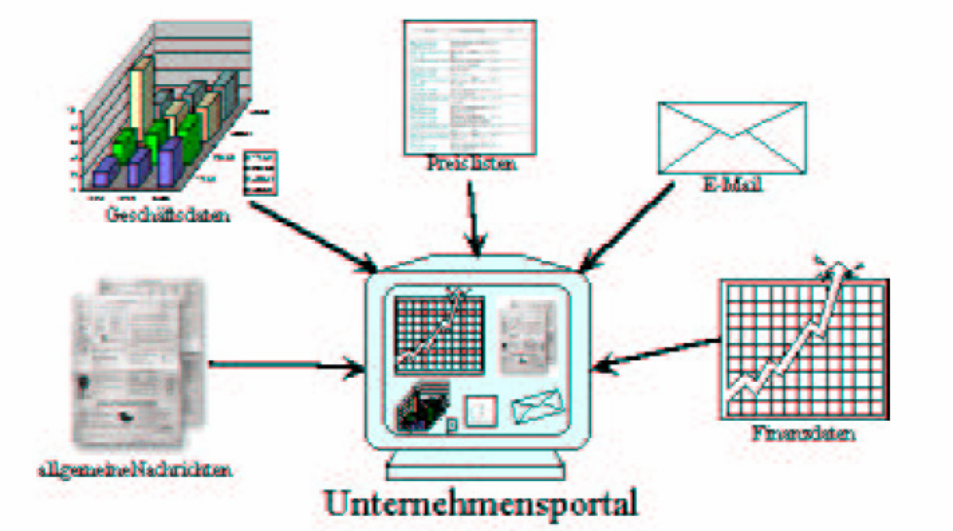


Abbildung 3: Unternehmensportale stellen Informationen aus verschiedenen Quellen auf einer Seite zur Verfügung [OF01]

2.3 Technologie der Portalsysteme

Aus technischer Sicht bilden Portalsysteme eine Art Framework in dem bestehende Anwendungen, Informationsdienste oder Workflowkomponenten eingebunden werden können. Dabei besteht dieses Framework aus einer Ansammlung verschiedener Funktionsblöcke, die zum Teil von unterschiedlichen Systemen realisiert werden. Abbildung 4 zeigt hier die Basisbestandteile, die ein Portalsystem bereitstellt.

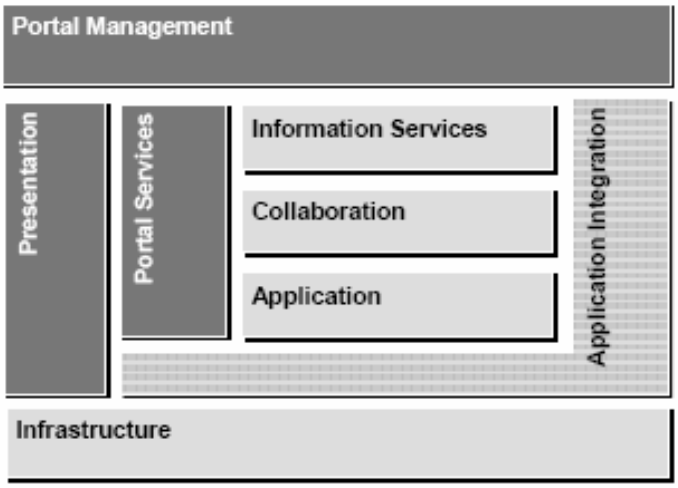


Abbildung 4: Bestandteile eines Portals [JR01]

Die Blöcke Information Services, Collaboration und Applikation repräsentieren die, schon vor Einführung des Portalsystems, vorhandenen Systeme. Die Infrastructure beinhaltet neben Hardware und Betriebssystem auch Application Server, Webserver sowie Verzeichnis- und Transaktionsdienste. Grob kann man die vorhandenen Systeme in drei Kategorien aufteilen:

- *Information Services* repräsentieren klassische Informationen wie z.B. Nachrichten, Dokumente, Informationsdienste. Sie werden üblicherweise mit Hilfe von Content-Management-Systemen (CMS) verwaltet.

- *Collaboration* beschreibt Systeme die in irgendeiner Weise den Arbeitsablauf des Nutzers unterstützen. Hierbei handelt es sich z.B. um E-Mail oder Kalender. Unter Collaboration fallen aber auch z.B. Workflowmanagementsysteme.

- *Application* umfasst alle herkömmlichen Anwendungen, die in das Portalsystem eingebaut werden sollen. Dabei müssen diese Anwendungen nicht zwangsläufig web-fähig sein, dies wird von anderen Komponenten durchgeführt.

Mittels der dunkelgrau gekennzeichneten Komponenten sollen die portalspezifischen Funktionen zusammengefügt werden. Dabei übernehmen die einzelnen Komponenten folgende Aufgaben:

- *Application Integration* lässt sich als orthogonale Integrationsschicht verstehen, deren Aufgabe es ist, die bereits vorhandenen Systeme *interoperabel* zu machen. Interoperabel bedeutet, dass die vorher isolierten Einzelsysteme untereinander verknüpft sind und Inter- bzw. Intranettauglickeit besitzen. Diese Komponente spielt eine sehr wichtige Rolle in dem Konzept von Priebe und Pernul, das ausführlich im Kapitel 4 beschrieben wird.

- *Portal Services* übernimmt die portalspezifischen Aufgaben und Funktionen wie z.B. die Personalisierung. Zusätzlich realisiert diese Komponente die gesamte Verwaltung der Portletinhalte (Content) sowie die Navigation.

- *Presentation* gewährleistet die geräteabhängige Darstellung der Inhalte und Anwendungen, damit das Portalsystem auch im Hinblick auf Portabilität voll flexibel ist.

- *Portal Management* ist die globale Schicht, die oberhalb der anderen Komponenten die globale Kontrolle des gesamten Portals ausübt. Ihre Aufgaben sind neben der globalen Benutzerverwaltung auch das Monitoring der einzelnen Subsysteme.

Portale sind somit keine eigenständigen Systeme sondern eher ein Zusammenschluss mehrerer Komponenten, die verschiedenste Anwendungen und Dienste, auf Basis einer bestehenden Infrastruktur integrieren. Um die Theorie der Portale nun ein wenig mit Leben zu füllen wird im nächsten Kapitel näher auf das freie Open-Source-Portalsystem *Jetspeed* eingegangen.

2.4 Jetspeed

Jetspeed ist ein Projekt von *Apache Software Foundation*[47] und ist Open-Source[48]. Das heißt der Quellcode unterliegt zwar der Java Apache Projekt Lizenz, welche aber die Nutzung und auch Veränderung erlaubt. Das Framework ist komplett in Java implementiert und hat somit volle Flexibilität im Bereich Portabilität und Erweiterbarkeit. Jetspeed stellt ausschließlich Klassen bereit, die für die Funktionalität des Portals nötig sind, jedoch nicht für den Inhalt. Der Portlet-Content wird mit einer eigenen Beschreibungssprache dargestellt, der *Portal Structure Markup Language*, kurz *PSML*. Diese Beschreibungssprache ermöglicht die personalisierte Darstellung der unterschiedlichen Informationen in den einzelnen Portlets. Das Portalkonzept beschreibt die Idee, einem Benutzer auf einer Portalseite beliebige Informationen und Anwendungen zusammen zu präsentieren. [OF01] Um verschiedenste Inhalte in das Portal einbinden zu können benötigt man Schnittstellen. Diese Schnittstellen werden in Jetspeed Portlets genannt. Der Begriff ist ein wenig irreführend gewählt, da die einzelnen Bereiche auf der Oberfläche der Portalseite ebenfalls Portlets genannt werden. Bei Portlets im Jetspeed Kontext handelt es sich aber um die Schnittstellendefinition zum Portalkern, der dann die speziellen Dienste als „visuelles“ Portlet realisieren kann.

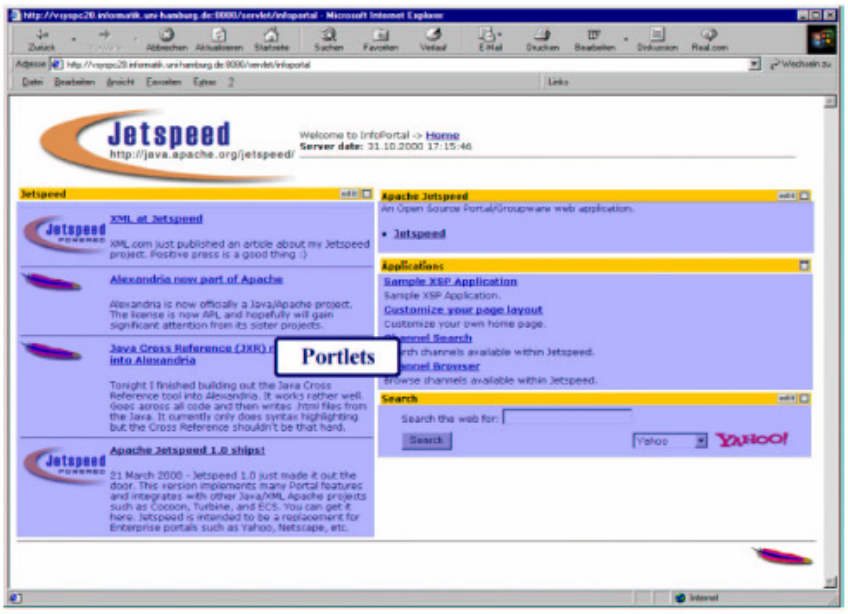


Abbildung 5: Eine Jetspeed-Portalseite [OF01]

[47] **Apache Software Foundation** - Eine Gruppe von Open Source-Softwareentwicklern, die mit dem gleichnamigen Webserver begonnen hat und inzwischen eine ganze Reihe von XML-Tools bereitstellt
[48] **Open-Source** - Bezeichnet Programme, deren Quellcode öffentlich ist. Jeder kann so unbeschränkt Einblick nehmen, Veränderungen vornehmen oder Anwendungen schreiben. Beispiel Linux / OpenOffice

Aus der Sicht des Benutzers eines Portalsystems wird eine Portalseite folgendermaßen generiert. Es wird anhand der PSML-Beschreibung, die für jeden Benutzer angelegt ist eine Art Layout für die Seite aufgebaut. In diesem Layout befinden sich die einzelnen Portlets an vorbestimmten Bereich. Danach werden die Portlets aufgerufen und in das Gesamtlayout eingefügt. Somit erhält der Benutzer gemäß seiner persönlichen Anforderungen eine individuelle Darstellung der benötigten Inhalte.

Spezielle Portlets, die so genannten *Controller* und *Controls*, sind Komponenten des Portalkerns und sind für das Layout und die Generierung der Portalseiten zuständig. *Controller* und *Controls* werden ebenfalls in der PSML-Seitenbeschreibung aufgenommen und garantieren so die personalisierte Darstellung für jeden einzelnen Benutzer des Systems. Siehe hierzu Abbildung 6.

