Universität Hamburg
Fakultät für Mathematik,
Informatik und Naturwissenschaften

Verteilte Systeme und Informationssysteme

Diplomarbeit

Intelligente Agenten zur Dokumentenrecherche im World Wide Web

Afiriyie Adwiraah
7adwiraa@informatik.uni-hamburg.de
Studentin der Informatik (Diplom)
Matr.-Nr. 5001443

Erstgutachter: Prof. Dr. Winfried Lamersdorf
Zweitgutachter: Prof. Dr. Christopher Habel

Erklärung

Ich versichere, daß ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Außerdem erkläre ich, daß ich mit der Einstellung dieser Diplomarbeit in den Bestand der Bibliotheken der Universität Hamburg einverstanden bin.

Hamburg, 10.05.2006

Inhaltsverzeichnis

1. Einleitung und Gliederung

Das World Wide Web ist neben den traditionellen Printpublikationen für die wissenschaftliche Informationssuche zu einer wichtigen Quelle geworden.
Auf den Webseiten vieler Wissenschaftler finden sich mittlerweile neben Informationen über ihre Person und ihre Projekte zunehmend auch Veröffentlichungen ihrer Forschungsarbeiten. Diese werden meist in Form von PDF-Dokumenten frei zugänglich zur Verfügung gestellt.

Ein großer Vorteil dieser Veröffentlichungen im Web ist die hohe Aktualität der Publikationen. Während zwischen der Fertigstellung einer Arbeit und der Veröffentlichung bei einem Verlag mitunter mehrere Monate vergehen können, sind auf den Webseiten der Autoren oft schon Vorveröffentlichungen zu finden.

Diese Publikationen im WWW ausfindig zu machen, ist jedoch nicht immer einfach. Viele der Publikationen lassen sich nicht über herkömmliche Suchmaschinen finden, da diese aufgrund der Größe des Webs nicht alle Webseiten und Dokumente in ihrem Index aufnehmen können. Online-Publikationen sind zudem ständigen Veränderungen unterworfen. Online-Dokumente lassen sich im Vergleich zu Printdokumenten schnell und einfach überarbeiten und veröffentlichen. Nachteilig für einen Informationssuchenden ist die dadurch entstehende hohe Flüchtigkeit der Informationen im WWW. Dies bedeutet, daß sich ein Dokument, das sich heute noch unter einer bestimmten URL auffinden ließ, morgen schon nicht mehr vorhanden sein kann.

Um einmal gefundene Publikationen trotzdem auf Dauer verfügbar zu halten, ist es notwendig, sich nicht nur die URL einer Publikation zu merken, sondern das Dokument als solches herunterzuladen und lokal abzulegen.
Je nach Anzahl der so gesammelten Dokumente kann dieses Vorgehen schnell zu einer großen und unübersichtlichen Menge an ungeordneten Daten führen. Ordnung in diese Daten zu bringen, erfordert ein großes Maß an Disziplin beim Speichern und Dokumentieren von gefundenen Online-Dokumenten und ebenso viel Zeitaufwand.

Ein weiteres Problem bei der Suche nach wissenschaftlichen Publikationen im WWW ist der Einsatz von Keyword-basierten Suchmaschinen. Hier ist es nötig, eine präzise Suchanfrage, bestehend aus nur wenigen Schlüsselworten, zu formulieren, um die gewünschten Informationen zu finden. Einfacher wäre eine auf Dokumentenähnlichkeit basierende Suche, die ausgehend von einem bei dem Suchenden bereits lokal vorhandenen Dokument andere ähnliche Dokumente findet. Eine solche Suche ist mit herkömmlichen Suchmaschinen derzeit nicht möglich. Zwar bieten viele Suchmaschinen auf ihren Ergebnisseiten die Option „nach Ähnlichen Dokumenten suchen“ an, jedoch müssen sich hierbei sowohl die gesuchten Dokumente als auch das Ausgangsdokument im Index der Suchmaschine befinden.

Aufgrund der beschriebenen Probleme bei der Suche im WWW soll im Rahmen dieser Arbeit ein System entworfen werden, welches die Recherche nach wissenschaftlichen Publikationen im Web unterstützt. Unterstützung soll zum einen durch eine Automatisierung des Suchvorganges realisiert werden. Hierfür wird eine intelligente Software benötigt, die in der Lage ist das WWW selbständig zu durchsuchen. Weiterhin sollen Informationssuchende auch über den reinen Suchvorgang hinaus bei der

Archivierung der gefundenen Publikationen unterstützt werden. Im WWW gefundene Dokumente sollen daher ihrem Inhalt nach klassifiziert und nach Themen geordnet lokal gespeichert werden.

Das entwickelte System wird im Rahmen dieser Arbeit unter dem Namen *DocAssistant* prototypisch umgesetzt. Für die Implementation des Prototyps wird das Jadex Agenten-Framework verwendet. Jadex wurde an der Universität Hamburg entwickelt und basiert auf dem BDI-Modell, welches sich an dem Prozeß der menschlichen Entscheidungsfindung und des menschlichen Handelns in Form von mentalen Konzepten (Beliefs, Desires und Intentions) orientiert.
Vorbild für die Architektur des Systems ist das Multiagentensystem MySpiders, welches auf einem evolutionären Ansatz beruht und in der Lage ist, das WWW in Echtzeit zu durchsuchen. Um Dokumente bei der Suche auf ihre Relevanz hin prüfen zu können, wird Latent Semantic Indexing (LSI), eine Technik des maschinellen Lernens, eingesetzt. LSI wird ebenso für die anschließende Klassifikation der im Web gefundenen Dokumente verwendet.

Für die Konzeption von DocAssistant wird in Kapitel 2 zunächst der Anwendungsbereich des zu entwickelnden Systems untersucht. Hierfür werden die Grundlagen der Informationssuche und der Wissensgewinnung im World Wide Web erörtert. Anschließend werden die Einsatzmöglichkeiten von intelligenten Agenten insbesondere für die Suche im Web dargestellt.
Kapitel 3 führt die Methode der Klassifizierung von Daten ein und erläutert anschließend das Vorgehen für die Klassifizierung von Webdokumenten. Es folgt eine vergleichende Vorstellung verschiedener Klassifikationstechniken. Genauer wird hier auf die Technik Latent Semantic Indexing eingegangen, da sie für das in dieser Arbeit entwickelte System DocAssistant eingesetzt wird.
In Kapitel 4 erfolgt eine Betrachtung bestehender System und Ansätze für die Verbesserung der Informationssuche im World Wide Web. Hier werden besonders die Systeme MySpiders und MELISSA hervorgehoben, da diesen eine besondere Bedeutung für die Entwicklung von DocAssistant zukommt.
Kapitel 5 erläutert die Konzeption und Umsetzung des Prototyps DocAssistant. Anschließend erfolgt eine Evaluierung und Bewertung des Systems.
In Kapitel 6 wird abschließend eine Zusammenfassung der Ergebnisse dieser Arbeit und ein kurzer Ausblick über zukünftige Entwicklungen und Verbesserungen von DocAssistant gegeben.
Angaben zur verwendeten Literatur erfolgen in Kapitel 7.

2. Grundlagen

In diesem Kapitel sollen die Grundlagen, die für die vorliegende Arbeit von Bedeutung sind, ermittelt und untersucht werden.
Hierzu gehört zunächst eine Erörterung der für diese Arbeit grundlegenden Begriffe des Data-Mining bzw. Web-Mining sowie die Darstellung der dazugehörigen Prozesse der Wissensgewinnung.
Anschließend sollen Methoden für die Recherche nach Dokumenten im World Wide Web und Probleme, die sich hierbei ergeben, erläutert werden. Speziell auf die Kriterien, die für wissenschaftliche Informationssuchende bei der Recherche relevant sind, soll eingegangen werden.
Grundlage für das im Rahmen dieser Arbeit entwickelte System ist der Einsatz von intelligenten Agenten. Daher widmet sich der letzte Teil dieses Kapitels der Untersuchung des Agentenparadigmas.

2.1. Wissensgewinnung im World Wide Web

Die Suche im WWW nach Informationen wird häufig auch als Web-Mining bezeichnet. Dieser Ausdruck leitet sich von dem aus dem Bereich der Datenbanken stammenden Begriff des Data-Mining ab.
In diesem Kapitel werden daher die für diese Arbeit grundlegenden Vorgänge des Data- bzw. Web-Mining erläutert. Hierfür wird zunächst der Prozeß des Knowledge Discovery in Databases vorgestellt, der als Teilschritt das Data-Mining beinhaltet. Dieser Prozeß der Wissensgewinnung wird anhand seiner einzelnen Schritte erörtert, um die ermittelten Vorgehensweisen anschließend auf das World Wide Web zu übertragen.
Die Unterschiede zur Wissensgewinnung in Datenbanken werden herausgearbeitet, um schließlich analog zum Knowledge Discovery in Databases einen Prozeß des Knowledge Discovery on the World Wide Web vorzustellen.

2.1.1. Data-Mining und Knowledge Discovery in Databases

Der Begriff Data-Mining ist ein Sammelbegriff für verschiedene Techniken und Methoden, die dazu benutzt werden, verborgene Strukturen in großen Datenmengen aufzudecken. Durch Data-Mining sollen Muster (Patterns) innerhalb von Datenmengen gefunden werden, die eine zusammenfassende Aussage über die jeweilige Datenmenge treffen.
Es hat sich als sinnvoll erwiesen, Data-Mining nur als Teil eines gesamten Prozesses zu sehen, bei dem Wissen aus einer Menge von Daten gewonnen wird (Fayyad et al., 1996). Dieser Prozeß wird als Knowledge Discovery in Databases (KDD) bezeichnet.
Viele Autoren verwenden die Begriffe Data-Mining und KDD synonym (Adriaans und Zantinge, 1996). Fayyad et al. (1996) heben jedoch hervor, daß Data-Mining nur ein Schritt im KDD-Prozeß ist und die übrigen Schritte des Prozesses nicht außer Acht gelassen werden sollten. Diese seinen für die Wissensgewinnung mindestens ebenso wichtig. Reines Anwenden von Data-Mining-Methoden auf eine Menge von Daten kann beispielsweise zur Extraktion von unwichtigen oder im schlimmsten Fall sogar falschen Datenmustern führen.
Der gesamte Prozeß des KDD beinhaltet nicht nur das Data-Mining, es werden auch weitere Vorgehensweisen betrachtet. Diese beinhalten die Art und Weise wie die Daten

abgelegt und gespeichert werden, wie darauf zugegriffen wird, wie große Mengen von Daten effizient analysiert werden können, wie Ergebnisse interpretiert und visualisiert werden können und wie die Mensch-Maschine-Interaktion gestaltet und unterstützt werden kann. Der Data-Mining-Vorgang für sich gesehen ist dagegen nur die Anwendung spezieller Algorithmen, um Muster aus Daten zu extrahieren.

2.1.2. Knowledge Discovery in Databases

Knowledge Discovery in Databases ist ein Gebiet dessen Ursprünge in vielen verschiedenen verwandten Themengebieten zu finden sind. Daher stammen auch die im KDD verwendeten Techniken aus vielen verschiedenen Disziplinen. Zu diesen zählen beispielsweise Machine Learning, Pattern Recognition, Künstliche Intelligenz, Datenbanken, Statistik, Visualisierung von Daten, um nur einige zu nennen (Fayyad et al., 1996).
Im wissenschaftlichen Bereich kommen KDD Systeme vor allem in der Astronomie zum Einsatz. Hier fallen beispielsweise bei der Untersuchung des Weltalls große Mengen von Daten an, die mit Hilfe von KDD analysiert und interpretiert werden können. In der Wirtschaft wird KDD unter anderem für Marketing-Untersuchungen eingesetzt. Zum Beispiel, um aus einer Kundendatenbank Daten zu extrahieren, die genutzt werden können, um einzelne Kunden einer bestimmten Gruppe mit bestimmtem Konsumverhalten zuzuordnen.
Ebenso können intelligente Agenten zur Informationssuche im WWW, wie sie in dieser Arbeit vorkommen, als eine Anwendung des KDD betrachtet werden. Das WWW entspricht dabei einer Datenbank von enormen Ausmaßen.

Knowledge Discovery in Databases ist ein Prozeß, bei dem es darum geht, nützliche Informationen, bzw. Wissen aus einer großen Menge von rohen Daten zu extrahieren und in eine leichter verständliche oder leichter faßbare Form zu bringen. Es geht also grob gesagt darum Daten in Wissen zu verwandeln. Fayyad et al. (1996) definieren KDD folgendermaßen: „KDD is the nontrivial process of identifying valid, novel, potentially usefull, and ultimately understandable patterns in data. “ Ist die Menge der vorhandenen Daten sehr groß, kann sich eine manuelle Suche nach solchem Wissen als ein sehr langsamer und teurer Prozeß erweisen, oder sogar schlicht unmöglich sein. Desweiteren sind Ergebnisse durch manuelle Bearbeitung der Daten meist subjektiv geprägt. Daher versucht man durch KDD den Prozeß möglichst zu automatisieren.
Wie bereits oben erwähnt, besteht der KDD-Prozeß aus mehreren Schritten. Der Prozeß ist Anwender gesteuert, d.h. der Anwender steuert den Prozeß interaktiv und bestimmt die Abfolge der Schritte die während des Prozesses durchlaufen werden. Nach Fayyad et al. (1996) besteht der KDD-Prozeß aus folgenden Schritten:

1. **Untersuchung des Anwendungsbereiches** und Festlegen des Ziels des KDD-Prozesses.
2. **Auswahl der Daten,** die durchsucht werden sollen. Falls nur ein Teilbereich der vorliegenden Daten durchsucht werden soll, kann hier die Datenmenge bereits reduziert werden.
3. **Vorverarbeitung (Pre-processing)** der Daten. Die Daten werden bereinigt, d. h. fehlerhafte Daten werden entfernt oder berichtigt, eventuell fehlende Einträge werden ergänzt.
4. **Transformation** der Daten. Die Daten können durch Reduktion von Dimensionen oder andere Transformationsmethoden reduziert werden. Hier findet beispielsweise die Extraktion von Feature-Werten statt.