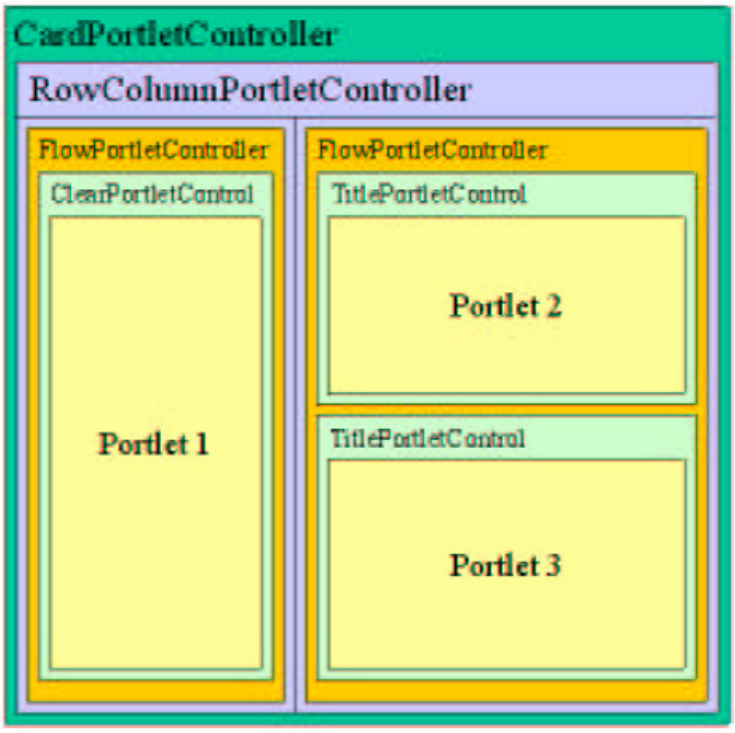


Abbildung 6: Generierung einer Portalseite mit Hilfe von Controller und Control [OF01]

Eingebundene Frameworks – speziell das Turbine-Framework

Das Jetspeed-Portalsystem bedient sich anderer Frameworks um verschiedene Funktionen, wie z.B. die Benutzerverwaltung oder das persistente Speichern von Objekten, zu realisieren. Dazu integriert Jetspeed andere, so genannte Open-Source Application Frameworks wie z.B. Turbine, Cocoon[49] oder Castor[50].

Eines der wichtigsten eingebunden Frameworks ist das Turbine-Framework. Turbine ist ein, ebenfalls in Java implementiertes, Framework zur Entwicklung von sicheren webbasierten Anwendungen. [OF01] Es stellt eine umfangreiche Bibliothek mit Funktionen bereit, die zur Realisierung dieser web-fähigen Anwendungen benötigt wird. Turbine erweitert aber auch den *Webserver* über eine geeignete Servletschnittstelle zu einem *Application Server* und ist daher mit das wichtigste eingebundene Framework von Jetspeed.

Turbine delegiert die Erstellung, der einzelnen Seiten einer Webanwendung, an mehrere andere Komponenten, die auch als Module bezeichnet werden:

- *Page-Modul*. Beschreibt die zu generierende Seite und verwendet wiederum zur Generierung der Ausgabe das *Layout-Modul*.

- *Layout-Modul.* Besteht aus mehreren *Navigation-Modulen* und einem *Screen-Modul*.

- *Navigation-Modul.* Ist für die Teile der darzustellenden HTML-Seite verantwortlich, die auf mehreren Seiten immer wieder kommen. Vorteil, dabei ist, dass dieses Modul bei den anderen Seiten wiederverwendet werden kann. Beispiel hierfür wäre z.B. eine Navigationsleiste oder ein Menü. Meistens wird dieses Modul am Seitenkopf oder –fuß dargestellt. Dieses Konzepts erinnert ein wenig an *Frames* in *HTML.*

- *Screen-Modul*. Erstellt den Inhalt der angeforderten Seite. Sie sind in aller Regel nicht wieder verwendbar.

Das *Layout-Modul* definiert welche *Navigations-Module* und welches *Screen-Modul* in der Seite verwendet werden sollen. Zusätzlich übernimmt das *Layout-Modul* auch noch die grafische Gestaltung, sprich wo welches Modul innerhalb der Seite eingefügt werden soll. Einen grafischen Überblick gibt Abbildung 7.

[49] **Cocoon** ist ein auf Java basierendes Open Source XML Publishing System (Framework) der Apache Software Foundation.
[50] **Castor** ist ein Open Source Data Binding System basierend auf Java[TM]

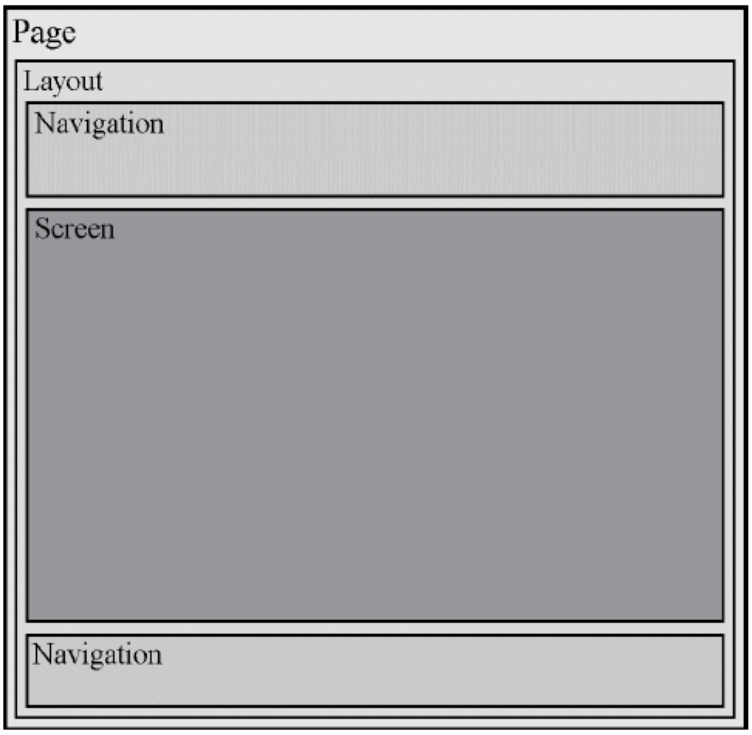


Abbildung 7: Module einer Seite beim Turbine Framework [OF01]

Turbine unterstützt neben diesem Modell, zur Generierung von Portalseiten, auch die Bearbeitung von Benutzerinteraktionen. Diese Aktionen werden durch Anklicken von Links, Abschicken von Formularen etc. ausgelöst. [OF01] Diese Aktionen sind so genannte *Requests*, die der Webserver und somit auch Turbine erhält. Turbine ist in der Lage diese *Requests* serverseitig individuell zu bearbeiten und somit verschiedenste Aktionen auszuführen. Dadurch findet eine Trennung, gemäß dem MVC[51]- (Model-View-Controller Pattern), in Darstellung, Aktionsbearbeitung und Datenhaltung statt.

Weitere Turbine-Funktionalitäten sind u.a.:

- API für Benutzerverwaltung und Benutzer-Login
- Sicherheitsmodell mit Rechten und Rollen
- URL-Parser zur Ermittlung von Aktionen aus der URL
- …

[51] **MVC** Es propagiert eine Dekomposition komplexer interaktiver Systeme in Modell-, Controlling- und View-Komponenten

Abbildung 8 zeigt ein Beispiel für eine angeforderte Portalseite. Gekennzeichnet sind hier die Module *Navigation* und *Screen*.

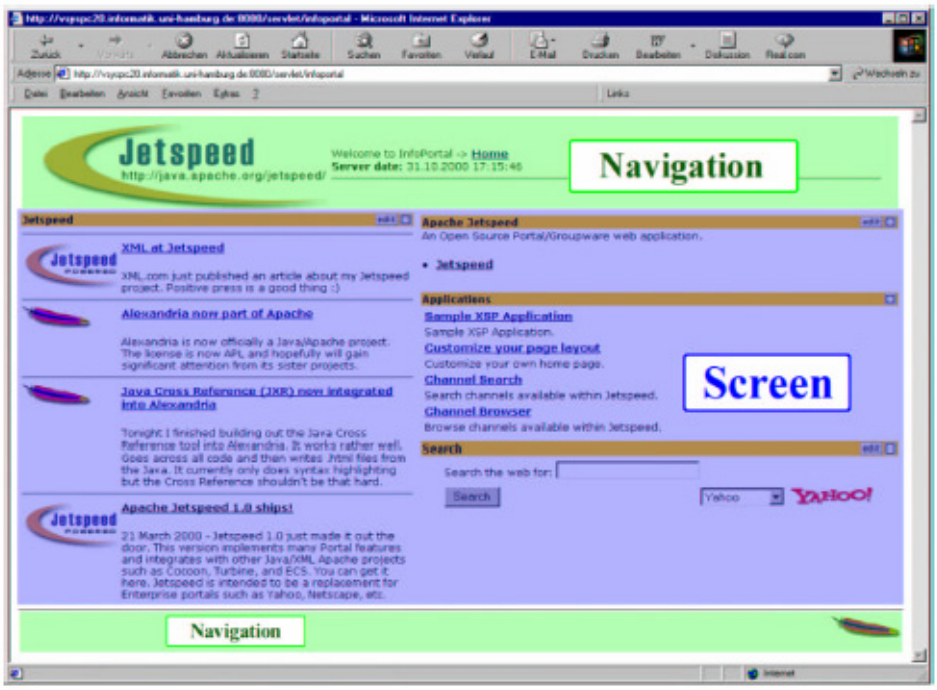


Abbildung 8: Screen-Modul und Navigation-Modul [OF01]

Im folgenden Kapitel soll nun der Versuch unternommen werden innerhalb des Portalsystems eine Kommunikation zwischen den einzelnen Portlets herzustellen. Diese Kommunikation ist Vorraussetzung für das Einbinden weiterer Funktionalität wie z.B. Kontextbasiertes Information Retrieval (s. Kapitel 4). Man merkt schnell, dass diese Kommunikation alles andere als trivial herzustellen ist. Dies liegt weitestgehend an der Heterogenität der einzelnen Portletinhalte. Egal ob einfache Informationen aus dem Internet/Intranet oder eine OLAP-Anfrage, zur Kommunikation benötigt man eine gemeinsame Basis. Ziel wird es sein einen gemeinsamen Kommunikationsstandard zu definieren, der für alle Portlets gleichermaßen „verständlich“ ist. Hier kommen wiederum Konzepte wie *Semantic Web*, das *Resource Description Framework (RDF)* und *Ontologien* zum Einsatz, die Gegenstand des folgenden Kapitels sind.

3. Technologie RDF/Ontologien und das Semantic Web

Ziel dieses Abschnitts ist es, einen groben Einblick in die Welt des semantik-basierten Wissensmanagements zu geben. Semantik-basiert bedeutet, das Wissen (z.B. Dokumente) semantisch zu adressieren. Sinn und Zweck dabei ist es, dieses Wissen so, maschinenverständlich zu machen, um, im Hinblick auf die stetig steigende, „Flut" an Informationen (z.B. aus dem Internet), effizientes Wissensmanagement und besonders effiziente Suche im Wissenspool zu ermöglichen. Ein bedeutsamer Ansatz ist hier die Vision des Semantic Web von Tim Barner-Lee.

Wesentliche Voraussetzung für die Realisierung von semantik-basiertem Wissensmanagement, oder des Semantic Web, sind die Technologien *Resource Description Framework (RDF)* und *Ontologien*. RDF bietet dabei eine standardisierte Metadatenstruktur auf Basis von *XML*[52] und Ontologien ermöglichen die Beschreibung übergreifende Zusammenhänge zwischen Konzepten in einer „gemeinsamen Sprache". Zusammen bilden RDF und Ontologien eine Art erweitertes Metadatenmodell. Im Folgenden wird näher auf diese Konzepte eingegangen.

3.1 Semantic Web

Eine fiktive Situation, die womöglich schon jedem einmal widerfahren ist, soll den Sachverhalt um das Semantic Web verdeutlichen. Auf der Suche nach Bildern zum Thema „Raubkatzen – Der Jaguar" erhalten wir bei verschiedenen Suchmaschinen auf die Suchanfrage „Jaguar" überraschenderweise nicht sofort Informationen zu dem Tier, vielmehr werden das gleichnamige Auto oder aber die Zerstörungkraft des Kampfjets „Jaguar", angepriesen. Hierbei erkennt man, dass „Jaguar" als alleinstehender Begriff keine eindeutige Semantik aufweist. Erst Zusatzbeschreibungen wie z.B. Tier, Auto oder Kampfjet ermöglichen eine eindeutige Zuordnung. Diese Daten könnten somit als Metadaten[53] fungieren, und somit die Kategorisierung der Informationen, Beispiel Bilder, einfacher zu gestalten. [KQ02]

Die Einführung dieser Metadaten ist zentraler Bestandteil des Konzepts des Semantic Web nach Barner-Lee: *«The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation»* [BL01] Der zusätzliche Aufwand bei der Datenaufbereitung, im Sinne des Semantic Web, durch die Erweiterung der Daten um Metadaten, soll durch die, dadurch ermöglichten, Vorteile gerechtfertigt werden.

---

[52] **XML (Extensible Markup Language)** bezeichnet einen Standard zur Definition von Auszeichnungs-Sprachen, der als vereinfachte Teilmenge von SGML konzipiert wurde.

[53] Als **Metadaten** oder **Metainformationen** bezeichnet man allgemein Daten, die Informationen über andere Daten enthalten. Bei den beschriebenen Daten handelt es sich oft um größere Datensammlungen (Dokumente) wie Bücher, Datenbanken oder Dateien.

Nach Barner-Lee könnten dadurch Suchmaschinen sehr viel bessere Ergebnisse liefern oder „intelligente Agenten“ miteinander kommunizieren um komplexere Aufgabenstellungen zu lösen. Der Wahrheitsgehalt der Aussagen wir durch das so genannte „Web of Trust“ sichergestellt. Die Inter-System-Kommunikation wird im weiteren Verlauf der Ausarbeitung auf das Konzept der Portalsysteme ausgeweitet, um kontextbasiertes Information Retrieval in das Portalsystem zu integrieren (s. Kapitel 4).

Folgende Abbildung 9 gibt einen Überblick über die Struktur und die verwendeten Technologien des Semantic Web.

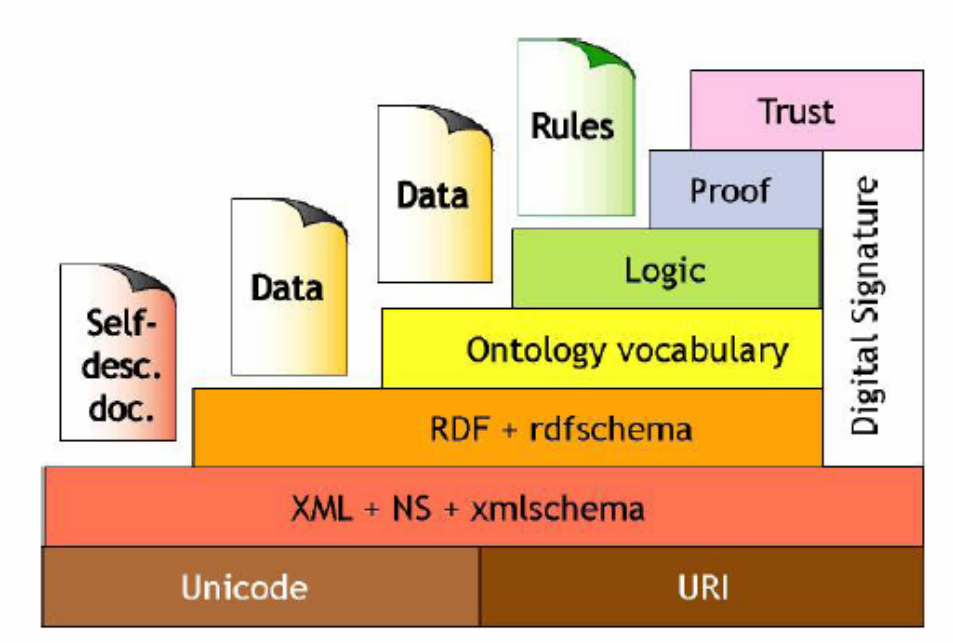


Abbildung 9: Die Schichten des Semantic Web [UK04]

Wichtigste Voraussetzung zur Verwirklichung des Semantic Web sind die Technologien *Uniform Resource Identifier (URI)*, *Metadaten* (in RDF) und *Ontologien*. Diese Konzepte haben folgende Aufgaben: [KS02]

- **Uniform Ressource Indentifier (URI)**
  URI ermöglicht die eindeutige Identifikation der Daten. Dies ist notwendig, um eine konsistente Kommunikation zwischen mehreren Agenten zu ermöglichen. Die bekannteste Form einer *URI* ist eine *URL* (*Uniform Resource Locator*), die eine Ressource identifiziert und lokalisiert.

- **Metadaten (z.B. RDF und RDFS(chema))**
  Metadaten dienen der exakten Beschreibung einer Ressource. Beispiel Bibliothek: Bücher werden mit Hilfe von zusätzlichen Informationen (Metadaten) katalogisiert und verwaltet. Hierzu zählen Autor, Titel oder auch Stichworte. Sie dienen dazu, Bücher in einer Bücherei zu suchen und dann auch wieder zu finden.

- **Ontologien**
  Ontologien stellen Beziehungen zwischen Objekten (Ressourcen) dar und ermöglichen damit, dass Maschinen mit diesen Objekten arbeiten können. Aus den definierten Beziehungen lassen sich mit Hilfe von Logiken, weitere Informationen ableiten.

Im Folgenden werden die Konzepte RDF als Form der Metadatenbeschreibung als auch Ontologien näher beschrieben.

3.2 Ontologien

Ontologien kommen ursprünglich aus dem Bereich der Philosophie und beschreiben die Natur sowie die Organisation der Wirklichkeit. In dem vorliegenden Kontext verstehen sich Ontologien als formale Modelle, die den Austausch von Wissen zwischen mehreren Maschinen gestatten. Eine Ontologie definiert ein *wohldefiniertes* Vokabular von Symbolen, sowie ein einheitliches Verständnis, welche Begriffe und Beziehungen die Begriffe beschreiben.

Folgende Kommunikationssituation verdeutlicht das Einsatzbedürfnis von Ontologien.

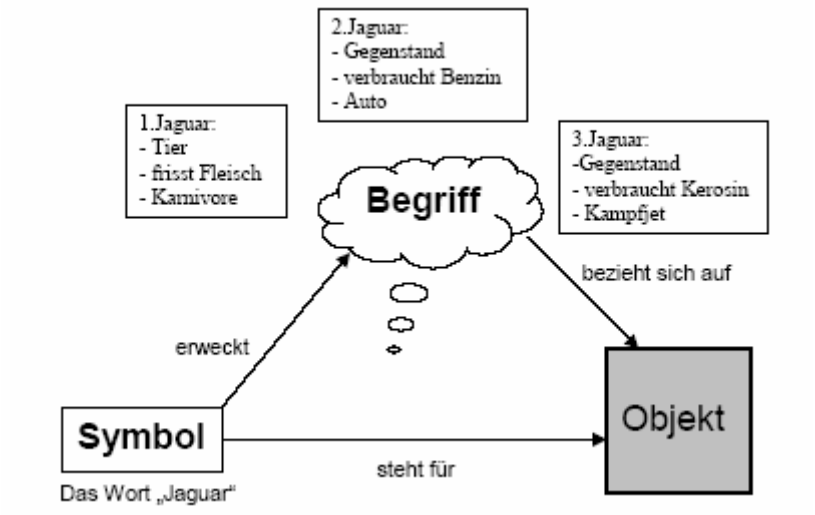


Abbildung 10: Bedeutungsdreieck – Ontologie am Beispiel „Jaguar" [KQ02]

Ein Symbol „erweckt“ bei jedem Akteur einen Begriff, der dessen gedankliche Interpretation des Symbols darstellt. Dieser Begriff „bezieht sich auf“ einen realen Gegenstand, ein Objekt. Ziel ist es, dass alle Akteure (Sender und Empfänger) sich das gleiche reale Objekt vorstellen. In dem Beispiel ist der Begriff „Jaguar“ aber mehrdeutig. Um alle Akteure eindeutig auf das gleiche reale Objekt zu „matchen“ wird nun die Hilfe eines gemeinsamen Sprachstandards benötigt. Dies leisten Ontologien wie sie in den in der Abbildung 10 dargestellt sind. Ein „Jaguar“ der Tiere frisst bezieht sich eindeutig auf das Tier und nicht auf ein Automobil[KQ02].

Ontologien sollen Wissen einer spezifischen Domäne derart darstellen, dass beliebige Akteure (Mensch oder Maschine) gleichermaßen ein allgemeines Verständnis dieser Domäne haben. Deshalb gliedert sich eine Ontologie hierarchisch, d.h. die Begriffe einer Domäne werden zueinander in Beziehung gesetzt und zusätzlich mit Beschreibungen, Attributen und Relationen erweitert. Dadurch wird die Verbindung eines Begriffs zu dem beschriebenen realen Gegenstand eindeutig. Im Gegensatz zur rein syntaktisch angelegten Austauschsprache XML hebt eine Ontologie die Kommunikation zwischen Akteuren auf eine semantische Ebene.

Eine Möglichkeit Ontologien zu realisieren bietet RDF.

3.3 Resource Description Framework (RDF)

Ontologien als Träger von Definitionen lassen sich mit dem RDF-Standard und *RDF Schemas (RDFS)* realisieren. RDF allein erfasst beliebige Metadaten über verschiedenste Objekte und macht sie somit maschinenlesbar (syntaktische Ebene). RDFS dagegen sorgt dafür, dass die Daten maschinenverständlich (semantische Ebene) sind [KQ02]. Grundlage von RDF ist XML, d.h. die RDF-Metadaten sowie die RDFS werden in dem systemneutralen Standard XML notiert.