5. **Data-Mining.** Hier geschieht die Extraktion der Muster aus den Daten. Diese Phase besteht aus folgenden drei Teilschritten:
    a. Auswahl der Data-Mining Methode, z.B. Klassifizierung, Regression, Clustering.
    b. Auswahl des Data-Mining Algorithmus und Bildung des Modells, mit dem die Muster gefunden werden sollen.
    c. Eigentliches Data-Mining, d.h. Suche nach Mustern mit Hilfe des ausgewählten Algorithmus.
6. **Interpretation und Evaluation** der gewonnen Daten. Die extrahierten Muster werden dem Benutzer in einer leicht verständlichen und faßbaren Form dargestellt. Dies kann beispielsweise eine Visualisierung der Daten und den Einsatz von Filtern beinhalten. Eine Evaluierung der Muster kann ergeben, daß ein weiteres Durchlaufen einer oder mehrerer der vorherigen Schritte mit veränderten Parametern notwendig ist.
7. **Weiterverarbeitung des gewonnenen Wissens.** Der letzte Schritt beinhaltet die Anwendung oder Weiterverarbeitung des gewonnenen Wissens. Beispielsweise kann das Wissen in ein anderes System einfließen oder einfach dokumentiert und so an Interessenten vermittelt werden.

Wie bereits erwähnt, wird der Prozeß vom Anwender gesteuert und kann daher mehrere Durchläufe und Schleifen enthalten. Im folgenden Abschnitt wird der Data-Mining-Schritt genauer betrachtet.

2.1.3. Data-Mining

Wie bereits oben dargestellt, geht es beim Data-Mining um das Entdecken von Regeln innerhalb einer großen Menge von Daten und das Extrahieren unbekannter Informationen anhand dieser Regeln. Diese Regeln innerhalb der Daten werden auch Muster (engl. Patterns) genannt. Dies geschieht mit Hilfe von Data-Mining-Methoden und dazu passenden Algorithmen.
Die Data-Mining-Phase wird auch als Modellbildungsphase bezeichnet. In dieser Phase wird mit Hilfe von schon erforschten und geprüften Mustern ein Modell aufgebaut. Mit diesem Modell können dann neue Muster gefunden werden.

Muster (Patterns)

Ein Muster ist eine Beschreibung einer Anzahl von Datensätzen, die in bestimmten Punkten Ähnlichkeiten aufweisen. Ein Muster trifft eine zusammenfassende Aussage über diese Daten und geht dabei über eine reine Aufzählung der Datensätze hinaus. Damit werden bestimmte Regelmäßigkeiten und Beziehungen zwischen den Datensätzen erfaßt (Hagedorn et al. ,1997).
Zum Beispiel ist innerhalb einer Datenbanktabelle ein Muster definiert als eine Menge von Zeilen, bei denen sich in zwei oder mehr Spalten jeweils die gleichen Werte befinden. Das untenstehende Beispiel stellt eine Datenbanktabelle dar, in der sich Daten über Objekte befinden, und zwar die Form, die Farbe und das Gewicht des jeweiligen Objektes. In dem Beispiel bilden die Zeilen eins, zwei und drei ein Muster, da bei allen drei Zeilen in der Spalte “Form” jeweils der Wert “Viereck” steht und in der Spalte “Farbe” jeweils der Wert “rot”. Betrachtet man nur diese drei Zeilen, dann haben alle Objekte, die viereckig sind, die Farbe rot. Dieses Muster beinhaltet also das Wissen:
- Alle Vierecke sind rot.

Diese Aussage läßt sich auch als Wenn-dann-Regel ausdrücken:

- Wenn Form = Viereck, dann Farbe = rot

Zeile #	Form	Farbe	Gewicht
1 ->	Viereck	rot	100
2 ->	Viereck	rot	200
3 ->	Viereck	rot	300
4	Viereck	blau	400
5	Kegel	blau	400

Um die Aussagekraft eines Musters bzw. einer Regel einzuschätzen zu können, werden zwei Parameter verwendet: S*upport* und C*onfidence*. Der Support eines Musters ist die Anzahl der Zeilen, die zu dem Muster gehören. Er ist ein Maß für die statistische Signifikanz des Musters. In unserem Beispiel ist der Support des obigen Musters 3, da es aus drei Zeilen besteht. Oft wird der Support auch als die relative Häufigkeit in Bezug auf alle Datensätze definiert. Muster mit hohem Support beziehen sich auf mehr Datensätze als Muster mit niedrigem Support und sind daher aussagekräftiger. Im Beispiel befindet sich in der Zeile 4 ein Viereck mit der Farbe blau. Nimmt man diese Zeile zu den drei vorherigen hinzu, so stimmt die obige Aussage „Alle Vierecke sind rot." nicht mehr ganz. Genauer gesagt stimmt sie nur noch zu 75%. Dies ist die Confidence des Musters. Die Confidence eines Musters mit den Attributen A und B ist die Wahrscheinlichkeit, daß ein Datensatz mit dem Attribut A auch das Attribut B enthält. In diesem Fall ist das Attribut A die Form „Viereck" und das Attribut B die Farbe „rot". Die Aussagekraft einer Regel läßt sich nun durch den Support und die Confidence dieser Regel beschreiben. Die Aussagekraft wird auch als Stärke (engl. Power) eines Musters bzw. einer Regel bezeichnet. Regeln mit hohem Support und hoher Confidence haben auch eine hohe Aussagekraft.

Modellbildung

Während ein Muster eine konkrete Aussage über eine Menge von Daten darstellt, ist ein Modell eine übergeordnete Methode, mit der solche Aussagen gewonnen werden können. Zur Verdeutlichung betrachten wir die obige Aussage „Alle Vierecke sind rot" als die Anwendung einer Funktion auf eine Form „Viereck", die „rot" als Ergebnis ausgibt, also f(Viereck) = rot. Das zugehörige Modell ist nun die allgemeine Funktionsvorschrift, die einer Form eine Farbe zuweist, also f: Form → Farbe. Ein Modell kann also als eine mathematische Funktion gesehen werden, die alle zugehörigen Muster beschreibt.
Beim Data-Mining gilt es nun, die Modelle zu finden, die am besten geeignet sind, um die aussagekräftigsten Muster zu erhalten. Ziel der Data-Mining-Forschung ist es, hierfür möglichst allgemein verwendbare und effiziente Methoden zu entwickeln (Hagedorn et al. 1997).

Modellrepräsentation

Es wird unterschieden zwischen der Funktion des Modells und der Repräsentation des Modells. Während die Funktion eines Data-Mining-Modells sich an dem Ziel des Data-Mining-Vorganges orientiert (Klassifikation, Segmentierung, …), dient die Modellrepräsentation zur Beschreibung der entdeckbaren Muster. Dies sind Data-Mining-Techniken wie Entscheidungsbäume, Klassifikationsregeln oder Neuronale Netze.
Verschiedene Techniken zur Repräsentation von Klassifikationsmodellen werden in Kapitel 3.2. vorgestellt.

Data-Mining-Methoden

Die Einsatzmöglichkeiten und der Erfolg verschiedener Modelle und Data-Mining Methoden richtet sich stark nach der konkreten Aufgabe und dem Ziel, welches durch das Data-Mining erreicht werden soll.
Nach Fayyad et al. (1996) läßt sich der Data-Mining-Vorgang zwei übergeordneten Zielen nach einteilen. Dies sind Vorhersage (prediction) und Beschreibung (description). Bei der Vorhersage werden die durch das Data-Mining gewonnenen Daten dazu benutzt, unbekannte oder zukünftige Werte vorherzusagen. Z.B. läßt sich die Überlebenswahrscheinlichkeit eines an Krebs erkrankten Patienten anhand der gesammelten Daten seiner Diagnosebefunde erkennen. Bei der Beschreibung dient das Data-Mining dem Auffinden von Strukturen innerhalb der Daten, die dann so aufbereitet werden, daß sie vom Menschen interpretierbar sind.
Je nach dem, ob eine Vorhersage oder eine Beschreibung das Ziel ist, werden unterschiedliche Data-Mining-Methoden eingesetzt. Die Methoden lassen sich nicht immer eindeutig den oben genannten Zielen zuordnen. Auch in der Literatur variiert die Einteilung der Methoden von Autor zu Autor. Die nachfolgende Einteilung orientiert sich an der Klassifizierung von Han und Kamber (2001). Zu den beschreibenden Methoden gehören demnach:

- Charakterisierung und Vergleich
- Assoziationsregeln
- Clustering
- Erkennung von Ausreißern

Zu den vorhersagenden Methoden gehören:

- Klassifikation
- Vorhersage von Werten / Regression

Diese Arbeit verfolgt das Ziel der Klassifikation von Webdokumenten. Bei der Klassifikation geht es darum, einzelne Datensätze in vorher definierte Klassen einzuordnen. Im Kapitel 3.1. Techniken wird diese Methode näher erläutert.
Sind die Klassen nicht vordefiniert, sondern werden sie während des Data-Minings erzeugt, so wird dies Clustering genannt. Auch diese Methode wird in Kapitel 3.1. vorgestellt.

Nachdem die Methode für das Data-Mining festgelegt ist, folgt im nächsten Schritt die Konstruktion und Anwendung eines Data-Mining-Algorithmus.

Der Data-Mining Algorithmus

Der Data-Mining-Algorithmus implementiert die im vorangegangen Schritt festgelegte Methode. Hierfür wird durch die folgenden Schritte ein Data-Mining-Modell gebildet.
Zunächst wird die sogenannte abhängige Variable (Zielgröße) definiert. Dies ist die Ausgabevariable, die durch das Data-Mining-Modell erklärt werden soll. Im vorhergehenden Beispiel ist dies die Farbe der in den Datensätzen gespeicherten Objekte.
Im nächsten Schritt wird das Data-Mining Modell erstellt. Bei den vorhersagenden Methoden geschieht dies anhand einer Menge von Trainingsbeispielen. Es werden verschiedene Data-Mining-Techniken, z.B. Techniken des maschinellen Lernens, eingesetzt, um über die Trainingsbeispiele automatisch ein Modell zu erstellen. Dies wird als Modellsuche bezeichnet.
Im dritten Schritt wird das Modell evaluiert. Hierfür wird das Modell auf eine Menge von Testdatensätzen angewendet und die Ergebnisse auf ihre Güte hin überprüft.
Im letzten Schritt wird das Modell so lange angepaßt, bis es optimale Ergebnisse erzielt. Die Güte des Data-Mining-Algorithmus hängt jedoch auch von der Wahl der Eingabeparameter für das Modell ab. Daher werden auch die Parameter wiederholt auf die erzeugten Ergebnisse hin überprüft und angepaßt. Dies bezeichnet man auch als Parametersuche.

2.1.4. Web-Mining

Bei der Suche nach Informationen im World Wide Web ergeben sich nach Kosala und Blockeel (2000) die folgenden Problemfelder:

- das Auffinden relevanter Informationen
- das Extrahieren von Wissen aus den im Web vorhandenen Informationen
- die Präsentation der erhaltenen Informationen
- das Personalisieren von Informationen aus dem Web

Web-Mining wird meist grob in drei verschiedene Kategorien eingeteilt: Web-Content-Mining, Web-Structure-Minig und Web-Usage-Mining. Die oben genannten Problemfelder lassen sich gut danach kategorisieren.

- Web-Content-Mining. Es wird der Inhalt von Webdokumenten analysiert. Zum Web-Content-Mining zählt das Auffinden relevanter Informationen, das Extrahieren von Wissen und das Präsentieren der im Web vorhandenen Informationen. Eine konkrete Anwendung von Web-Content-Mining sind Suchmaschinen wie z.B. Google.
- Web-Structure-Mining. Web Structure Mining benutzt die Linkstruktur des WWW, um Informationen über Webdokumente zu sammeln. Auch dies bezieht sich vornehmlich auf das Extrahieren von Wissen aus den im Web vorhandenen Informationen und deren Präsentation. Web-Structure-Mining wird beispielsweise in Suchmaschinen für das Page-Ranking eingesetzt.
- Web-Usage-Mining. Web-Usage-Mining sammelt Informationen aus den Logfiles eines Web-Servers. Hierunter fällt vor allem das Lernen von Verhaltensweisen und Wünschen von Benutzern sowie das Personalisieren von Informationen aus dem Web.

Diese Einteilung muß nicht durchgängig gelten, so könnten beispielsweise auch Web-Content-Mining und Web-Structure-Mining für die Personalisierung von Webinformationen eingesetzt werden. Ein großer Unterschied besteht jedoch zwischen Web-Content- und Web-Sturcture-Mining auf der einen Seite und Web-Usage-Mining auf der anderen. Das Web-Usage-Mining geschieht immer lokal auf einem Web-Server,

während die beiden anderen Mining-Verfahren sich auf das gesamte Netz, oder auf einen Teilbereich des WWW beziehen.
Für diese Arbeit sind nur Verfahren, die sich auf Web-Content-Mining und Web-Structure-Mining beziehen, relevant.

Wissensgewinnung im World Wide Web

Web-Mining ist die Anwendung von Data-Mining-Techniken zum automatischen Auffinden und Extrahieren von Informationen in Webdokumenten (Etzioni, 1996). Analog zum Data-Mining kann man auch Web-Mining als Teilschritt eines Prozesses von Wissensgewinnung sehen. In diesem Prozeß geht es, ebenso wie beim Knowledge Discovery in Databases, darum, Wissen aus einer großen Menge von rohen Daten zu gewinnen. Die Datenmenge ist in diesem Fall das World Wide Web. Webdokumente unterscheiden sich jedoch wesentlich von Datenbeständen in Datenbanken. Dadurch ergeben sich gewisse Schwierigkeiten die Data-Mining-Techniken auf das WWW zu übertragen.
Die wichtigsten Unterschiede zum Data-Mining in Datenbanken sind:

1. Der Inhalt von Webdokumenten ist meist unstrukturiert oder semi-strukturiert.
2. Webdokumente sind heterogen. Das bedeutet, sie sind nicht einheitlich aufgebaut und unterscheiden sich daher untereinander stark.
3. Texte im WWW sind in vielen unterschiedlichen Sprachen verfaßt.
4. Webdokumente sind untereinander stark vernetzt. Während es bei Datenbanken meist nur wenige Beziehungen, z.B. über Fremdschlüssel, gibt, sind fast alle Webdokumente über Links mit anderen Webdokumenten verbunden.
5. Das WWW wächst ständig und ändert sich sehr schnell.
6. Die Menge an Daten ist um ein vielfaches größer als in herkömmlichen Datenbanken.

Dennoch weist der Prozeß der Wissensgewinnung aus dem Web Parallelen zum KDD-Prozeß auf. Daher soll für die Wissensgewinnung aus dem Web *(Knowledge Discovery on the World Wide Web)* in dieser Arbeit ein Prozeß analog zum Knowledge Discovery in Databases erarbeitet werden.
Der Web-Mining-Prozeß besteht in Anlehnung an Etzioni (1996), Pal et al. (2002) und Kosala und Blockeel (2000) aus folgenden Teilaufgaben:

- Information Retrieval / Resource Finding
- Information Extraction / Information Selection und Pre-Processing
- Generalization
- Analysis

Im Folgenden sollen nun diese Schritte mit dem KDD-Prozeß verglichen werden und Parallelen aufgezeigt werden.

Web Information Retrieval / Resource Finding
Information Retrieval beschäftigt sich mit der Informationssuche in wenig strukturierten Datenbeständen. Durch Information Retrieval (IR) wird versucht möglichst alle relevanten Informationen zu finden und dabei so wenig wie möglich unrelevante Daten zu bekommen (Pal et al. ,2002).
In den meisten Fällen wird IR für das Finden von Informationen in Texten eingesetzt. Andere IR-Systeme zur Suche von Bildern oder Musik sind noch wenig erforscht und kommen seltener zum Einsatz. Anwendungen für Text-IR-Systeme sind beispielsweise Digitale Bibliotheken und Suchmaschinen im WWW.

Information Retrieval im WWW (Web Information Retrieval) entspricht dem zweite Schritt im KDD-Prozeß. Der IR-Schritt umfaßt sowohl das Suchen, wie auch das Indizieren und Repräsentieren von Dokumenten. Parallel zum KDD-Prozeß sollte diesem Schritt allerdings eine Untersuchung des Anwendungsbereiches und das Festlegen des Zieles des KDW-Prozesses vorausgehen.

Information Extraction / Information Selection und Pre-processing
Information Extraction (IE) im WWW hat das Ziel eine Menge von Dokumenten so zu transformieren, daß die enthaltenen Informationen besser und schneller extrahiert und analysiert werden können. Dies kann beispielsweise das Entfernen von nicht relevanten Worten aus einem Text sein, oder aber das Finden und Extrahieren von Sätzen in einem Text.
Dem Extrahieren von Informationen aus Dokumenten geht das Abrufen der jeweiligen Dokumente voraus. Daher kann man IR und IE als zwei aufeinander folgende Schritte betrachten. Durch das IR werden die relevanten Dokumente gefunden, während beim IE die relevanten Informationen aus den gefundenen Dokumenten herausgesucht werden. In einigen Fällen vermischen sich jedoch IR und IE, vor allem wenn das Extrahieren von Daten schon in der Intention des IR liegt oder die Daten so strukturiert sind, daß das Abrufen des Dokumentes schon einen Extraktionsprozeß beinhaltet.
Aufgrund des ständig wachsenden und sich ständig verändernden WWWs, ist es schlicht nicht möglich IE-Systeme für das WWW manuell zu erstellen. Daher ist eine Möglichkeit sich bei dem Extrahieren von Informationen auf eine spezielle Art von Webseiten zu beschränken. Eine Andere Möglichkeit ist Machine-Learning-Techniken anzuwenden um Regeln oder Patterns für das IE automatisch oder semi-automatisch zu erstellen.
Dieser Schritt entspricht den Schritten drei und vier im KDD-Prozeß.

Generalization
Der Schritt Generalization oder Generalisierung ist mit der Data-Mining-Phase des KDD-Prozesses vergleichbar. Hier werden Techniken des maschinellen Lernens oder andere Techniken eingesetzt, um allgemeine Regeln (Patterns) in den im vorhergehenden Schritt extrahierten Daten zu finden.

Analyse
Hier findet analog zu Schritt sechs im KDD-Prozeß die Validierung und Interpretation der gefundenen Regeln statt.
Diesem Schritt schließt sich, ebenso wie beim KDD, der siebte Schritt an, in dem das gewonnene Wissen weiterverarbeitet wird.

2.1.5. Knowledge Discovery on the World Wide Web

Entsprechend der vorangegangenen Analyse der einzelnen Schritte im KDD, ergibt sich anlog des KDD-Prozesses, ein Prozeß zur Wissensgewinnung im WWW, der an dieser Stelle *Knowledge Discovery on the World Wide Web* (KDW) genannt werden soll.
Der KDW-Prozeß gliedert sich in die folgenden Schritte:

1. **Untersuchung des Anwendungsbereiches** und Festlegen des Ziels des KDW-Prozesses.
2. **Web Information Retrieval / Rescource Finding.** Suchen, Indizieren und Repräsentieren von Dokumenten.

3. **Information Extraction / Information Selection und Pre-processing.** Transformation der Dokumente, zur besseren und schnelleren Extraktion von Wissen. Extraktion von Feature-Werten (vgl. Kap. 3.1.2).
4. **Generalization / Web-Mining.** Hier geschieht die Extraktion der Muster aus den Daten. Diese Phase besteht aus folgenden drei Teilschritten:
    a. Auswahl der Web-Mining Modells, z.B. Klassifizierung, Regression, Clustering.
    b. Auswahl des Web-Mining Algorithmus und Bildung des Modells, mit dem die Muster gefunden werden sollen.
    c. Eigentliches Web-Mining, d.h. Suche nach Mustern mit Hilfe des ausgewählten Algorithmus.
5. **Analyse / Interpretation und Evaluation** der gewonnen Daten. Die extrahierten Informationen werden dem Benutzer in einer leicht verständlichen und faßbaren Form dargestellt. Anschließend wird eventuell ein erneutem Durchlauf der vorherigen Schritte mit veränderten Parametern durchgeführt.
6. **Verarbeitung des gewonnenen Wissens.** Anwendung oder Weiterverarbeitung des gewonnenen Wissens.

Der Prozeß kann ebenso wie der KDD-Prozeß durch den Anwender gesteuert werden und mehrere Durchläufe und Schleifen enthalten.

2.2. Dokumentensuche im World Wide Web

In diesem Kapitel werden Methoden für die Suche im WWW erläutert. Es wird ein Überblick über die verschiedenen Arten von Suchdiensten im WWW gegeben. Anschließend erfolgt eine Analyse der Probleme, die bei der Suche entstehen können. Schließlich werden Kriterien dargestellt, die für wissenschaftliche Informationssuchende relevant sind

2.2.1. Suchdienste im WWW

Es gibt sehr viele verschiedene Suchdienste im Internet für die unterschiedlichsten Zwecke und mit den unterschiedlichsten Funktionen. Im Folgenden soll daher eine Klassifizierung der verschiedenen Suchdienste vorgenommen werden.
Die Einteilung orientiert sich im Groben an der Typologisierung von T. Koch. (1997). Hiernach lassen sich Suchdienste zunächst in zwei Gruppen einteilen:

1. **Allgemeine Suchdienste für alle Typen von Webressourcen.** Hierbei handelt es sich meist um die Suche nach HTML-, PDF-, und Word-Dokumenten im WWW.
2. **Suchdienste, die auf bestimmte Ressourcentypen oder bestimmte Internetprotokolle spezialisiert sind.** Zu diesen zählen z.B. WAIS, OPAC, Gopher und Z39.50.

In dieser Arbeit soll es jedoch um die Suche nach wissenschaftlichen Dokumenten im World Wide Web gehen. Daher sind an dieser Stelle nur Ressourcen von Bedeutung, die im WWW, also über das HTTP-Protokoll, erreichbar sind. Zudem sollen unter der Bezeichnung „Dokumente“ ausschließlich Textdokumente in den im Web gebräuchlichen Formaten HTML und PDF gefaßt werden. Daher werden hier nur die Dienste der ersten Gruppe behandelt.

Diese Gruppe läßt sich weiter unterteilen in **serverbasierte und clientbasierte Suchdienste**. Koch bezeichnet diese auch als kollektive öffentliche Suchdienste bzw. individuelle private Suchdienste.
Zu den öffentlichen oder serverbasierten Suchdiensten gehören alle herkömmlichen Suchmaschinen, die öffentlich über das WWW zugänglich sind, beispielsweise Google oder Yahoo.
Die privaten oder clientbasierten Suchdienste sind nicht so verbreitet wie die serverbasierten. Das Prinzip besteht darin die Umsetzung der Suchanfrage, den Suchalgorithmus und die Aufbereitung der Ergebnisse auf den Client zu verlagern. Hier hat der Benutzer die Möglichkeit, das Client-System und so die Suche selbst seinen persönlichen Bedürfnissen anzupassen.
Ein weiterer Vorteil besteht darin, daß Teile des Suchprozesses weiter automatisiert werden können als bei serverbasierten Diensten. So könnte ein Client-System so aufgebaut sein, daß der Suchdienst die Rolle eines Benutzer übernimmt und durch verschiedene Hypertexte navigiert, um zu relevanten Dokumenten zu gelangen. Die Gefundenen Dokumente könnte es dem Nutzer dann nach dessen Wünschen aufbereitet präsentieren. Der Nutzer könnte die Suchtiefe und Art der Relevanzanalyse bestimmen oder die Suche mitverfolgen und beliebig anhalten und fortsetzen.
Zu den privaten oder clientbasierten Suchdiensten gehören beispielsweise WebFerret (www.ferretsoft.com) und WebSeeker (www.bluesquirrel.com/ products/webseeker/). Auch das im Rahmen dieser Arbeit erstellte System gehört zu den clientbasierten Suchdiensten.

In beiden Kategorien lassen sich nun jeweils drei Untergruppen bilden.
A) Einfache Suchdienste,
B) Metasuchdienste und
C) Suche mit intelligenten Agenten.

Zu den öffentlichen einfachen Suchdiensten zählen
a) Crawler-basierte Dienste
b) Verzeichnisbasierte Dienste
c) Hybride Suchdienste

Suchdienste der Kategorie a) werden auch einfach als **Suchmaschinen** bezeichnet. Sie durchsuchen das Web mittels Suchalgorithmen, die Robots, Spiders, Wanderer oder Crawler genannt werden. Die Webressourcen werden maschinell erschlossen, indem die Crawler das Web über Hyperlinks durchwandern und dabei einen Index der gefundenen Seiten anlegen. Bei einer Benutzeranfrage wird dann nicht das Web direkt, sondern der von den Robots aufgebaute Index durchsucht.

Verzeichnisbasierte Suchdienste (Web-Directories) bieten einen Index aus meist manuell erschlossenen, nach Kategorien geordneten Webressourcen an. Durch die manuelle Bearbeitung ist die Qualität der Suchergebnisse in einem solchen Index meist höher als bei Crawler-basierten Diensten.

Nicht alle Dienste lassen sich strikt nach dieser Einteilung klassifizieren. So gibt es auch **hybride Suchdienste**. Dies sind Suchmaschinen, die zusätzlich noch einen Verzeichnisbasierten Dienst anbieten.
In der heutigen Zeit sind Suchdienste der Kategorie a) und b) beinahe ausgestorben. Die meisten Suchmaschinen gehören heute zu den hybriden Suchdiensten. So werden bei

Yahoo, welches ursprünglich nur aus einem verzeichnisbasierten Dienst bestand, nun auch Suchergebnisse angezeigt, die von Crawlern geliefert werden. Ebenso hat Google mittlerweile seinen Dienst um ein Web-Directory erweitert.

Metasuchdienste sind Suchdienste, die die Suchanfrage des Benutzers parallel an andere Suchdienste weiterleiten und aus den Ergebnissen eine eigene Ergebnisliste zusammenstellen. Auf diese Weise kann mit einer Anfrage gleich über mehrere Suchdienste gesucht werden.