Folgendes Beispiel (Abbildung 11) macht den Zusammenhang zwischen RDF und RDFS sichtbar. Über das RDF-Schema werden Klassen definiert und in einer so genannten Taxonomie[54] angeordnet (*rdfs:subclassOf* → unbeschriftete Pfeile). In RDF werden dann die Instanzen der Klassen abgebildet (*rdf:type* → beschriftete Pfeile).

[54] Die **Taxonomie** ist die Einteilung von Dingen, insbesondere Organismen, in Taxa (Sing.: Taxon) (Gruppen).

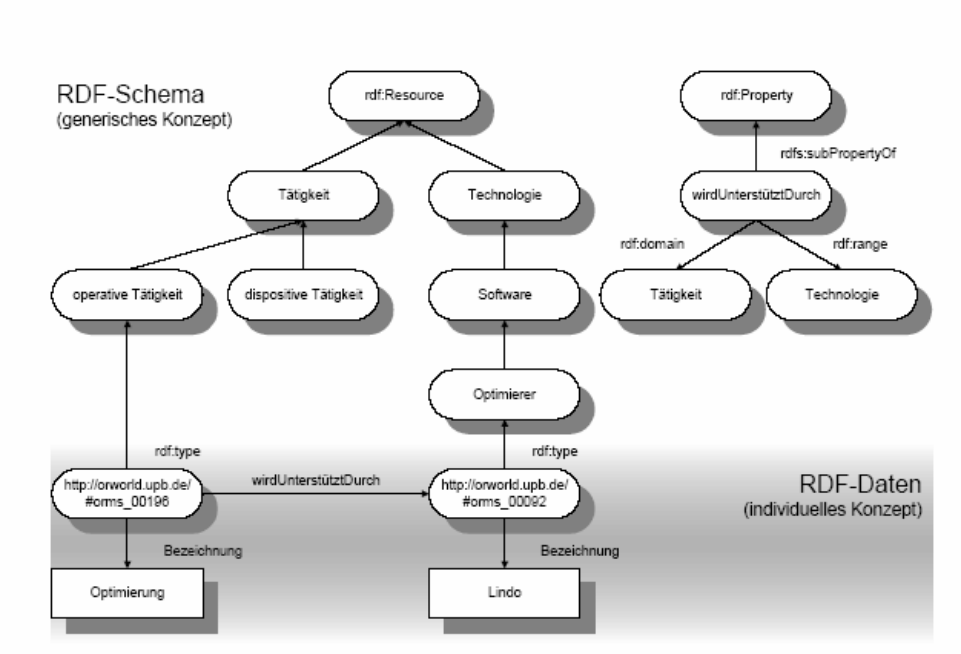


Abbildung 11: Zusammenwirken von RDF und RDFS [AR02]

RDF-Schema

RDF als Repräsentation von Metadaten ist auf Instanzebene definiert. Um Strukturinformationen des generischen Konzeptes (Schemaebene) zu beschreiben, benötigt man zusätzlich RDFS. Es erlaubt die Definition von Schemata, die Begriffe der Ontologie maschinenverständlich und damit semantisch festlegen. Hierbei bedient man sich folgender Konstrukte:[AR02]

- *rdfs:Class* beschreibt Klassen von ähnlichen Objekten.
- *rdfs:subClassOf* ist eine vordefinierte Eigenschaft, welche eine Klasse als Unterklasse einer anderen definiert.
- *rdf:Resource* ist die Superklasse in jedem Schema. Jede definierte Klasse ist rdfs:subClassOf rdf:Resource.
- *rdf:type* ist eine Eigenschaft, die eine Ressource als Instanz einer bestimmten Klasse definiert. Der Wert dieser Eigenschaft definiert diese Klasse.

- *rdf:Property* stellt die allgemeinste Eigenschaft dar. Jede definierte Eigenschaft ist rdfs:subPropertyOf rdf:Property.

- *rdfs:subPropertyOf* beschreibt die Beziehungen zwischen einer allgemeineren und einer spezielleren Eigenschaft.

- *Constrains* In RDFS können Eigenschaften dadurch näher spezifiziert werden, indem der Anwendungsbereich dieser Eigenschaft auf bestimmte Klassen beschränkt wird (rdfs:domain) oder der Wertebereich definiert wird (rdfs:range).

RDF-Datenmodell

RDF hat vier grundlegende Komponenten: [AR02]

- *Ressourcen:* Eine Ressource kann, ähnlich wie in einem ERM, ein materielles oder immaterielles Objekt sein, welches in RDF eindeutig über einen so genannten URI (Unified Resource Identifier) identifizierbar ist. Somit kann z.B. auf Webseiten, Personen, Unternehmen und auch andere (Wissens-) Objekte durch natürliche oder künstliche Schlüssel gezeigt werden.

- *Eigenschaften (Properties):* Eine Property ist eine Eigenschaft, ein Attribut oder eine Relation, die dazu benutzt wird, eine Ressource zu beschreiben. Die Bedeutung eines Propertys wird durch die Definition der zulässigen Werte, der Typen der Ressourcen, die es beschreiben kann, und die Definition der Beziehung dieses Propertys zu anderen Properties festgelegt.

- *Literale (Literals):* Literale können atomare Werte wie Unicode-Zeichenketten beinhalten. Eigenschaften können durch Literale ausgedrückt werden.

- *Aussagen (Statements):* In einem Statement wird der Property einer Ressource ein Wert zugewiesen, wobei dieser Wert auch wieder eine Ressource sein kann. Somit stellen die Statements als Tripel Subjekt-Prädikat-Objekt die Grundidee des RDF dar.

RDF wird vollkommen in XML-Syntax spezifiziert. Im Rahmen dieser Ausarbeitung soll aber keine umfassende Einführung dieser Beschreibungsform gegeben werden.

In den folgenden Kapiteln wird nun der Bogen zurück zu den Portalsystemen und dem Ziel der Integration kontextbasiertem Information Retrieval gespannt. Ontologien und das Konzept des Resource Description Framework spielen eine wichtige Rolle bei dem, im Kapitel 4 vorgestellten, Ansatz von Priebe und Pernul.

4 Integration von kontextbasiertem Information Retrieval

In diesem Kapitel wird der Ansatz von Torsten Priebe und Günther Pernul genauer betrachtet. Hierbei wird zunächst das generelle Konzept für die kontextbasierte Integration von Wissensportalen beleuchtet. Anschließend wird der Begriff Kontext und dessen Darstellung im Vorschlag von Priebe und Pernul erläutert. Daraufhin wird das dahinter stehende Information Retrieval Model beschrieben, die allgemeine Architektur von Portalen dargestellt und schließlich der Prototyp INWISS[55] vorgestellt.

4.1 Das Konzept von Priebe und Pernul

Der im folgenden beschriebene Entwurf von Priebe und Pernul [PePK03] geht davon aus, dass es „[i]m Sinne einer effizienten Wissensnutzung [...] wünschenswert [wäre], wenn der in einem Portlet durch (wie auch immer geartete) Navigation offenbarte Informationsbedarf des Benutzers auch in anderen Portlets zum Auffinden passender Informationen genutzt werden könnte“ ([PePK03], Seite6).

Als mögliche Anwendung nennen die Autoren hier ein Finanzportal, das es seinen Benutzern ermöglicht, sich in einem Portlet über die Aktienkurse von Unternehmen zu informieren. Gleichzeitig würden den Usern in einem News Portlet Nachrichten angeboten, die das jeweilige Unternehmen betreffen, dessen Aktienkurs gerade betrachtet wird. In diesem Fall würde das Aktienkurs Portlet dem News Portlet mitteilen, welches Unternehmen für den Nutzer aktuell interessant ist, damit dieses nach entsprechenden Nachrichten suchen kann.

Das Beispiel verdeutlicht, wie sinnvoll es sein kann, wenn Portlets ihren Kontext untereinander kommunizieren, d.h. wenn ein Portlet, dessen Inhalt gerade vom Benutzer verändert wird, diesen an andere Portlets weitergibt, damit diese sich ihrerseits den Aktionen des Benutzers anpassen können. Durch eine Interaktion der Portlets könnte der Nutzer zusätzlich bei seiner Informationssuche unterstützt werden.

Ziel von Priebe und Pernul ist es hierbei, eine generische Lösung zu finden, die in Standard-Portalplattformen integriert werden kann. Der Architekturrahmen soll die Kommunikation des aktuellen Benutzerkontextes über Portletgrenzen hinweg und damit eine Integration der Portlets ermöglichen.

[55] Genauere Informationen unter http://www.inwiss.org

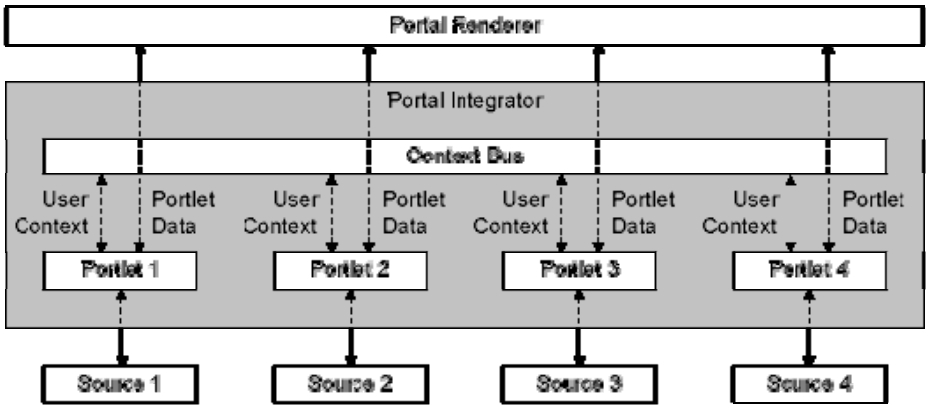


Abbildung 12: Architekturrahmen für kontextbasierte Portlet-Integration

Abbildung 12 zeigt den Architekturrahmen zur kontextbasierten Portlet-Integration. Bei der Veränderung eines Portlet setzt dieses eine Nachricht mit Informationen über seinen aktuellen Kontext auf den so genannten Context Bus. Andere Portlets können diese Nachricht abgreifen und sich entsprechend anpassen. [PePK03]

Hier stellt sich nun die Frage, was den kommunizierten Kontext der einzelnen Portlets ausmacht und wie er sich darstellen lässt. Henrich und Morgenroth [HM02] unterscheiden bei dem Begriff Kontext drei Kategorien, den Benutzerkontext, den Arbeitskontext und den Interaktionskontext.

- Der Benutzerkontext beschreibt den physikalischen und organisatorischen Kontext, sowie das Profil des Benutzers. Es beinhaltet beispielsweise dessen geographischen Aufenthaltsort, seine Position innerhalb der Organisation eines Unternehmens und Informationen über seine Wissensgebiete und Fähigkeiten. Hierbei handelt es sich um relativ statischen Kontext, d.h. er verändert sich nur sehr langsam in Zeiträumen von Monaten und Jahren.

- Die aktuelle Tätigkeit des Benutzers wird durch den Arbeitskontext ausgedrückt. Dieser kann entweder durch die in einem Workflow wohl definierten Aufgaben oder bei einer unstrukturierten Tätigkeit durch die ausgeführten Aufgaben und Aktionen des Benutzers beschrieben werden. Eine Änderung des Arbeitskontext ist wesentlich häufiger zu erwarten, als eine Änderung des Benutzerkontext. Ändern sich die Aufgaben des Users im Unternehmen, so geschieht dies auch mit dessen Arbeitskontext.