Auch **intelligente Agenten** können für die Suche im Web eingesetzt werden. Intelligente Agenten können autonom und intelligent im Auftrag eines Benutzers handeln. Agenten können also Aufgaben, die ein menschlicher Benutzer bei der Suche im Web ausführt, übernehmen und an seiner Stelle erledigen.
Auf diese Weise können weite Teile des Suchprozesses automatisiert werden. Dies kann eine große Arbeitserleichterung bedeuten.

1. Allgemeine Suchdienste für alle Typen von Webressourcen
   - 1.1. Serverbasierte Suchdienste
     - A) Einfache Suchdienste
       - a) Suchmaschinen: Crawler-basierte Dienste
       - b) Verzeichnisbasierte Dienste
       - c) Hybride Suchdienste
     - B) Metasuchmaschinen
     - C) Intelligente Agenten
   - 1.2. Clientbasierte Suchdienste
     - A) Einfache Suchdienste
       - a) Suchmaschinen: Crawler-basierte Dienste
       - b) Verzeichnisbasierte Dienste
       - c) Hybride Suchdienste
     - B) Metasuchmaschinen
     - C) Intelligente Agenten
2. Spezialisierte Suchdienste für spezielle Ressourcentypen und Protokolle

Typologisierung von Suchdiensten nach T. Koch. (1997)

2.2.2. Probleme bei der Suche im WWW

Wie bereits in Kapitel 2.2.3. dargestellt, verfügt das WWW über einige Eigenschaften, die die Suche nach Informationen im Web im Vergleich zur Suche in klassischen Datenbanken erschweren.

Hierbei ist vor allem die Größe des Webs zu nennen. Das WWW besteht aus einer riesigen Menge von Webseiten (derzeitige Schätzungen belaufen sich auf 10-15 Milliarden frei zugängliche Seiten) und wächst ständig.

Die Webseiten sind zudem weltweit auf verschiedenen Hostrechnern verteilt. Ein Zugriff auf die Seiten ist im allgemeinen nur über Links möglich. Dabei bilden die Verlinkungen der Seiten die wesentliche Struktur des Webs.
Ein weiteres Hindernis für die maschinelle Erfassung der Inhalte von Webseiten ist der Umstand, daß diese meist unstrukturiert (reine Textdokumente) oder semistrukturiert (HTML-Dokumente) sind und zudem untereinander stark heterogen.
Desweiteren sind Webseiten ständigen Veränderungen unterworfen. Sowohl die Inhalte als auch die URLs der Webseiten ändern sich häufig und willkürlich.

Suchmaschinen sind die derzeit meistgenutzten Suchdienste für die Suche im WWW. Obwohl Web-Directories beim Auffinden von bestimmten Informationen durch die Kategorisierung der Webseiten einen Vorteil bieten, haben sich Suchmaschinen aufgrund der breiteren Abdeckung und der Aktualität ihrer Indexe durchgesetzt. Daher wird in diesem Abschnitt speziell auf die Suche mit Suchmaschinen eingegangen.
Andere Suchdienste, wie die Suche mit Hilfe von intelligenten Agenten, werden noch relativ wenig eingesetzt, obwohl auch sie Vorteile gegenüber Suchmaschinen bieten können. Welche Vorteile dies sind wird in Kapitel 2.4. erörtert.

Obwohl die Indexe der großen Suchmaschinen, wie Google, MSN oder Yahoo, sich in der Vergangenheit stark vergrößert haben, ist es allein aufgrund der Größe des Webs nicht möglich, alle Seiten des Webs zu indizieren. Daher werden Webseiten nur bis zu einer gewissen Link-Tiefe von Suchmaschinen indiziert.

Eine besondere Herausforderung für Suchdienste, stellt die hohe Veränderungsfrequenz der Webseiten dar. Für Suchmaschinen und Web-Directories bedeutet dies den Index ständig aktuell halten zu müssen. Gerade durch die enorme Anzahl sich ständig verändernder Webseiten, wird dies zum Problem.
Untersuchungen zeigen, daß die bestehenden Suchmaschinen nicht in der Lage sind, der hohen Veränderungsfrequenz der Webseiten gerecht zu werden. Es ergibt sich im Durchschnitt eine Verzögerung von dreißig Tagen, bis eine Seite im Index einer Suchmaschine aktualisiert wird (Lewandowski, 2005).

Ein weiteres Problem stellt die Vernetzung der Webseiten über Links dar. Da es eine große Menge von Webseiten gibt, die nicht über Links erreichbar sind, können diese auch nicht von Suchmaschinen indiziert werden. Abhilfe kann nur in begrenztem Umfang durch das manuelle Anmelden von Webseiten bei Suchdiensten geschaffen werden.

Diese Probleme bei der Indizierung des Webs machen deutlich, daß bei der Suche mit Suchmaschinen einerseits nur ein Teil des WWW abgedeckt werden kann und andererseits die Ergebnisse der Suche oft nicht mehr aktuell sind.
Eine Maßnahme, um aktuellere Suchergebnisse zu erzielen und auch Seiten zu erreichen, die nicht von Suchmaschinen erfaßt werden, ist der Einsatz von personalisierten Suchagenten, die das Web in Echtzeit durchsuchen. Zwei solche Systeme werden in Kapitel 4.2. vorgestellt.

Eine weitere Schwäche herkömmlicher Suchmaschinen ist die Keyword-bezogene Suche. Sie erfordert vom Suchenden, die Informationen nach denen gesucht wird, mit einigen wenigen Schlüsselworten zu beschreiben. Die Auswahl der richtigen Keywords ist dabei ausschlaggebend für die Relevanz der Suchergebnisse. Oft ist es für den Suchenden nicht möglich, auf Anhieb die richtigen Schlüsselworte zu wählen, so daß mehrere Anläufe mit

verschiedenen Suchworten unternommen werden müssen, um zu den gewünschten Ergebnissen zu gelangen.
Hier könnten Benutzerschnittstellen helfen, die es erlauben, natürlichsprachliche Anfragen an die Suchmaschine zu stellen. Solche Systeme sind derzeit noch im Entwicklungsstadium. Einige lauffähige Versionen sind jedoch bereits im Bereich der Online-Bibliothekskataloge zu finden. Beispiele hierfür sind NLI-Z39.50 (Natural Language Interface für den Zugriff auf Bibliotheksdatenbanken, die das Z39.50 Protokoll unterstützen) und NEBIS (Netzwerk von Bibliotheken und Informationsstellen in der Schweiz).
Eine Alternative wäre ein System, das es dem Suchenden erlaubt, ganze Texte als Suchanfrage in die Suchmaschine einzugeben, um dann im Web nach ähnlichen Texten zu suchen. Solche Suchanfragen nach ähnlichen Dokumenten sind bereits in mehreren Suchmaschinen integriert, jedoch ist es nur möglich, nach Webseiten zu suchen, die einer Seite des vorangegangenen Suchergebnisses ähneln. Es ist nicht möglich, eigene, lokale Texte für die Suche zu verwenden.
Das im Rahmen dieser Arbeit entwickelte System DocAssistant bietet daher die Möglichkeit, ausgehend von lokal gespeicherten Texten, nach ähnlichen Dokumenten zu suchen. Das genaue Vorgehen bei einer solchen Suche wird in Kapitel 5 erläutert.

Das Suchergebnis besteht bei Suchmaschinen meist aus einer nach Relevanz geordneten Liste von Webseiten. Bei einem umfangreichen Ergebnis kann diese Darstellung problematisch sein, da meist nur die obersten Webseiten der Liste beim Suchenden Beachtung finden. Zudem ist es auch gar nicht möglich, bei einem Ergebnis von beispielsweise 8.560.000 Webseiten, die den Begriff “Java Server Faces” enthalten, alle Seiten manuell zu durchblättern. Hier muß man sich auf die Relevanzanalyse der Suchmaschine verlassen, und hoffen, daß die gewünschten Seiten auf den obersten Plätzen des Ergebnisses zu finden sind. Ist dem nicht so, bleibt nur eine erneute Suchanfrage mit veränderten Suchworten.
Hilfreich sind hier Suchdienste, die ihre Ergebnisse nach Themen geordnet ausgeben. So ist es dem Suchenden möglich, mit einem Blick zu erkennen, ob und welche Seiten zu dem gewünschten Thema gefunden wurden. Beispiele für solche Suchdienste werden in Kapitel 4.1.1. vorgestellt.

2.2.3. Kriterien wissenschaftlicher Informationssuchender

Als wissenschaftliche Informationssuchende können alle Informationssuchenden angesehen werden, die im World Wide Web nach wissenschaftlichen Publikationen und Informationen suchen. Dies können Studierende, Doktoranden, Professoren und wissenschaftliche Mitarbeiter sein. Zudem können auch wissenschaftlich interessierte Laien Bedarf an wissenschaftlichen Informationen aus dem WWW haben.

Wissenschaftliche Publikationen im WWW sind in den meisten Fällen Publikationen, wie sie in wissenschaftlichen Fachzeitschriften veröffentlicht werden. Diese Artikel werden immer häufiger auch als PDF-Dokumente frei zugänglich im World Wide Web veröffentlicht.
Online-Publikationen von wissenschaftlichen Artikeln lassen sich häufig direkt auf den Webseiten der jeweiligen Autoren finden. Auch auf den Webseiten von Hochschulen und Instituten werden wissenschaftliche Publikationen frei zugänglich veröffentlicht. Weitere Quellen für Online-Artikel sind die Webseiten von Konferenzen, Tagungen und anderen wissenschaftlichen Veranstaltungen.

Diese Publikationen werden von Suchmaschinen indiziert und sind so im Web auffindbar und durchsuchbar. Wissenschaftliche Publikationen können auch über spezialisierte Suchmaschinen wie Scirus (www.scirus.com) und Google Scholar (scholar.google.com) oder Literaturdatenbanken wie die Elektronische Zeitschriftenbibliothek Regensburg (rzblx1.uni-regensburg.de/ezeit/) oder Citeseer (citeseer.ist.psu.edu) gefunden werden.

Vorgehen bei der Suche

Wissenschaftliche Informationssuchende haben zunächst die gleichen Ansprüche an Suchdienste wie andere Suchende auch. Sie wollen möglichst schnell und ohne viel Aufwand die Informationen finden, nach denen sie suchen.
Darüber hinaus haben Informationssuchende mit wissenschaftlichem Hintergrund jedoch weitergehende Ansprüche:

- Es wird regelmäßig nach neuen Veröffentlichungen zu einem bestimmten Themenbereich gesucht. Dabei werden möglichst umfassend alle zu einem Thema vorhandenen Quellen benötigt.
- Es werden qualitativ anspruchsvolle Ergebnisse erwartet.
- Die gefundenen Publikationen müssen für die spätere Nutzung wieder auffindbar sein. Also müssen sie archiviert werden oder Informationen über den Fundort gespeichert werden.

Um Veröffentlichungen zu einem bestimmten Thema zu finden, können Suchdienste, wie die oben genannten, eingesetzt werden.
Um mit Hilfe einer Suchmaschine möglichst viele Quellen zu einem bestimmten Thema zu finden, ist es nötig, geeignete Suchanfragen zu formulieren. Hierfür müssen die Suchworte gefunden werden, die für das Thema charakteristisch sind und im Index der eingesetzten Suchmaschine mit den entsprechenden Seiten verknüpft sind.
Oft sind dem Suchenden bereits Publikationen zu dem gesuchten Thema bekannt, bzw. entsprechende Dokumente sind bereits lokal gespeichert vorhanden. Daher könnte hier von einer Suche nach ähnlichen Dokumenten, wie im vorherigen Abschnitt erläutert, profitiert werden. Damit wäre die Auswahl von zur Suche geeigneten Keywords überflüssig.

Problematisch bei der Suche über Suchmaschinen ist, wie bereits dargestellt, die Aktualität der Ergebnisse. Daher kann es sinnvoll sein, Webseiten, die einen zu dem gesuchten Thema relevanten Inhalt versprechen, direkt aufzurufen und manuell zu durchsuchen. Auf diese Weise werden auch Publikationen gefunden, die von den Suchmaschinen nicht erfaßt wurden und wissenschaftliche Informationssuchende gelangen so zu einem umfassenderen Suchergebnis.
Viele neue Publikationen zu einem bestimmten Thema können beispielsweise ausfindig gemacht werden, indem, ausgehend von einer bereits gefunden Veröffentlichung, nach weiteren Publikationen des selben Autors gesucht wird. Diese können oft direkt auf den Homepages der Autoren gefunden werden.

Die Qualität von Publikationen wird oft an der Häufigkeit, mit der diese zitiert werden, gemessen. Daher bietet es sich an, dieses Kriterium auch bei der Suche nach Online-Publikationen als Qualitätsmaßstab zu verwenden. Die Zitierhäufigkeit spielt bei den meisten, auf wissenschaftliche Literatur spezialisierten Suchmaschinen eine große Rolle.

So dient die Zitierhäufigkeit bei Google Scholar dem Ranking der Suchergebnisse, bei Citeseer wird die Zitierhäufigkeit als wichtiges Merkmal direkt neben dem Titel der Publikation angezeigt.
Durch den Einsatz von spezialisierten Suchdiensten ist also eine gewisse Qualität der Ergebnisse bereits gewährleistet.

Ein wichtiger Aspekt, der bei der Recherche nach digitalen Dokumenten im WWW nicht außer Acht gelassen werden darf, ist die anschließende Archivierung der gefundenen Dokumente.
Es bietet sich an, gefundene Dokumente nach Themen geordnet, beispielsweise in einer Ordnerstruktur, zu speichern. Zudem sollten Quellenangaben, Autor, Titel und die URL gespeichert werden.