- Der Interaktionskontext schließlich besteht aus den Nachrichten von Menübefehlen oder Dialogen wie beispielsweise dem Hervorheben von Wörtern oder dem Öffnen bestimmter Menüpunkte. Er ist nicht direkt für ein Retrieval System nutzbar, sondern erfordert eine gewisse Aufbereitung. So reicht die Information, dass ein Wort in einem Text markiert wurde ohne Kenntnis des Wortinhaltes nicht aus. Der Interaktionskontext ist dynamisch, d.h. er unterliegt einem permanenten Wandel innerhalb weniger Sekunden oder Minuten. [HM02]

Im Ansatz von Priebe und Pernul [Pr04] werden nur der Benutzerkontext, vor allem aber der Interaktionskontext betrachtet. Der Arbeitskontext wird außer Acht gelassen, da eine strukturierte Repräsentation der Aufgaben eines Nutzers, die für eine genaue Beschreibung des Kontext nötig wäre, nicht vorausgesetzt werden kann. Nur im Fall der Nutzung eines Workflow Management Portlet, das auf einer definierten Struktur aufsetzt, könnte die Beschreibung der Aufgaben des Nutzers auch der Beschreibung des Inhalts dieses Portlet dienen. Der Portalkontext bei Priebe und Pernul bezieht sich somit auf den statischen Benutzerkontext sowie eine Menge von Portlet Kontexten. [Pr04]

Die größte Schwierigkeit bei der Repräsentation des Kontext stellt die Heterogenität der verschiedenen Informationsquellen und somit auch der Portlets dar. Während der Inhalt eines OLAP Portlet anhand der in ihm dargestellten Kennzahlen, Dimensionen und der Selektionskriterien beschrieben werden kann, ist dies für ein Portlet mit Zugriff auf ein Dokumenten-Management-System nicht möglich. [PePK03, Pr04]

4.2 Darstellung des Kontext

Priebe und Pernul verwenden zur Kontextrepräsentation das Resource Description Framework (RDF), um eben die Repräsentation des Kontext, sowie die Abbildung des Kontext unterschiedlicher Portlets zu ermöglichen. Konkret geschieht dies mit Hilfe von Taxonomien und Ontologien, die es erlauben, die Kontextinformationen unterschiedlicher Portlets in Beziehung setzen oder aufeinander abzubilden.

Um die Interoperabilität zu gewährleisten, die mit Hilfe des Context Bus realisiert werden soll, basieren die verwendeten Metadaten-Elemente auf dem Dublin Core Standard[56]. Dieser definiert 15 Elemente mit einer festgelegten Semantik. Einige dieser Elemente sind beispielsweise *Title*, *Creator*, *Subject*, *Description* und *Coverage*. Dabei sind jedoch nur die Elemente *Subject* und *Coverage* zur Beschreibung des semantischen Kontext der Ressourcen für den vorliegenden Ansatz von Nutzen. Während der Inhalt des Elementes *Subject* für gewöhnlich als Bestandteil einer hierarchischen Taxonomie dient, enthält *Coverage* den semantischen Kontext der jeweiligen Ressource. Priebe und Pernul schlagen dazu vor, eine unternehmensweite Ontologie als kontrolliertes Vokabular einzusetzen.

[56] [DCM03] zitiert in [Pr04]

Bei einer Taxonomie handelt es sich um eine hierarchisch aufgebaute Menge von Kategorien. Eine Ressource kann dabei Teil mehrerer Kategorien sein, d.h. mehreren Themen zugeordnet werden.

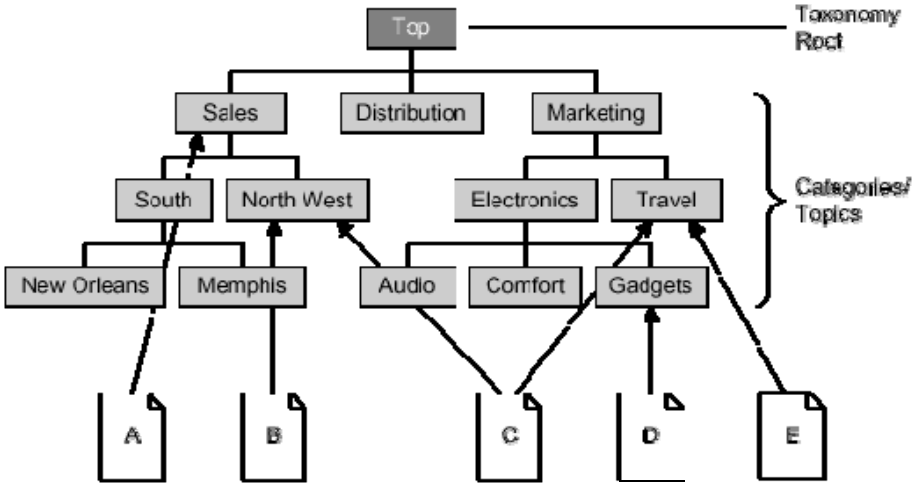


Abbildung 13: Kategorisierung von Ressourcen mit Hilfe einer hierarchischen Taxonomie

Abbildung 13 zeigt eine beispielhafte Taxonomie für ein Unternehmen, das verschiedene Konsumgüter über Calling Center vertreibt. Hier ist zu erkennen, dass zuerst eine Unterteilung in die Kategorien Verkauf, Vertrieb und Marketing erfolgt. Der Verkauf wird anschließend nach Regionen untergliedert, während im Marketing nach unterschiedlichen Produkten unterschieden wird.

Taxonomien werden vor allem für das Browsing, also eine Navigation durch die Kategorien, verwendet. Aus diesem Grund sind einfache, sprechende Bezeichnungen wichtig. Realisiert wird dies im vorliegenden Beispiel mit Hilfe des Elements *Subject* des Dublin Core Standards.

Um den Inhalt einer Ressource vollständig zu beschreiben, bedarf es jedoch komplexerer Ausdrucksmittel. Für diesen Zweck wird im vorgestellten Ansatz eine Ontologie verwendet.

Ontologien beschreiben einen bestimmten Ausschnitt aus der Realität. Ihre Instanzen stellen die Objekte der jeweiligen Domäne dar. Ontologien sind im Allgemeinen deutlich größer und komplexer als Taxonomien, deshalb für eine Navigation auch ungeeignet. Aus diesem Grund schlagen die Autoren vor, nach Möglichkeit sowohl eine Taxonomie für die Navigation, als auch eine Ontologie für komplexe, inhaltliche Zusammenhänge wie etwa transitive Beziehungen zwischen Dokumenten zu nutzen.

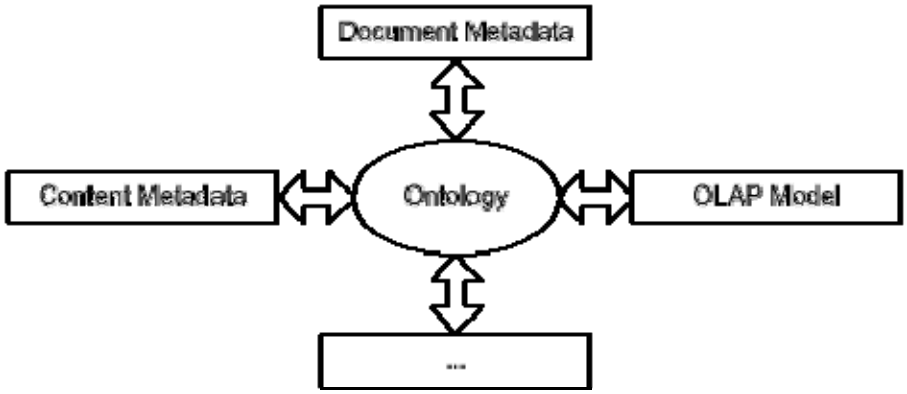


Abbildung 14: Ontologie für Modell-Mapping

Hier bietet sich gemäß Priebe und Pernul nun eine Lösung, um die Heterogenität der Portlets und der dahinter liegenden Systeme in den Griff zu bekommen. Mit Hilfe einer OWL Ontologie und ontologischen Konzeptmappings sowohl auf Schema- als auch auf Instanz-Ebene ist es möglich, unterschiedliche Repräsentationen von Kontext zueinander in Beziehung zu setzen bzw. aufeinander abzubilden. Abbildung 14 zeigt, wie eine Ontologie das Mappen verschiedener Metadaten-Modelle unterstützen kann. Sie ist als globales Vokabular zwischengelagert und kann die Metadaten der unterschiedlichen Portlets aufeinander abbilden.[Pr04]

4.3 Das Information Retrieval Modell

Im Folgenden wird nun kurz das im Prototyp INWISS verwendete Information Retrieval Modell vorgestellt. Ziel des Modells ist es, ein Ranking von relevanten Dokumenten zu erstellen, in dem die relevantesten Dokumenten zuerst, weniger relevante Dokumente erst später angezeigt werden. Da die Technologie des Semantic Web darauf ausgelegt ist exakte, maschinenlesbare Ergebnisse zu liefern, stellt sich hier die Aufgabe aus eindeutigen Daten und einer konkreten Anfrage ein fuzzy Ergebnis zu erzeugen, um so ein Ranking zu ermöglichen. Im vorliegenden Fall handelt es sich bei der Anfrage um den mit Hilfe der Taxonomie oder der Ontologie vereinheitlichten und für alle Portlets verständlich repräsentierten Kontext. Dieser wird an die anderen Portlets beziehungsweise an den Context Bus kommuniziert und anschließend genutzt, um aus allen vorhandenen Dokumenten die relevanten zu ermitteln. Das Ergebnis der Suche wird dann in den Portlets angezeigt, die ihren Kontext dem des ersten Portlet anpassen. In [PSP04] stellen die Autoren ein Ähnlichkeitsmaß vor, mit dessen Hilfe ein solches Ranking gemäß der Relevanz des jeweiligen Dokumentes erstellt werden kann.

```
# Sample resources
<Resource1> dc:title "The Freeplay(TM) Solar Radio" .
<Resource1> dc:creator <Tina> .
<Resource1> dc:coverage <FreeplaySolarRadio> .

<Resource2> dc:title "New Calling Center opened in Atlanta, GA"
<Resource2> dc:creator <Calvin> .
<Resource2> dc:coverage <CC_Atlanta> .

<Resource3> dc:title "Audio Sales 1998"
<Resource3> dc:creator <Ted> .
<Resource3> dc:coverage <FreeplaySolarRadio> .
<Resource3> dc:coverage <MicroFMRadio> .
<Resource3> dc:coverage <ShowerCompanion> .

<Resource4> dc:title "Electronics Sales 1998"
<Resource4> dc:creator <Ted> .
<Resource4> dc:coverage <Audio> .
<Resource4> dc:coverage <Comfort> .
<Resource4> dc:coverage <Gadgets> .

# Sample queries
<Query1> dc:coverage <FreeplaySolarRadio> .

<Query2> dc:creator <Calvin> .

<Query3> dc:coverage <FreeplaySolarRadio> .
<Query3> dc:coverage <MicroFMRadio> .
<Query3> dc:coverage <ShowerCompanion> .

<Query4> dc:coverage <Audio> .
<Query4> dc:coverage <Comfort> .
<Query4> dc:coverage <Gadgets> .
```

Abbildung 15: Beispiel-Metadaten in N3 Notation

Hierzu haben Priebe und Pernul zuerst ein vorläufiges Ähnlichkeitsmaß entwickelt und dieses dann verfeinert. Ausgehend von den beispielhaften Metadaten in Abbildung 15 soll dieses Verfahrens im Folgenden aufgezeigt werden.