Unterstützung der Suche

Das oben skizzierte Vorgehen bei der Suche nach wissenschaftlichen Publikationen im WWW soll durch das in dieser Arbeit vorgestellte System unterstützt werden. Es ergeben sich zwei Aspekte, die durch ein Softwaresystem unterstützt werden können.
Zum einen ist dies der eigentliche Suchvorgang nach relevanten Publikationen. Dies umfaßt den Einsatz von verschiedenen Suchdiensten sowie auch die manuelle Suche durch das direkte Aufsuchen von Webseiten mit relevanten Dokumenten. Dieser Suchvorgang läßt sich mit Hilfe von intelligenten Agenten automatisieren, wie in den Kapiteln 4.2. und 5. gezeigt wird.
Zum anderen läßt sich auch die Archivierung der gefundenen Dokumente unterstützen. Dies geschieht in dem hier entwickelten System durch das Ordnen der gefundenen Publikationen nach Themen und die Speicherung der Dokumente in einer entsprechenden Ordnerstruktur.

2.3. Intelligente Agenten

Agenten stellen eine Metapher für Systeme dar, die im Auftrag eines Benutzers Aufgaben ausführen können und dabei autonom, also selbstbestimmt, handeln. Der Begriff des „Intelligenten Agenten“ wird schon seit längerem in der Künstlichen Intelligenz (KI) verwendet. Im Bereich der Praktischen Informatik ist der Agentenbegriff jedoch noch relativ neu. Er bedeutet einen Paradigmenwechsel von direkter Manipulation (Direct Manipulation) zu einer entkoppelten Ausführung von Aufgaben durch einen Agenten (Indirect Management) (Maes, 1994).
Das bisherige Paradigma der direkten Manipulation steht für das direkte Ausführen aller Aktionen durch den Benutzer selbst. Beim Indirect Management tritt der Agent als Mittler zwischen dem System und dem Benutzer auf. Dies ermöglicht es einem Benutzer, der nur geringes Wissen über das System hat, den Agenten sozusagen als Experten zu beauftragen, eine bestimmte Aufgabe zu erfüllen. Ebenso kann ein Agent dem Benutzer Aufgaben abnehmen, die monoton und daher langweilig und lästig für menschliche Anwender sind.

Es existiert keine einheitliche Definition dafür, was ein Agent ist. Es werden jedoch üblicherweise typische Eigenschaften von Agenten herangezogen, um Agenten näher zu beschreiben.

Jennings und Wooldridge (1995) fassen in einer "Weak Notion of Agency" die wichtigsten Attribute eines Agenten zusammen. Demnach ist ein Agent ein System, welches die folgenden Eigenschaften besitzt:

Autonomie (autonomy)	Agenten arbeiten selbständig ohne das direkte Einschreiten des Anwenders. Sie haben eine gewisse Kontrolle über ihren internen Zustand und ihre Aktionen.
Reaktivität (reactivity)	Agenten nehmen ihre Umwelt wahr und reagieren auf Veränderungen, die in ihr stattfinden. Die Umwelt kann dabei beispielsweise aus der physikalischen Welt, den Eingaben eines Benutzers, anderen Agenten oder dem Internet bestehen.
Voraushandlung (pro-activeness)	Agenten reagieren nicht nur auf die Umwelt, sondern verhalten sich auch aus eigener Initiative heraus, um bestimmte Ziele zu erreichen.
Soziale Fähigkeiten (social ability)	Agenten interagieren mit dem Anwender und mit anderen Agenten durch eine Agentenkommunikationssprache.

Durch diese Eigenschaften werden Agenten bei Wooldridge eindeutig von „normalen" Programmen abgegrenzt.

Betrachtet man nur die Eigenschaft der Voraushandlung, die gleichzusetzen ist mit zielorientiertem Verhalten, würden schon einfache Prozeduren oder Funktionen, wie man sie in Pascal, C oder Java schreibt, unter den Agentenbegriff fallen. Eine solche Prozedur besteht aus Preconditions und Effekten (Postconditions). Wenn die Prozedur korrekt ausgeführt wurde und die Preconditions zutreffen, treten anschließend die Postcondtions ein. Die Postconditions kann man als die Ziele der Prozedur betrachten. Die Ziele werden durch den Programmierer festgelegt, der die Prozedur programmiert hat. Die Prozedur wird ausgeführt, wenn die Preconditions erfüllt sind. Sie terminiert, wenn das Ziel erreicht ist. Dies könnte man als zielorientiertes Verhalten betrachten.

Ein Agent muß jedoch mehr können. Agenten müssen auch in Umgebungen zurechtkommen, in denen normale Programme nicht funktionieren würden.
Ein wichtiges Kriterium ist, daß Agenten auch noch funktionieren müssen, wenn sich ihre Umwelt verändert. D.h. sie müssen sich an ihre Umwelt anpassen können. Die oben beschriebene Prozedur würde wahrscheinlich einen Fehler produzieren, wenn sich die Umgebung, also die Precondtions der Prozedur während ihres Ablaufs ändern würde, oder aber das Ergebnis wäre nicht vorhersehbar. Ebenso könnten sich auch die Ziele

während des Ablaufs der Prozedur ändern. Dann wäre es gar nicht mehr notwendig, die Prozedur bis zum Ende auszuführen.

Um auf solche Änderungen reagieren zu können, müssen Agenten die Eigenschaft der Reaktivität besitzen. In Multiagenten Systemen ist es z.B. oft der Fall, daß sich Umwelt und Ziele des Agenten ändern, wenn jeder Agent Einfluß auf seine Umgebung nimmt und somit den Zustand der Umwelt verändert.

Auch Systeme, die sich reaktiv verhalten, lassen sich einfach durch herkömmliche Programme implementieren. Agenten müssen jedoch beide Eigenschaften, Proaktivität und Reaktivität, vereinen. Dabei muß eine Balance hergestellt werden, die es dem Agenten erlaubt, auf externe und interne Ereignisse zeitnah zu reagieren. Gleichzeitig darf ihn eine übersteigerte Reaktivität, also das ständige Reagieren auf seine Umwelt, nicht davon abhalten, seine Ziele zu verfolgen. Diese Balance ist nach Wooldrigde (1995) schwer zu erreichen und eines der Kernprobleme beim Entwurf von Agentensystemen.

Die Eigenschaft der Autonomie beschreibt die Fähigkeit eines Agenten selbständig Aufgaben für den Benutzer zu erledigen. Zusammen mit den Eigenschaften der Proaktivität und Reaktivität ergibt dies einen Agenten, der in der Lage ist Aufträge eines Benutzers zu erledigen und dabei auf Änderungen in der Umgebung angemessen zu reagieren, ohne auf ein regulierendes Eingreifen des Benutzers angewiesen zu sein.

Die letzte Eigenschaft betrifft die sozialen Fähigkeiten, die ein Agent haben sollte. Für diese Eigenschaft sind nicht nur rein kommunikative Fähigkeiten notwendig. Reines Interagieren von Agenten mit anderen Agenten oder dem Benutzer könnte man auch mit dem Austausch von Daten zwischen herkömmlichen Programmen gleichsetzen. Statt dessen sind soziale Fähigkeiten, wie beispielsweise Verhandlungsfähigkeit oder die Fähigkeit zur Kooperation gefragt. Ein Agent muß zum Beispiel in der Lage sein, seine Ziele zu überdenken und eventuell Abstriche zu machen, um sie erreichen zu können. Er muß mit anderen Agenten kooperieren und daher auch ein gewisses Verständnis für die Ziele anderer Agenten aufbringen können.
Diese Fähigkeit unterscheidet sich daher vom reinen Austausch von Informationen oder der Fähigkeit Instruktionen entgegen nehmen zu können.

Wann ist ein Agent intelligent?

Wann nun ist ein Agent intelligent? Auch diese Frage ist nicht eindeutig geklärt. Es existieren die unterschiedlichsten Ansätze, um eine Antwort auf diese Frage zu finden. Die denkbar einfachste Erklärung ist die folgende:
Intelligente Systeme sind Systeme, die intelligentes Verhalten an den Tag legen. Da es für den Begriff “Intelligenz” keine allgemein gültige Definition gibt, setzt man menschliches intelligentes Verhalten als Maßstab an. Dazu gehören die kognitiven Fähigkeiten eines Menschen, wie Wahrnehmen, Erkennen, Denken, Schlußfolgern, Lernen usw. Um sich intelligent verhalten zu können, muß ein Agent also über ähnliche Fähigkeiten verfügen.

Die Frage ist, wie diese Intelligenz im Agenten umgesetzt ist. Demnach können Agenten in zwei Gruppen unterteilt werden. Deliberative Agenten und Reaktive Agenten (Nwana, Wooldridge). Deliberative Agenten basieren auf dem Ansatz der *symbolischen künstlichen Intelligenz*. Die symbolische künstliche Intelligenz ist der klassische Zweig der KI. Sie basiert auf der Darstellung von Objekten und Subjekten der realen Welt, sowie ihrer Eigenschaften und Beziehungen, durch Symbole. Wissen über die Umwelt wird im

Computer durch Symbole repräsentiert und unter Nutzung logischer Konzepte wie beispielsweise Prädikatenlogik erster Ordnung verarbeitet.[1] Im Gegensatz dazu steht der Ansatz der *Neuen künstlichen Intelligenz* oder auch *subsymbolischen KI*, der auf Rodney Brooks zurück geht[2]. Hier wird davon ausgegangen, daß intelligentes Verhalten auch ohne explizite symbolische Repräsentation und abstraktes logisches Schlußfolgern auskommt. Intelligenz ist nach diesem Paradigma eher eine emergente Eigenschaft eines komplexen Systems.

Deliberative Architekturen

Nach Wooldridge (1994) verfügen Deliberative Agenten über ein internes umfassendes symbolisches Modell ihrer Umwelt als Wissensbasis und außerdem über die Fähigkeit zur logischen Schlußfolgerung.
Der Anspruch an einen Agenten über die oben genannten Fähigkeiten zu verfügen wird auch als *Stronger Notion of Agency* bezeichnet, im Gegensatz zu der vorher beschriebenen *Weaker Notion of Agency*, also eine starke, bzw. enger gefaßte Definition des Agentenbegriffs. (Wooldridge, 1994)

Die Nachteile des deliberativen Ansatzes liegen in den Problemen der symbolischen KI begründet. Erstens besteht ein Problem, die reale Welt in angemessener Zeit in eine adäquate symbolische Beschreibung zu überführen. Das zweite Problem besteht darin, wie sich überhaupt komplexe Begebenheiten und Prozesse der realen Welt symbolisch repräsentieren lassen. Außerdem besteht die Schwierigkeit, anhand dieser Symbole in angemessener Zeit logisch zu schlußfolgern. Nach Wooldridge sind diese Probleme nur schwer zu lösen.

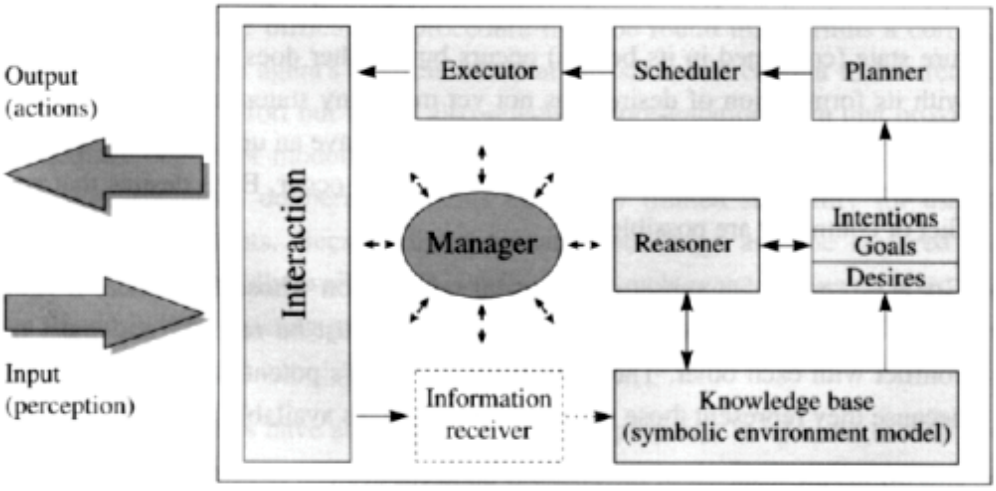

Abbildung 1: Deliberative Agentenarchitektur (Brenner et al., 1998)

[1] Die Grundlagen der symbolischen KI wurden bereits 1976 von Newell und Simon in Form ihrer „physical-symbol system hypothesis“ formuliert. Siehe dazu (Newell und Simon, 1976).
[2] Grundlagen dieses Ansatzes sind zu finden in: (Brooks, 1986) und (Brooks, 1991).

Reaktive Architekturen

Die Probleme der symbolischen KI führten zu dem reaktiven Ansatz der subsymbolischen KI. Reaktive Agentenarchitekturen sind Architekturen, die kein zentrales symbolisches Weltbild besitzen und keine komplexe symbolische Logik zum Schlußfolgern verwenden. Intelligentes Verhalten kann nach Brooks nur durch Interaktion mit der Umwelt entstehen. Seine Theorie setzte er in Form einer reaktiven Agentenarchitektur um. Der reaktive Agent besteht aus einer Hierarchie von Schichten, die alle für verschiedene Aufgaben innerhalb des Agenten verantwortlich sind, beispielsweise eine Schicht für das Vorwärtsbewegen eine andere für das Ausweichen. Dabei erfüllen niedrigere Schichten weniger komplexe Aufgaben als höhere. Jede Schicht konkurriert mit den anderen Schichten um die Kontrolle über den Agenten. Dadurch ist es dem Agenten möglich stets *situationsbezogen* zu operieren. Das Verhalten des Agenten bei Änderungen in der Umwelt kann unmittelbar ein der Situation angemessenes Verhalten hervorrufen. Reaktive Agenten verfügen im Gegensatz zu deliberativen Agenten über eine sehr einfache Architektur, trotzdem können reaktive Agenten intelligentes Verhalten produzieren und auch anspruchsvolle Aufgaben lösen.

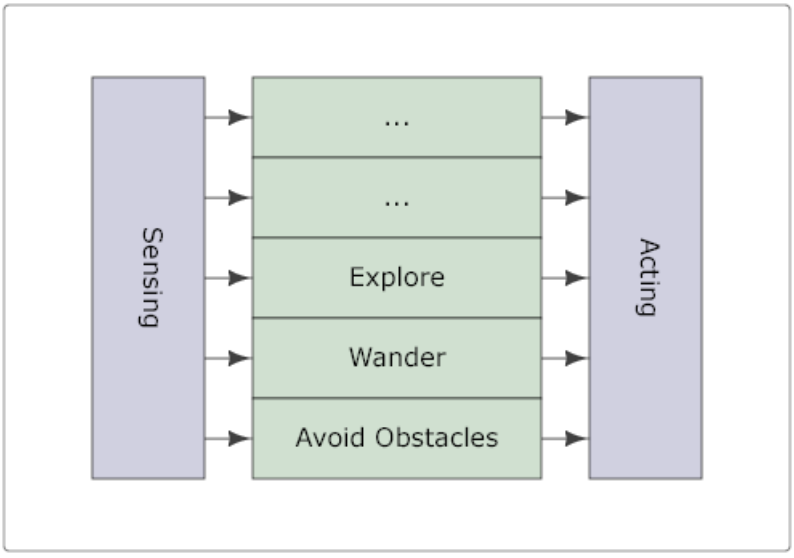

Abbildung 2: Reaktive Agentenarchitektur nach Nwana (1996)

Hybride Architekturen

Hybride Agenten sind Agenten, die ein symbolisches Weltmodell besitzen und nach den Regeln der symbolischen KI schlußfolgern, jedoch ebenfalls über ein reaktives System verfügen, daß in der Lage ist, auf Ereignisse zu reagieren ohne komplexe Schlußfolgerungen anstellen zu müssen. Ein bekanntes Beispiel für eine solche Architektur ist das PRS von Georgeff und Lansky (Georgeff und Lansky, 1987). Außerdem unterstützt das Framework Jadex (Braubach et al., 2005) die Entwicklung hybrider Agenten. Solche Systeme können die Vorteile von deliberativen und reaktiven Systemen vereinen.

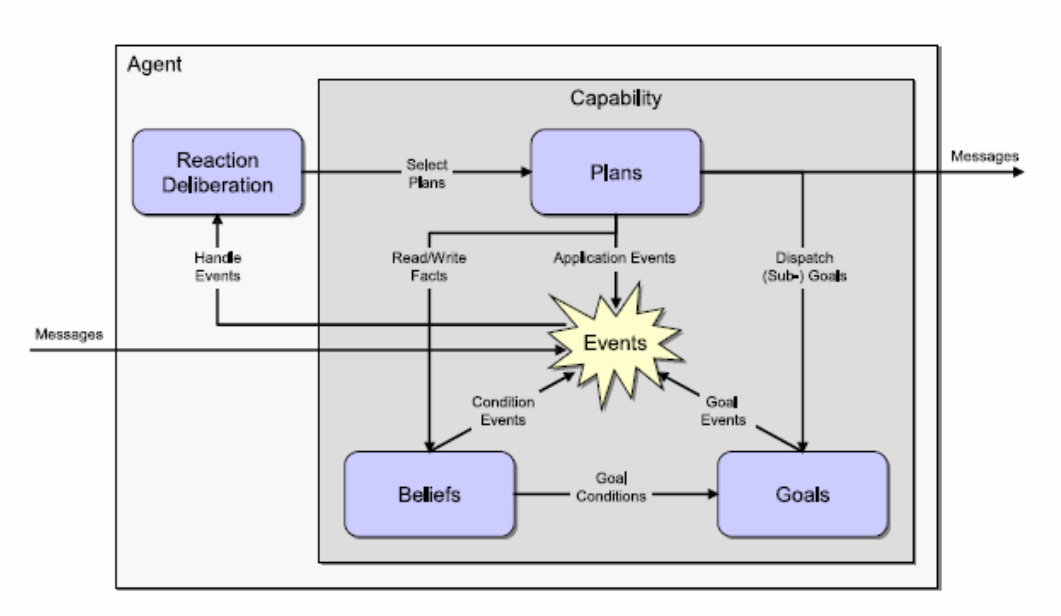

Abbildung 3: Architektur eines Jadex-BDI-Agenten (Pokahr et al., 2005)

2.3.1. Klassifikation nach Aufgabenfeldern

Nwana (1996) versucht mit seiner Agententypologie Agenten nicht nur nach ihrer Architektur zu klassifizieren sondern auch nach den Aufgaben, die sie ausführen. Er nennt zusätzlich zu den bereits im vorigen Kapitel beschriebenen Klassen folgende:

Collaborative Agents: Mit dieser Klasse faßt Nwana Agenten zusammen, die dem vorher beschriebenen Agentenbegriff von Wooldridge entsprechen. Agenten dieser Klasse sollten also über die Fähigkeiten Autonomy, Social Ability, Responsivness (entspricht Reactiveness) und Proactiveness verfügen. Er betont jedoch zusätzlich die Zusammenarbeit von mehreren Agenten an einem gemeinsamen Ziel, die für Agenten dieser Klasse ausschlaggebend ist. Damit sind Collaborative Agents immer im Kontext eines Multiagentensystems zu sehen.

Interface Agents: Dies sind personalisierte Helfer, die mit ihren Benutzern zusammenarbeiten, um die ihnen gestellte Aufgabe zu lösen. Sie arbeiten autonom und lernen durch Interaktion mit Benutzern.

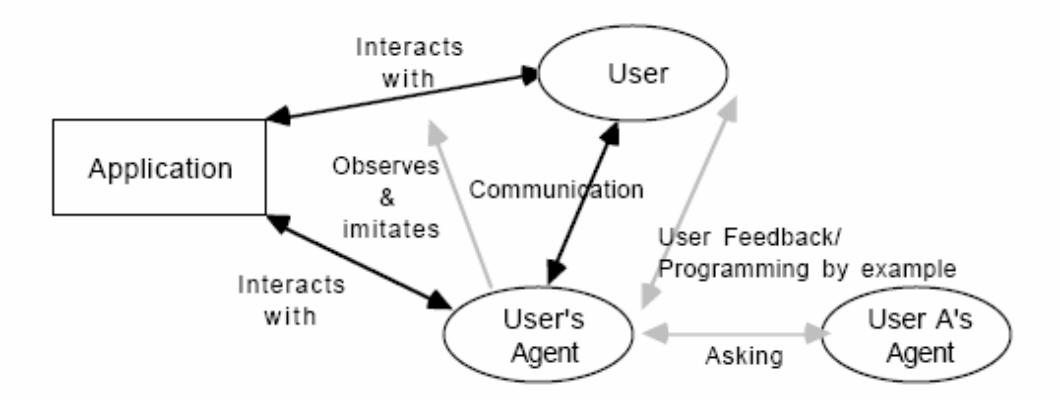

Abbildung 4: Interface Agentenarchitektur nach Maes (1994)

Mobile Agents: Mobile Agenten wandern über ein Netzwerk von einem Host zum anderen und lösen dort ihre Aufgaben durch die direkte Interaktion mit dem fremden Host. Anschließend kehren sie zu ihrem Ausgangspunkt zurück, um Bericht zu erstatten oder mit den gesammelten Ergebnissen weitere Aufgaben zu lösen. Dies reduziert vor allem die Kommunikationskosten und die Belegung von lokalen Ressourcen, birgt aber auch Sicherheitsrisiken.

Information/Internet Agents: Information/Internet Agents werden benutzt, um Informationen aus verschiedenen Quellen, beispielsweise dem Internet, zusammenzutragen, diese zu manipulieren und zu managen. Information/Internet Agents sollen durch das Finden und Bearbeiten von Informationen helfen, die seit dem Entstehen des Internets ständig wachsende Informationsflut zu bewältigen.

Das im Rahmen dieser Arbeit entwickelte Agentensystem ist ein Multiagentensystem, bestehend aus Information/Internet Agents. Daher wird diese Klasse von Agenten im nächsten Abschnitt detaillierter behandelt.

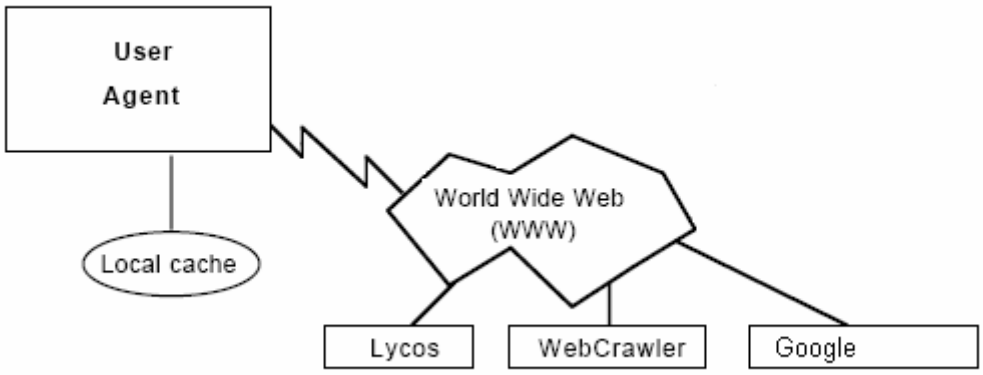

Abbildung 5: Architektur eines Internetagenten

Heterogenous Agent Systems: Dies sind Agentensysteme, die aus mehreren Agenten jeweils unterschiedlicher Agentenklassen bestehen.

Nwana räumt ein, daß es nicht immer gelingen kann, einen Agenten eindeutig in diese Typologie einzuordnen. Insbesondere im Bereich von Agenten, deren Anwendungsbereich das Internet ist. Diese lassen sich meist aufgrund ihres Wirkungsbereiches als Internetagenten klassifizieren. Dabei kann es jedoch ebenso richtig sein, sie aufgrund ihrer Architektur als Collaborative Agents oder Interface Agents zu bezeichnen.

2.3.2. Informationsagenten

Wie bereits im den Kapitel 2.2. und 2.3. erläutert, besteht bei der Suche nach Informationen im WWW die Schwierigkeit, die relevanten Informationen zu finden und aus ihnen Wissen zu extrahieren. Eine weitere Herausforderung ist die sinnvolle und nützliche Präsentation der gefundenen Informationen, sowie auch ein auf den Nutzer individuell zugeschnittenes, also personalisiertes Vorgehen bei diesen Aufgaben.
Dabei wurden bereits die Probleme erörtert, die sich in diesem Zusammenhang ergeben, wie die Masse der Informationen, die Verteilung, die Flüchtigkeit und die Vielfalt der Daten.

Bisher werden für die Informationssuche meist Web-Directories oder Suchmaschinen eingesetzt. Diese lösen jedoch nur annähernd die beschriebenen Probleme.
Hier kommen Informationsagenten zum Einsatz. Diese bieten eine intelligente Unterstützung, um für einen bestimmten Nutzer relevante Informationen zu finden. Informationsagenten können die Funktionen von Suchmaschinen und Web-Directories auf sinnvolle Weise ergänzen und erweitern und so die Suche im WWW verbessern.

Nach Klusch (2000) sind Informationsagenten herkömmlichen Suchmaschinen deutlich überlegen.
Sie helfen Informationssuchenden

a) beim Erwerb und Management von Informationen,
b) bei der Synthese und Präsentation von Informationen,
c) durch intelligente, individuelle Benutzerassistenz.

Dabei können sie durchaus herkömmliche Suchmaschinen in ihre Architektur integrieren und auf deren Ergebnissen aufbauen.
Informationsagenten sind jedoch nicht nur auf den Einsatz im Internet beschränkt. Nach Klusch (2000) ist ein Informationsagent ein Softwareagent, der autonom und proaktiv nach relevanten Informationen in einer Menge von verschiedenen, eventuell geographisch verteilten Informationsquellen sucht.

Ein wichtiger Aspekt von Informationsagenten ist die persönliche, individuell auf den Benutzer zugeschnittenene Unterstützung, die diese bei der Informationssuche bieten.
Godoy und Amandi (2005) heben die Bedeutung der Personalisierung bei Informationsagenten besonders hervor. Die Informationen, Interessen und Vorlieben des Benutzers werden innerhalb des Agenten durch ein Nutzerprofil repräsentiert. Informationsagenten, die über ein solches Nutzerprofil verfügen, bieten ihren Benutzern erweiterte Möglichkeiten mit der großen Menge an Informationen, die im Web verfügbar sind, umzugehen. So können sie Informationen direkt auf die Bedürfnisse des Benutzers zugeschnitten suchen, filtern und managen.
Je mehr ein Informationsagent dabei über die Interessen, Angewohnheiten und Vorlieben seines Benutzers weiß, desto besser kann er seine Aufgaben erledigen. Daher ist der

Aufbau, die Verwaltung und Aktualisierung des Nutzerprofils von besonderer Wichtigkeit.
Godoy und Amandi (2005) identifizieren drei Möglichkeiten, ein Nutzerprofil aufzubauen:

- durch direkte Programmierung über den Benutzer (*User-Programming*),
- Programmierung des Agenten für eine spezielle Aufgabe durch einen Knowledge Engeneer (*Knowledge-Engeneering*) und
- durch Techniken des maschinellen Lernens (*Machine-Learning*).