Das verwendete Ähnlichkeitsmaß soll die Distanz zwischen zwei Dokumenten (der Anfrage und der Ressource) ausdrücken. In einem ersten naiven Ansatz wurde die Ähnlichkeit zwischen Dokumenten aus der Anzahl der property-value-Paare, die in beiden Dokumenten vorkommen, relativ zur Gesamtanzahl der Paare aus beiden Dokumenten errechnet.

Formal lässt sich die Ähnlichkeit zwischen dem Dokument und der Anfrage wie folgt berechnen:

$$sim(r_{doc}, r_{query}) = \frac{|D'(r_{doc}) \cap D'(r_{query})|}{|D'(r_{doc}) \cup D'(r_{query})|}$$

Abbildung 16: Berechnung des vorläufigen Ähnlichkeitsmaßes

Dabei ist D(r) die Beschreibung der Ressource und D'(r) die Menge der property-value-Paare von D(r).

Auf Basis der Metadaten aus Abbildung 15 ergibt sich somit folgende Ähnlichkeitsmatrix:

	<Resource1>	<Resource2>	<Resource3>	<Resource4>
<Query1>	0.3333	0	0.2	0
<Query2>	0	0.3333	0	0
<Query3>	0.2	0	0.6	0
<Query4>	0	0	0	0.6

Abbildung 17: Ähnlichkeitsmatrix bei Berechnung mit vorläufigem Ähnlichkeitsmaß

Für die Anfrage <Query1> würden so die relevanten Dokumente <Resource1> und <Resource3> gefunden. Dabei steht jedoch <Resource1> im Ranking höher, da dieses Dokument nur von Freeplay Solar Radio und nicht wie <Resource3> von weiteren Produkten handelt.

Diese Vorgehensweise zieht jedoch die Nutzung von Ontologien noch nicht in Betracht. Mit Hilfe der Ontologie aus Abbildung 18 sollen nun auch indirekte Beziehungen zwischen Metadaten in die Suche nach relevanten Dokumenten mit einfließen.

```
<Calvin> rdf:type <Manager> .
<Calvin> worksIn <CC_Atlanta> .
<Calvin> worksFor <Sales> .

<Tina> rdf:type <Employee> .
<Tina> worksIn <Headquarters> .
<Tina> worksFor <Marketing> .

<Ted> rdf:type <Employee> .
<Ted> worksIn <Headquarters> .
<Ted> worksFor <Finance> .

<CC_Atlanta> rdf:type <CallingCenter> .
<CC_Atlanta> locatedIn <Atlanta> .

<Headquarters> rdf:type <Office> .
<Headquarters> locatedIn <NewYork> .

<Atlanta> rdf:type <City> .
<Atlanta> belongsToRegion <SouthEast> .

<NewYork> rdf:type <City> .
<NewYork> belongsToRegion <NorthEast> .

<FreeplaySolarRadio> rdf:type <Item> .
<FreeplaySolarRadio> belongsToSubcategory <Audio> .

<NicroFMRadio> rdf:type <Item>
<NicroFMRadio> belongsToSubcategory <Audio> .

<ShowerCompanion> rdf:type <Item>
<ShowerCompanion> belongsToSubcategory <Audio> .

<Audio> rdf:type <Subcategory> .
<Audio> belongsToCategory <Electronics> .
```

Abbildung 18: Auszug aus der Ontologie

Für <Resource1> ergibt sich aus dieser Ontologie der Graph aus Abbildung 19. Hier werden transitive Beziehungen deutlich. <Resource1> wurde von Tina erstellt, die wiederum in den Headquarters in New York arbeitet, was sich in der Verkaufsregion North East befindet.

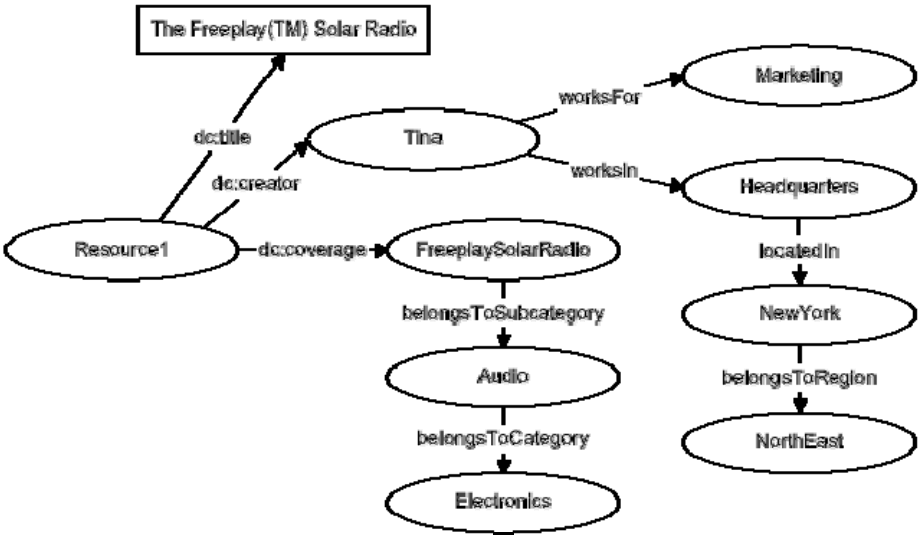


Abbildung 19: Beispiel-Metadaten als RDF-Graph

Mit Hilfe der Inferenzmaschine können solche implizit vorhandenen Informationen automatisch generiert werden. Anschließend werden die Metadaten um die so erzeugten zusätzlichen Daten erweitert. Für <Resource1> ergibt sich so die in Abbildung 20 dargestellte Beschreibung.

```
<Resource1> dc:title "The Freeplay(TM) Solar Radio" .
<Resource1> dc:creator <Tina> .
<Resource1> dc:creator <Marketing> .
<Resource1> dc:creator <Headquarters> .
<Resource1> dc:coverage <FreeplaySolarRadio> .
<Resource1> dc:coverage <Audio> .
<Resource1> dc:coverage <Electronics> .

<Query1> dc:coverage <FreeplaySolarRadio> .
<Query1> dc:coverage <Audio> .
<Query1> dc:coverage <Electronics> .
```

Abbildung 20: Metadaten mit Inferenz-Tripeln

Das erste Ähnlichkeitsmaß hat zwei Nachteile, die unter Einbeziehung der Ontologie sehr deutlich werden. Ressourcen mit vielen Metadaten wie aus der Formel in Abbildung 16 ersichtlich werden bestraft im Vergleich zu solchen mit wenigen Metadaten, da sich für sie ein höherer Nenner ergibt. Eigenschaften mit vielen Werten dominieren gegenüber Eigenschaften mit wenigen Werten, da eine Eigenschaft, die mehrwertig vorkommt, einen höheren Anteil am Gesamtgewicht des Dokuments erhält. Deshalb haben Priebe und Pernul ihr Ähnlichkeitsmaß verfeinert, indem zum einen die Ähnlichkeiten für jede Eigenschaft einzeln berechnet werden. Zum Anderen werden nur die Eigenschaften betrachtet, die in der Anfrage vorkommen, da es wenig Sinn macht, Metadaten zum Autor in das Ergebnis mit einzubeziehen, wenn nur nach dem Element *Coverage* gefragt wird.

Daraus ergibt sich nun ein verfeinertes Ähnlichkeitsmaß:

$$match(r_{doc}, r_{query}) = \sum_{p_j \in P(r_{query})} \frac{1}{|P(r_{query})|} \frac{\left| D'_{p_j}(r_{doc}) \cap D'_{p_j}(r_{query}) \right|}{\left| D'_{p_j}(r_{doc}) \cup D'_{p_j}(r_{query}) \right|}$$

Abbildung 21: Berechnung des verbesserten Ähnlichkeitsmaßes

Hier ist P(r) die Menge der Eigenschaften in D(r) und D'_{p_j} die Menge der property-value-Paare von D'(r) mit der Eigenschaft p_j.

Auf Basis der um transitive Beziehungen erweiterten Metadaten aus Abbildung 20 ergibt sich somit folgenden Ähnlichkeitsmatrix:

	<Resource1>	<Resource2>	<Resource3>	<Resource4>
<Query1>	1	0	0.6	0.4
<Query2>	0	1	0	1
<Query3>	0.6	0	1	0.2857
<Query4>	0.4	0	0.2857	1

Abbildung 22: Ähnlichkeitsmatrix bei Berechnung mit verbessertem Ähnlichkeitsmaß

Es zeigt sich, dass mit Hilfe des verfeinerten Ähnlichkeitsmaßes weitere Dokumente gefunden werden, die implizit in Beziehung zur Anfrage stehen. So wird auf Anfrage <Query1> neben <Resource1> und <Resource3>, die von dem Produkt Freeplay Solar Radio handeln, auch <Resource4> gefunden, in der von der übergeordneten Kategorie Audio die Rede ist, zu der das Produkt Freeplay Solar Radio zählt. Dieses Dokument erscheint im Ranking jedoch deutlich später als <Resource1>, die sich ausschließlich mit diesem Produkt beschäftigt.

Aufgrund dieser Berechnung des Ähnlichkeitsmaßes ergibt sich der Algorithmus in Abbildung 23.

```
add query triples to repository and do inferencing
retrieve all query triples from repository
query properties := distinct properties from query triples
property count := number of query properties

start with query to retrieve all triples
for each query triple t do {
  add where clause for subject and object of t
}
retrieve query results as candidate triples
candidate resources := distinct resources from candidate triples

for each candidate resource r do {
  match value := 0

  for each query property p do {
    match count := 0
    mismatch count := 0
    remaining candidate property values := all values for resource r with property p

    for each query triple t with property p do {
      if corresponding triple exists for r {
        increment match count
        remove object of t from remaining candidate property values
      } else increment mismatch count
    }

    add number of remaining candidate property values to mismatch count
    add (match count / (property count * (match count + mismatch count))) to match value
  }

  memorize match value for r
}

remove query triples from repository
return candidate resources with match values
```

Abbildung 23: Matching-Algorithmus

Das hier vorgestellt Information Retrieval Modell wird im Prototyp INWISS eingesetzt, der in Abschnitt 4.5 genauer erläutert wird.