Die meistgenutzte Methode ein Nutzerprofil aufzubauen ist die Anwendung von maschinellem Lernen. Diese Methode hat den Vorteil, daß das Nutzerprofil dynamisch veränderbar ist und sich an den Benutzer anpassen kann. Zudem erfordert diese Technik kein Expertenwissen seitens des Benutzers.
Durch maschinelles Lernen ist es dem Agenten möglich, aus dem Benutzerverhalten und den Informationen, die ihm der Benutzer zur Verfügung stellt, die Interessen und Vorlieben des Benutzers zu lernen und daraus auf zukünftige Aktionen zu schließen.
Ein Nutzerprofil befähigt einen Agenten mit Informationen umzugehen, die ihm bisher unbekannt waren. Beispielsweise ist er in der Lage, Dokumente aufgrund eines Nutzerprofils zu klassifizieren, auch wenn der Agent nicht die Möglichkeit hat, die volle Bedeutung des Dokumenteninhaltes zu erschließen.

Wie bereits erwähnt, können die meisten Informationsagenten aufgrund ihrer Architektur auch einer der im vorherigen Abschnitt vorgestellten Klassen zugeordnet werden. So gibt es beispielsweise Informationsagenten in Multiagentensystemen, die nach der Klassifizierung von Nwana (1996) den Collaborative Agents zuzuordnen wären. Diese bezeichnet Klusch (2000) als kooperative Informationsagenten. Ebenso gibt es reaktive, deliberative und hybride Informationsagenten, Informationsagenten, die aus Benutzerverhalten lernen oder sich an die Strukturen ihrer Umgebung anpassen.

Das Multiagentensystem MySpiders von Pant und. Menczer (2002), das im Kapitel „Bestehende Ansätze und Systeme“ vorgestellt wird, dessen evolutionärer Ansatz die Grundlage des in dieser Arbeit erarbeiteten Agentensystems bildet, besteht aus Informationsagenten. Diese zählen jedoch ebenso zu den kooperierenden Agenten, die aus ihrer Umgebung lernen und sich anpassen.

2.3.3. Multiagentensysteme

Ein Multiagentensystem (MAS) ist ein Verbund von mehreren Agenten, die zusammenarbeiten, um ein gemeinsames Ziel zu erreichen. Dabei ist festzuhalten, daß durch die Zusammenarbeit der Agenten im System Aufgaben gelöst werden können, die über die Fähigkeiten eines einzelnen Agenten hinausgehen (Sycara, 1998).

Die Eigenschaften eines Multiagentensystems sind nach Sycara (1998) folgende:

- Jeder einzelne Agent des Systems hat nur unzureichende Informationen oder Fähigkeiten, um das gemeinsame Ziel zu erreichen. Daher hat jeder Agent eine eingeschränkte Sicht auf das Gesamtsystem.
- Es existiert keine globale Kontrollinstanz.
- Daten werden dezentral verwaltet.
- Berechnungen geschehen asynchron.

Multiagentensysteme wurden zuerst im Forschungsgebiet der verteilten künstlichen Intelligenz (Distributed Artificial Intelligence, DAI) untersucht. Ein Schwerpunkt der DAI-Forschung ist unter anderem Verteiltes Problemlösen (Distributed Problem Solving). Diese Forschung bildet die Grundlage für die Kooperation von Agenten in einem MAS. In einem klassischen DAI-System verfügen alle Agenten über ein gemeinsames Wissen. Außerdem ist die Methode zur Lösung des Problems zentral vorgegeben. Dies widerspricht der obigen Definition von Sycara. Oft werden jedoch beide Arten von Systemen vermischt, so daß auch Systeme mit gemeinsamem Wissen und zentraler Organisation als Multiagenten Systeme bezeichnet werden.

MASs werden vor allem eingesetzt, um Probleme zu lösen, die zu komplex oder zu aufwendig sind, um von einem einzelnen Agenten gelöst zu werden.
Bei einem zentralisierten System können außerdem Performanz- oder Ressourcen-Probleme auftreten, die man durch Einsatz eines MAS umgehen kann. Durch die Aufteilung in voneinander unabhängig arbeitende Agenten, können Aufgaben parallel ausgeführt und so Flaschenhälse vermieden werden. Ausfälle von einzelnen Agenten bei kritischen Fehlern können eher verkraftet werden als bei einem zentralisierten System. Ein weiterer Vorteil ist die einfache Skalierbarkeit eines MAS. Oft kann eine Anwendung einfach erweitert werden, indem man die Anzahl der Agenten im System erhöht oder der Aufgabe entsprechend ihre Fähigkeiten ergänzt.
Ein Einsatz eines MAS bietet sich ebenso an, wenn in der Aufgabe die Struktur eines Systems von einzelnen autonomen miteinander interagierenden Komponenten bereits immanent vorhanden ist. Dies wäre z.B. bei einer Anwendung zur Koordination von Terminen der Fall.
Multiagentensysteme können ebenso verwendet werden, um bereits bestehende heterogene Systeme miteinander zu vernetzen. Dafür werden sogenannte Wrapper Agenten eingesetzt, die die Funktionalität der bisherigen Systeme kapseln und einheitliche Schnittstellen zur Verfügung stellen.

Eine weitere Einsatzmöglichkeit für ein MAS ist das Erfassen und Verarbeiten von Informationen aus verteilten Informationsquellen.
Dies ist beispielsweise im Internet der Fall. Die Modularität, die hohe Parallelität, die Möglichkeit der Wiederverwendung und dynamischen Anpassung von Agenten bieten ideale Voraussetzungen für den Einsatz von Multiagentensystemen im Internet. Wenn eine Anwendungsdomäne wie das Internet sehr komplex, heterogen, sehr groß und verteilt, nicht vollständig erfaßbar ist oder sich auf unvorhersehbare Weise ständig verändert, dann gehören MAS zu den besten Möglichkeiten damit umzugehen. Ein MAS bietet eine Anzahl von spezialisierten, autonomen, modularen Komponenten, die sich an ihr jeweiliges Problemfeld anpassen und ihr spezielles Einzelproblem perfekt lösen können.

Die in dieser Arbeit entwickelte Anwendung ist ein MAS bestehend aus einzelnen Informationsagenten. Die spezialisierten, autonomen und modularen Einheiten (die Agenten) des Systems sind Experten für das Finden und Klassifizieren von Dokumenten im Internet. Sie arbeiten parallel und unabhängig voneinander, daher beeinflussen Ausfälle von einzelnen Agenten nicht die Arbeit des Gesamtsystems. Bei der Vielzahl von ins Leere führenden Links, sogenannten „broken links“, ist dies ein Großer Vorteil, da Fehler durch nicht auffindbare Dokumente einfach ignoriert werden können.
Durch die dynamische Skalierbarkeit des Systems – Agenten vermehren sich während des Programmablaufes und sterben wieder – kann sich die Anwendung optimal an die jeweilige Umgebung anpassen und wird der aktuellen Verlinkungsstruktur der

Webdokumente gerecht. Regionen des Webs, in denen relevante Dokumente zu finden sind könne so voll ausgeschöpft werden.
Die Architektur des Systems wird in Kapitel 5 genauer erläutert.

2.3.4. BDI Agenten

Das BDI-Modell wurde ursprünglich im *Rational Agency Project* am Standford Research Institute in den 1980'er Jahren entwickelt. Es basiert auf der Arbeit "Intention, Plans, and Practical Reason" von Michael E. Bratman (1987), Professor für Philosophie in Stanford. In seiner Arbeit entwickelt Bratman ein philosophisches Modell menschlichen Handels, welches sich mit der menschlichen Entscheidungsfindung befaßt. Er entwickelt die "Planning Theory of Intention" in der er Absichten (Intentions) als Teile von Plänen, die zu Handlungen führen, darstellt. Ein wichtiger Teil seiner Untersuchungen bezieht sich auf das *Practical Reasoning*. Practical Reasoning bezeichnet das menschliche Vorgehen, um zu einer Entscheidung zu gelangen. Es ist ein Prozeß des Nachdenkens, der vollzogen wird, um zu entscheiden, was zu tun ist. Pratical Reasoning unterscheidet sich hierbei vom Prozeß des Theroetical Reasoning. Während Theoretical Reasoning das Wissen (Beliefs) eines Menschen verändert, beeinflußt Practical Reasoning sein Handeln.
Das menschliche Practical Reasoning besteht aus zwei Schritten: 1. Deliberation. In diesem Schritt wird entschieden, welches Ziel (Desire) erreicht werden soll. 2. Means-Ends Reasoning. In diesem zweiten Schritt wird ein Vorgehen entwickelt, wie dieses Ziel zu erreichen ist.
Diese Theorie des menschlichen Denkens und Handelns wird mit Hilfe des BDI-Modells auf deliberative Agenten angewendet. Hierfür wird der Agent mit den oben erwähnten mentalen Eigenschaften: Beliefs, Desires und Intentions (BDI) ausgestattet. Sie repräsentieren das Wissen, die Ziele und die Absichten des Agenten und ermöglichen ihm, das Practial Reasoning durchzuführen.
Das BDI-Modell wurde zuerst von Bratman selbst implementiert, in IRMA, der Intelligent Resource-bounded Machine Architecture (Bratman et al., 1988). Eine weitere frühe Implementation ist PRS, das Procedural Reasoning System von Georgeff und Lansky (1987).

2.3.5. Jadex

Jadex steht für „JADE extension" und ist eine Erweiterung der JADE-Agentenplattform um das BDI Modell.

JADE (Java Agent DEvelopement Framework) wurde am Telecom Italia Lab (TILAB) entwickelt und stellt eine FIPA[1]-konforme Agentenplattform bereit, die vollständig in Java implementiert wurde. JADE bietet ein Framework für die Entwicklung von agentenorientierter Software und Multiagentensystemen. Hierbei ist eine graphische Oberfläche zum Starten, zur Analyse und zum Debugging von Agenten integriert.
Die Agentenplattform kann auf mehrere Einzelsysteme verteilt werden und ist aufgrund der Implementation in Java auf sehr vielen Betriebssystemen einsetzbar.

[1] Die FIPA (Foundation for Intelligent Physical Agents) ist eine IEEE Computer Society Organisation, die Standards für heterogene und interagierende Agenten und agentenbasierte Systeme entwickelt.

Jadex (JADE eXtension) wurde an der Universtität Hamburg im Arbeitsbereich „Verteilte Systeme und Informationssysteme“ entwickelt.
Bisherige Agentenplattformen können in grob zwei Gruppen eingeteilt werden:

1. FIPA-kompatible Plattformen, die auf die Unterstützung von offenen Systemen und die Realisierung einer Middleware ausgerichtet sind,
2. Plattformen, deren Fokus auf der Unterstützung der Agenteneigenschaften wie Rationalität und Zielgerichtetheit liegt

Jadex versucht diese beiden Ansätze zu vereinen, indem es die JADE-Agentenplattform um das im vorherigen Absatz beschriebene BDI-Konzept erweitert. Beliefs, Goals und Plans werden in die Agentenarchitektur integriert. Beliefs können dabei alle möglichen Java-Objekte sein, sie werden in einer Belief Base gespeichert. Goals stellen die Zustände dar, die von einem Agenten erreicht werden sollen. Pläne stellen prozedurale Rezepte dar, die ausgeführt werden, um die Goals zu erreichen. Pläne können dabei auch die Beliefs ändern. Die Beliefbase speichert Beliefs und bietet eine vereinheitlichte Sicht auf das gesamte Wissen des Agenten. Auf Beliefs kann über die Beliefbase mit Hilfe einer OQL-ähnlichen Query Language zugegriffen werden.
Zur Verbesserung des BDI-Ansatzes wurden Goals als ein zentrales Konzept implementiert. In anderen BDI-Systemen tauchen Goals dagegen nicht explizit, sondern nur in Form eines speziellen Ereignisses (Event) auf.
Zur Entwicklung von Jadex Agenten werden objektorientiertes Software-Engeneering und Techniken wie Java und XML eingesetzt. So ist die einfache unkomplizierte Entwicklung von Agenten auch für nicht KI-ler möglich.

Da Jadex auf dem BDI-Konzept basiert, unterstützt es die Entwicklung von deliberativen Agenten. Die Architektur umfaßt jedoch ebenso auch reaktive Aspekte. Jadex ist ein Eventbasiertes System, da alle Aktionen die ein Agent ausführt, einen Event als Auslöser haben. Zum Beispiel kann die Bearbeitung eines Planes über einen Message-Event ausgelöst werden. Pläne können jedoch genauso auch über Änderungen in den Beliefs ausgelöst werden, hierbei werden interne Events ausgelöst. Daher sind Jadex Agents weder rein reaktiv noch rein deliberativ, sondern können zu den hybriden Architekturen gezählt werden.

3. Techniken

In diesem Kapitel wird die Methode der Klassifizierung vorgestellt. Zunächst wird das Vorgehen der Klassifizierung anhand von klassischen Datenbanken erläutert. Anschließend wird die Klassifizierung von Webdokumenten dargestellt.
Zudem werden verschiedene Techniken zur Klassifizierung vergleichend dargestellt. Schließlich wird erörtert, aus welchen Gründen die Latent Semantic Indexing Technik für das DocAssistant-System gewählt wurde.