4.4 Architektur eines Portalsystems

Im Anschluss soll die Architektur von Portalen im Allgemeinen vorgestellt und im nächsten Kapitel die entsprechenden Erweiterungen im von Priebe und Pernul erstellten Prototypen INWISS erläutert werden. Der Begriff Portal bezieht sich hier auf unternehmensweite Wissensportale, so genannte Enterprise Information Portals. Ein solches Portal soll dem Nutzer über eine personalisierte Oberfläche jederzeit Zugang zu allen für ihn wichtigen Informationen bieten. In Unternehmen kann er so über die einheitliche Schnittstelle des Portals auf Transaktionen von Anwendungssystemen und Web Services, auf OLAP-Daten oder einfache Webdokumenten zugreifen. [PrPK03]

Ein Portalsystem gliedert sich wie Abbildung 24 zeigt in drei Schichten.

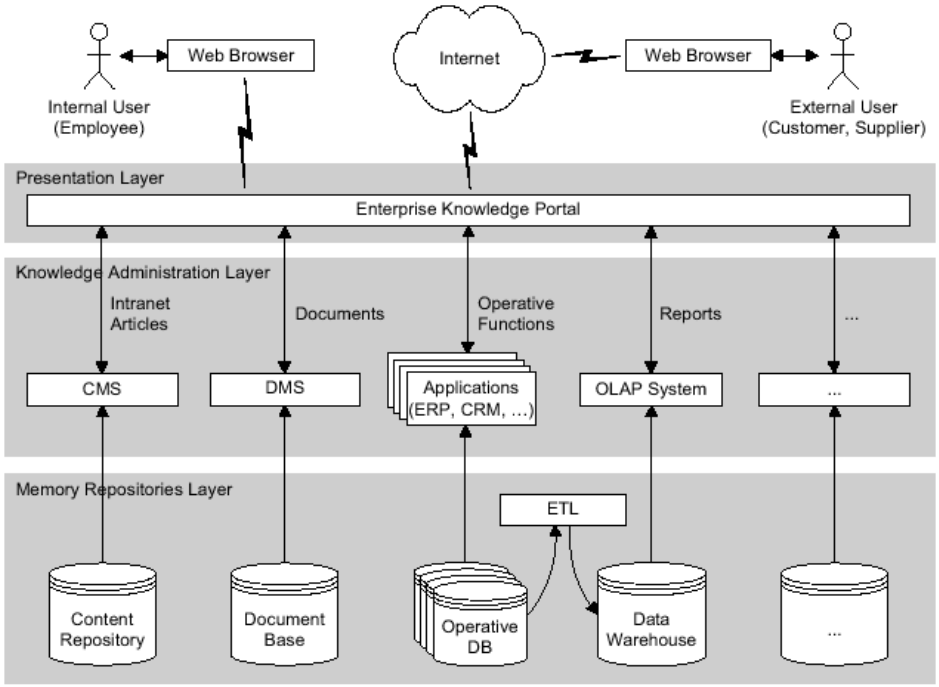


Abbildung 24: Architektur eines Portalsystems

In der ersten Schicht, dem Memory Repositories Layer, sind die Daten beispielsweise in einer Dokumenten-Datenbank, operativen Datenbanken und einem Data Warehouse hinterlegt.

Die darüber liegende Schicht, der Knowledge Administration Layer, enthält die Software-Komponenten, die den Zugriff auf die jeweilige Datenquelle und die Auswertung der Daten erlauben. Hier ist etwa ein OLAP-System angesiedelt, dass den Zugriff auf ein Data Warehouse ermöglicht, sowie Anwendungen wie ERP- oder CRM-Systeme, die mit operativen Datenbanken verknüpft sind. In diesem Zusammenhang könnte man sich auch ein Information Retrieval System vorstellen, dass sowohl in unternehmensinternen Datenbanken als auch in unternehmensexternen Datenbanken oder im Internet nach relevanten Dokumenten sucht.

Die Architektur eines Portalssystems beinhaltet als dritte Schicht den Presentation Layer. Hierbei handelt es sich um das eigentliche Portal, dass Informationen über eine einheitliche Oberfläche an den Nutzer weitergibt. Der User hat über die verschiedenen Portlets Zugriff auf die unterschiedlichen Systeme und deren Daten.

Nutzer eines Enterprise Knowledge Portal sind im Regelfall vor allem interne Nutzer, also Beschäftigte des jeweiligen Unternehmens, die sich direkt in das Portal einwählen. Aber auch externe Nutzer wie Lieferanten und Kunden, die über das Internet Zugang zum Portal erhalten, sind denkbar. [Pr04b]

4.5 Der Prototyp INWISS

Der größte Nachteil von Portalen liegt nach Priebe und Pernul vor allem in der fehlenden Interaktion der einzelnen Portlets untereinander. Die Portlets werden zwar gemeinsam auf einer Webseite angezeigt, ihr Inhalt aber ist genau wie die dahinter liegenden Systeme und deren Daten getrennt. Wenn ein Nutzer in einem Portlet nach Informationen sucht, so bleibt der Inhalt der anderen Portlets unverändert. In ihrem Prototypen INWISS haben Priebe und Pernul ihr in Abschnitt 4.1 beschriebenes Konzept der kontextbasierten Integration von Portlets mit Hilfe eines so genannten Context Bus umgesetzt. Dieser soll im diesem Abschnitt näher beschrieben werden.

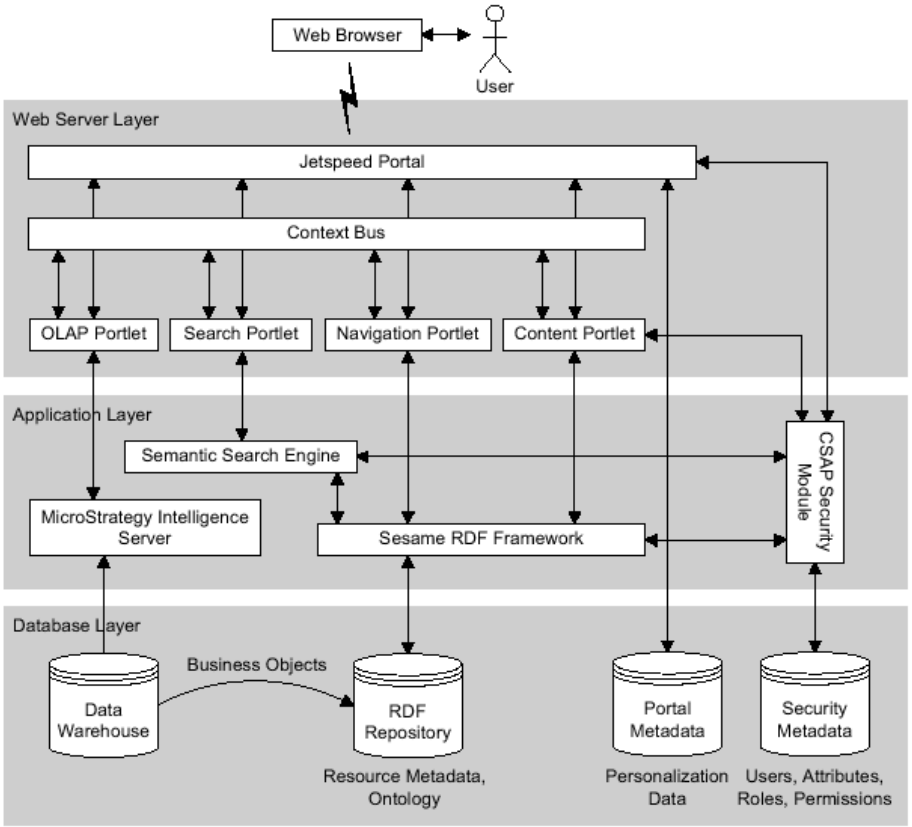


Abbildung 25: Architektur des INWISS Prototypen

Abbildung 25 zeigt die Architektur des Prototypen. Die Daten selbst liegen in Form eines Data Warehouse oder als Metadaten im RDF Repository auf dem Database Layer vor. Während die Autoren für Content und Document Management Systeme davon ausgehen, dass diese nutzbare Schnittstellen zum RDF Repository besitzen, muss für OLAP Metadaten eine eigene Integration erfolgen. Die OLAP Dimensionen werden hier zur Generierung von Business Objects genutzt, die wiederum in die Ontologie eingehen.

Die von den Portlets verwendeten Anwendungen befinden sich im darüber liegenden Application Layer. In dem auf dieser Ebene angesiedelten Sesame RDF Framework sind die Metadaten, die Taxonomie und die Ontologie enthalten, die von der Suchmaschine genutzt werden. Das CSAP Security Module ist für die Authentifizierung und den Zugang zum Portal verantwortlich. Dies zeigt, dass der Sicherheitsaspekt, der im Zusammenhang mit Portalen von hoher Bedeutung ist, im Prototypen ebenfalls berücksichtigt wurde.

Der Web Server Layer beinhaltet die gesamte Architektur der Kontext-Integration mit den einzelnen Portlets, dem sie verbindenden Context Bus und der eigentlichen Zusammenführung im Portal. Der Context Bus ist als Erweiterung der Apache Jetspeed Portal Plattform implementiert und erlaubt eine kontextbasierte Integration von derzeit vier verfügbaren Portlets. Beim ersten dieser vier Portlets handelt es sich um ein OLAP Portlet, das den Zugriff auch das dahinter liegende MicroStrategy 7i OLAP System ermöglicht. Weiterhin ist ein Search Portlet nutzbar, über das eine metadaten-basierte Suche erlaubt. Im Navigation Portlet ist ein taxonomie-basierten Browsing der unterschiedlichen Themengebiete möglich und im Content Portlet werden konkrete Inhalte wie etwa Artikel aus dem Intranet dargestellt.

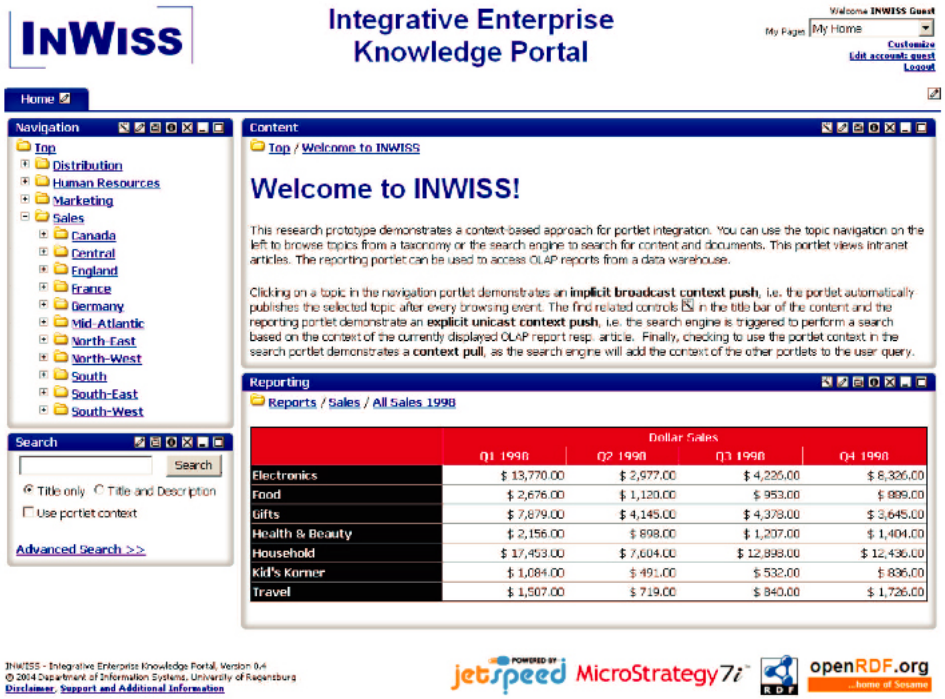


Abbildung 26: Screenshot des INWISS Prototypen

Abbildung 26 zeigt einen Screenshot des Portals, Abbildung 27 eine genauere Ansicht des Search Portlet. Die Nutzung des umgesetzten Konzeptes ist über einen „Find related"-Button möglich. Generiert ein Nutzer nun einen OLAP Bericht, so verursacht ein Klick auf den „Find related"-Button die Generierung einer RDF-Beschreibung des Kontext. Das Search Portlet nutzt diese Information um eine Ähnlichkeitssuche durchzuführen mit dem Ziel verwandte Ressourcen wie Dokumente oder Artikel zu finden. [Pr04b]

Bei den Portlets handelt es sich um Java Klassen, die das Jetspeed Interface Portlet implementierten. Der Lebenszyklus eines jeden Portlet besteht aus drei Phasen, Init, Render und Destroy. In der Init-Phase wird mit Hilfe der `init()`-Methode ein Portlet-Objekt erzeugt. Solange es sich im Cache befindet, bleibt es aktiv und kann somit genutzt werden. In der darauf folgenden Render-Phase wird bei jeder Änderung des Inhalts des Portlet die Methode `getContent()` aufgerufen und das Portlet entsprechend neu aufgebaut. In der abschließenden Destroy-Phase wird das Objekt zerstört, also aus dem aus dem Cache gelöscht[57].

[57] . [Apa04] zitiert in [Pr04]

Die Kommunikation der Portlets untereinander wurde über ein Redirect realisiert, das den Rendering-Zyklus erneut startet, da ein Portlet andernfalls nur dann erneut gerendert werden würde, wenn sich sein eigener Kontext geändert hat. Zur Umsetzung des Context Bus wurde das gemeinsame `RunData`-Objekt genutzt, das in den `context`-Variablen die aktuellen Kontext-Stati der einzelnen Portlets enthält. Diese Implementierung erlaubt es Portlets einerseits, den Kontext anderer Portlets nachzufragen (pull), aber auch den eigenen Kontext zu publizieren (push). [Pr04]

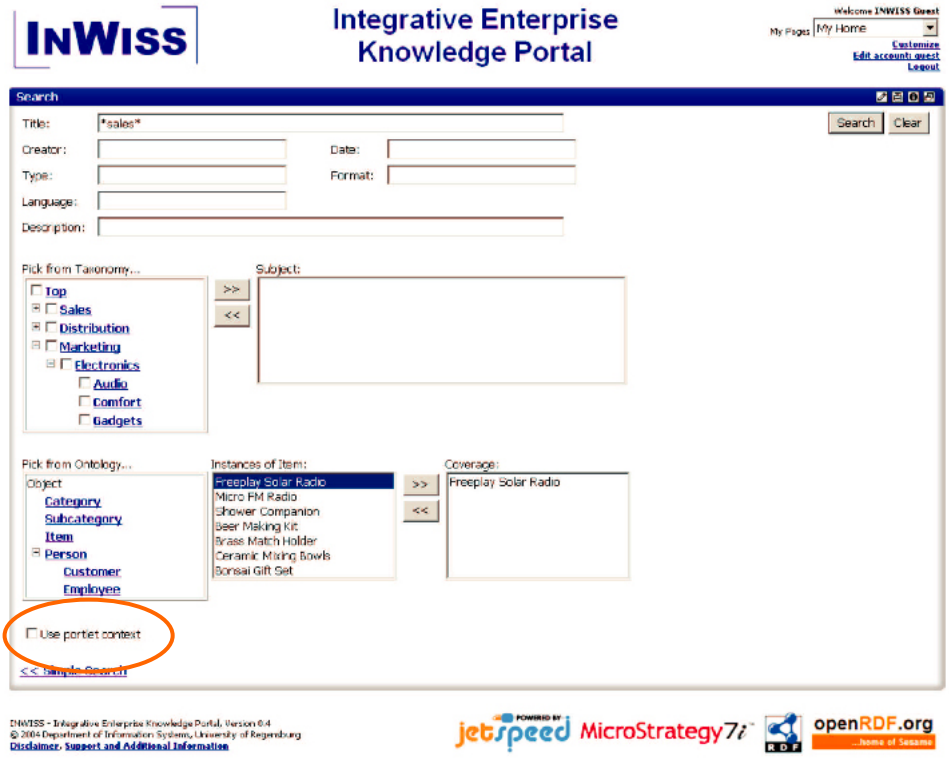


Abbildung 27: Screenshot des Search Portlet

5 Zusammenfassung und Ausblick

Diese Arbeit betrachtete die Integration von kontextbasiertem Information Retrieval in Enterprise Information Portals. Sinn einer solchen Integration ist es, den Nutzer eines Portals bei seiner Suche nach relevanten Informationen gezielt zu unterstützen. So soll ihm ein schneller Zugriff auf die richtigen Informationen ermöglicht werden, was in der heutigen Zeit von großer Bedeutung ist. Während sich der erste Teil der Arbeit vor allem mit den Grundlagen von Portalen und den verwendeten Technologien wie RDF und Ontologien auseinandersetzte, beschrieb der zweite Teil das Konzept von Torsten Priebe und Günther Pernul in seinen Einzelheiten.

Im Gegensatz zu bisher existierenden Lösungen ist es Ziel des Konzeptes von Priebe und Pernul eine automatische Anpassung der Portlets an den sich dynamisch ändernden Benutzerkontext zu ermöglichen. Ein Hauptaugenmerk lag somit auf der Kommunikation des Benutzerkontext über die Grenzen einzelner Portlets hinweg. Anders als in bisherigen „fest verdrahteten“ Spezialanwendungen sollte es sich dabei um eine generische Lösung handeln, die in Standardplattformen integriert werden kann. Die Autoren schlagen hierzu die Nutzung eines Context Bus vor, der Nachrichten über Kontextänderungen in einzelnen Portlets erhält, die wiederum von anderen Portlets abgegriffen und zur Anpassung genutzt werden können.

Der Ansatz von Priebe und Pernul verspricht eine hohe Flexibilität, muss sich jedoch mit der Heterogenität von so verschiedenen Informationsquellen wie Data Warehouses, Dokumenten Management Systemen oder anderen operativen Datenbanken auseinandersetzen. Daraus ergibt sich die Forderung nach einer einheitlichen, für alle Portlets verständlichen Repräsentation des Kontext. Zu diesem Zweck wurde die Nutzung einer Ontologie vorgeschlagen, die eine Abbildung der unterschiedlichen Metadaten-Modelle aufeinander ermöglicht. Ein News Portlet könnte dann anhand des auf dem Context Bus vorliegenden Kontext eines OLAP Portlet eine Ähnlichkeitssuche in den mit ihm verknüpften Dokumenten-Sammlungen starten und entsprechende Ergebnisse anzeigen. Zur Informationssuche haben die Autoren außerdem ein eigenes Information Retrieval Modell entwickelt, welches die Ähnlichkeit des Kontext mit hinterlegten Ressourcen errechnet.

In ihrem Prototypen INWISS haben Priebe und Pernul schließlich ihr Konzept der kontextbasierten Integration in einem Portal umgesetzt. Dabei erfolgte eine Konzentration auf unternehmensinterne Informationsquellen wie etwa ein Data Warehouse.

Bei der Umsetzung ergab sich auch die Frage nach notwendigen Sicherheitsmechanismen, die Gegenstand weiterer Forschung sein werden. Ebenso ist die Integration weiterer Informationsquellen, sowie das für Portalsysteme typische Konzept der Personalisierung in zukünftigen Arbeiten genauer zu betrachten. [PrPK03, Pr04b]

Literaturverzeichnis

[Apa04] The Jetspeed Portal Tutorial 1.5. Apache Portals project website, 2004. http://portals.apache.org/jetspeed/tutorial/. Visited August 2004.

[AR02] Alexander Roth: Modellierung und Anwendung von Ontologien http://dsor.upb.de/~aroth/rdfdb/RDFOntologien.pdf, Stand 12.07.2002

[BL01] T.Berners-Lee; J.Handler; O.Lassila: The Semantic Web, Scientific American, 2001

[DCM03] Dublin Core Metadata Initiative. DCMI Metadata Terms. DCMI recommendation, November 2003. http://dublincore.org/documents/2003/11/19/dcmi-terms/.

[HM02] Henrich, A.; Morgenroth, K.: Integration von kontextunterstütztem Information Retrieval in Portalsysteme. http://ai1.inf.uni-bayreuth.de/forschung/publikationsliste/2002/MKWI_2002_Henrich_Morgenroth.pdf, Stand 28.06.2004.

[JR01] Jochen Rütschlin: „Was ist ein Portal?“
Erschienen in: Kurt Bauknecht, Wilfried Brauer und Thomas Mück (Herausgeber). Informatik 2001: Wirtschaft und Wissenschaft in der Network Economy – Visionen und Wirklichkeit. Tagungsband der GI/OCG-Jahrestagung, 25.-28. September 2001, Universität Wien. Seiten 691-696. ISBN 3-85403-157-2

[OF01] Sven Offermann: Entwurf und Realisierung einer generischen Portalplattform und prototypischer Komponenten für das PublicationPORTAL
http://vsis-www.informatik.uni-hamburg.de/publications/readstu.phtml/SA/99/Offermann2001-SA-Portal.pdf, Stand 2001

[PrPK03] Priebe, T.; Pernul, G.; Krause, P.: Ein integrativer Ansatz für unternehmensweite Wissensportale. http://www-ifs.uni-regensburg.de/PDF_Publikationen/PrPK03.pdf, 2003, Stand 11.06.2004.

[Pr04] Priebe, T.: Context-based Portlet Integration. http://www-ifs.uni-regensburg.de/inwiss/whitepapers/ContextIntegration.pdf, 26. August 2004, Stand 30.08.2004.

[Pr04b] Priebe, T.: INWISS - Integrative Enterprise Knowledge Portal. http://www-ifs.uni-regensburg.de/inwiss/whitepapers/Overview.pdf, 26.August 2004, Stand 30.08.2004.

[PSP04] Priebe, T.;Schläger, C.; Pernul, G.: A Similarity-based Information Retrieval Model for the Semantic Web. Submitted for Publication, 29. März 2004.

[PSP04b] Priebe, T.;Schläger, C.; Pernul, G.: A Search Engine for RDF Metadata. http://www-ifs.uni-regensburg.de/PDF_Publikationen/PrSP04.pdf, 26. August 2004, Stand 30.08.2004.

[UK04] Uwe Krüger: Semantic Web – Eine Wegbeschreibung zum Web 2.0 http://www.minet.uni-jena.de/~sack/ WS0304/seminar/arbeiten/kr%FCger/ausarb.pdf, Januar 2004

ISBN: 3-00-015171-0