3.1. Klassifizierung

In dieser Arbeit wird ein System vorgestellt, das in der Lage ist Textdokumente ihrem Inhalt nach zu klassifizieren. In diesem Kapitel wird daher die Methode der Klassifizierung näher erläutert.
Klassifizierung ist eine Technik des *maschinellen Lernens*. Als maschinelles Lernen wird die automatische Generierung von Wissen aus Erfahrung bezeichnet. Dabei lernt ein System anhand von Beispielen, Aussagen über allgemeinere Daten zu treffen.
„Classification is learning a function that maps (classifies) a data item into one of several predefined classes" (Fayyad et al., 1996). Die Methode der Klassifizierung wird also verwendet, um Objekte anhand ihrer Merkmale vordefinierten Klassen zuzuordnen. Wie bereits in Kapitel 2.2. dargestellt gehört die Klassifizierung zu den vorhersagenden Methoden, da mit ihrer Hilfe Vorhersagen über die Art eines Objektes getroffen werden können. Es werden historische Daten verwendet, um mit ihrer Hilfe ein Klassifikationsmodell aufzustellen. Das Modell dient dann dazu, zu bestimmen, welcher Klasse ein Objekt angehört.

3.1.1. Klassifizierung von Daten

Eine klassische Anwendung der Klassifizierung ist beispielsweise die Einteilung von Kunden aus einer Kundendatenbank in verschiedene Klassen. Zum Beispiel könnte eine Autoversicherung ihre Versicherungsnehmer in verschiedene Risikoklassen einstufen, je nach dem, wie alt diese sind, welchen Wagentyp sie fahren und in wieviele Unfälle sie schon verwickelt waren.
Für die Klassifizierung wird nun nach einer Funktion f gesucht, die es erlaubt, aufgrund der gegebenen Merkmale (Alter, Wagentyp, usw.) einen Kunden einer bestimmten Risikoklasse zuzuordnen.
Die Funktion f stellt also das Data-Mining-Modell dar: $f(x_1, x_2, x_3) = y$. Wird f für eine Klassifikationsaufgabe gebraucht, so wird f auch Klassifikator genannt. Die abhängige Variable y ist in diesem Beispiel die Risikoklasse. Die einzelnen Kundenmerkmale (x_1, x_2, x_3) bilden die Eingabeparameter.
Um den Klassifikator zu erstellen wird die vorhandene Menge an Kundendaten zweigeteilt, in eine Trainingsmenge und eine Testmenge. Durch die Testmenge wird der Klassifikator anschließend evaluiert.
Die Funktion f wird durch den Data-Mining-Algorithmus erstellt, der für die vorliegende Aufgabe am besten geeignet ist.

Für die Klassifizierung existieren eine Reihe von verschiedenen Techniken, mit deren Hilfe sich der Klassifikator erstellen läßt. Sie werden in Kapitel 3.2. näher beschrieben.

Clustering

Ein ähnliches Verfahren ist das Clustering von Daten. Clustering-Methoden teilen eine Menge von Daten in Partitionen auf, in sogenannte Cluster. Die Cluster werden so gebildet, daß einander ähnelnde Daten den gleichen Clustern zugeordnet werden. Die so gebildeten Cluster können dann wiederum für die Klassifizierung von Daten in die Cluster verwendet werden.

Der Unterschied zur Klassifizierung besteht folglich darin, daß die Klassen bzw. Cluster beim Clustering automatisch gebildet werden, während sie beim Klassifizieren von Hand angelegt werden. Für die Bildung des Klassifikators müssen zudem Trainingsbeispiele von Hand in die jeweiligen Klassen eingeordnet werden.
Daher zählen Klassifizierungsmethoden auch zu den *überwachten Lernmethoden* (*supervised learning*), wohingegen Clustering zu den *unüberwachten Lernmetohden* (*unsupervised learning*) gehört.

Es existieren verschiedene Techniken für das Clustering von Daten. Bekannte Verfahren sind beispielsweise das k-Nearest-Neighbour Verfahren (Lu und Fu, 1978) und das Fuzzy-Clustering Verfahren (Höppner et al, 1999).
Clustering Verfahren unterscheiden sich auch darin, ob ein Datenobjekt zu mehreren Clustern gehören darf oder nur zu einem Cluster. Zudem gibt es auch hierarchisch aufgebaute Cluster. Hier können auch Cluster innerhalb von Clustern existieren.

In Bezug auf Webdokumente wird Clustering häufig zur Erzeugung von hierarchischen Strukturen, ähnlich denen von Web-Directories, eingesetzt. Diese bieten Vorteile beim Web-Document-Retrieval und können Benutzer bei der Suche nach Dokumenten unterstützen. Hierarchische Cluster können auch als Ontologien betrachtet werden, die auf Subkonzept-Relationen basieren.

3.1.2. Klassifizierung von Webdokumenten

Wie bereits in Kapitel 2.2.3 erwähnt, besteht beim Web-Mining, im Gegensatz zum Data-Mining in Datenbanken, wo Datenobjekte mit homogenen Attributen verarbeitet werden, die Schwierigkeit, daß die vorhandenen Dokumente sehr unterschiedlich sind. Um Web-Dokumente mit herkömmlichen Mining-Techniken behandeln zu können und sie klassifizieren zu können, müssen sie vorher aufbereitet werden. Dies geschieht wie beim klassischen Text-Information-Retrieval durch Extraktion von einzelnen Termen aus den Dokumenten. Terme sind meist einzelne Worte, die in dem Text vorkommen, es können aber auch mehrere Worte oder ganze Sätze sein. Bei Webdokumenten können zusätzlich HTML-Tags eine Rolle spielen. Die extrahierten Terme sind die für das Information Retrieval bzw. die Web-Mining-Aufgabe relevanten Merkmale eines Dokumentes. Dieser Vorgang wird auch als Feature Extraction bezeichnet.

Feature Extraction

Es existieren, je nach Anwendungsgebiet, verschiedene Vorgehensweisen bei der Extraktion von Features.

Die meistgenutzte Vorgehensweise ist die Extraktion aller Worte eines Textes. Die auf diese Weise erhaltene Feature-Menge kann dann durch weitere Methoden noch reduziert werden. Beispielsweise werden häufig nur die Nomen aus dem Text extrahiert, Verben und Adjektive werden ignoriert und Formatierungen werden herausgefiltert.
Weiterhin wird meistens eine Liste von Worten gebildet, die für den Retrievalvorgang nicht relevant sind (Stopwords). Diese Stopwords werden ebenfalls herausgefiltert.
Um die Anzahl an verschiedenen Features zu verringern und ein besseres Vergleichen von Termen zu ermöglichen, wird zudem sogenanntes Wordstemming eingesetzt. Dies bedeutet einzelne Worte auf ihren Wortstamm zu reduzieren. Z.B. wird aus „Information" „Inform". Dadurch werden alle verwandten Worte wie „informieren", „informiert", „Informationen" durch ein einziges Feature abgedeckt. Oft ist jedoch nicht nur das Auftauchen eines Terms in einem Dokument relevant sondern auch die Häufigkeit seines Auftretens oder die Position des Terms im Text. Daher muß je nach Anwendungsfall eine Form der Featurerepräsentation gefunden werden, in der die entsprechenden Informationen nicht verloren gehen.

Methoden die, wie oben beschrieben, einzelne Worte als Feature-Werte wählen, beachten dabei nicht das zusammenhängende Auftreten von bestimmten Worten, also die semantischen Korrelationen. Beispielsweise hat der Begriff „Hamburger Dom" eine andere Bedeutung als die beiden Worte „Hamburger" und „Dom" für sich alleine stehend.
Daher gibt es auch Ansätze, die nicht nur einzelne Worte betrachten, sondern sogenannte *n*-Gramme oder auch ganze Sätze als Feature-Werte extrahieren. Ein *n*-Gram ist eine Sequenz von *n* aufeinander folgenden Buchstaben in einem Wort, wobei *n* eine natürliche Zahl ist. Als *n*-Gramme können jedoch auch n aufeinander folgende Worte bezeichnet werden.

Obwohl durch die Extraktion von n-Grammen oder Sätzen der semantische Zusammenhang der Worte besser berücksichtigt wird als bei reinen Wortextraktionen, zeigen Experimente aus dem Bereich der Klassifikation und dem Information Retrieval, daß durch größere Komplexität der Features, die Effektivität des Systems nicht verbessert werden kann. Es werden teilweise sogar schlechtere Ergebnisse erzielt (Sebastiani, 2002).
Die Ursache hierfür ist darin zu sehen, daß statistische Methoden bei der Klassifikation und beim Information Retrieval eine große Rolle spielen. Feature-Werte, die aus einzelnen Worten bestehen, haben hier mehr Gewicht, da die Frequenz von Worten in einem Text höher ist als die von Sätzen oder *n*-Grammen.

Die in dieser Arbeit vorgestellte Klassifikations-Methode *LSI (Latent Semantic Indexing)* umgeht die Nachteile, die bei der Extraktion von n-Grammen oder ganzen Sätzen entstehen, ermöglicht aber dennoch den Erhalt semantischer Korrelationen im Text. Dies wird durch eine spezielle Reduktions-Methode der Feature-Werte ermöglicht. Diese Methode wird in Kapitel 3.2.4. Latent Semantic Indexing näher erläutert.

Bei der Klassifikation entsprechen die Features den Eingabeparametern (x_1, x_2, ..., x_n) der Funktion f, wie im vorherigen Abschnitt 3.1.1. beschrieben.

Feature Repräsentation

Eine häufig eingesetzte Methode zur Darstellung von Features ist die Vektordarstellung. Dabei werden die einzelnen Features als Elemente eines Vektors aufgefasst. Hierfür

existieren zwei verschiedene Methoden, die als Sets-of-words und Bags-of-Words bezeichnet werden.
Die Sets-of-words Methode gibt nur an, ob ein Wort in einem Dokument vorhanden ist oder nicht. Beispielsweise würde ein Dokument welches die Features „Eier", „Schinken" und „Frühstück" enthält als Vektor (Eier, Schinken, Frühstück) repräsentiert werden.
Bei der Bags-of-Words Methode wird außerdem noch gezählt, wie häufig ein Term in dem Dokument vorkommt. Beispielsweise kommt das Wort „Eier" dreimal in dem Text vor, das Wort „Schinken" einmal und „Frühstück" zweimal. Dann wäre eine mögliche Bags-of-Words Vektorrepräsentation des Dokumentes (3, 1, 2).
Bei keiner der beiden Methoden spielt die Reihenfolge der Worte eine Rolle. So ist es unwichtig, ob ein Wort ganz am Anfang des Dokumentes vorkommt oder erst am Ende.

Das Vektorraummodell

Im Vektorraummodell (engl. vector space model) werden Dokumente als Vektoren in einem hochdimensionalen, metrischen Vektorraum repräsentiert. Die Anzahl der Dimensionen entspricht dabei der Anzahl der Features. Beispielsweise wäre das oben beschriebene Dokument (3, 1, 2) ein Vektor in einem dreidimensionalen Vektorraum.
Bei der Klassifizierung von Dokumenten werden zum Aufbau eines Dokumentvektorenraumes die Dokumente der Trainingsmenge eingesetzt. Jedes Dokument bildet durch seine Vektor-Repräsentation einen Punkt im Vektorraum. Faßt man die Vektor-Repräsentationen aller Dokumente, die einer Klasse angehören, zusammen, erhält man das Feature Set der Klasse. Anhand dieses Feature Sets lassen sich, durch ein entsprechendes Modell, weitere Dokumente der Klasse zuordnen.
Weiterhin ist für jeweils zwei Dokumente eine Distanzfunktion definiert, die die Ähnlichkeit zweier Dokumente zueinander darstellt. Es gibt hierfür verschiedene Ansätze, die Distanz zweier Dokumente zu definieren. Eine häufig verwendete ist die Cosinus-Distanzfunktion. Dabei ist die Distanz zwischen den Vektoren x und y der Cosinus des Winkels $\alpha(x,y)$ zwischen x und y.

3.2. Techniken zur Klassifizierung

3.2.1. Naive Bayes

Naive Bayes ist eines der bekanntesten Klassifikationsverfahren und basiert auf einem probabilistischen Modell.
Grundlage zur Erstellung eines Naive Bayes Klassifikators ist das Bayessche Theorem zur

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Berechnung bedingter Wahrscheinlichkeiten (siehe u.a. Mitchell, 1996).

Mit Hilfe des Theorems läßt sich die bedingte Wahrscheinlichkeit des Eintretens eines Ereignisses A berechnen, unter der Bedingung, daß ein Ereignis B bereits vorher eingetreten ist.

Auf Dokumente und Dokumentklassen angewandt, läßt sich auf diese Weise auch die Wahrscheinlichkeit P (C_j | d) bestimmen, die Wahrscheinlichkeit, mit der man das Dokument d der Klasse C_j zuordnen kann. Anders ausgedrückt, bestimmt P (C_j | d) die Wahrscheinlichkeit, daß man die Klasse C_j als Ergebnis bekommt, wenn man Dokument d als Eingabe betrachtet. Berechnet man diese Wahrscheinlichkeit für alle Klassen C_n , so kann man das Dokument d der Klasse zuordnen, für die P (C_j | d) maximal ist. Daraus folgt, die beste Klasse für ein Dokument d ist:

$$\underset{C_j}{\operatorname{argmax}}\ P(C_j|d) = \frac{P(d|C_j)\ \ P(C_j)}{P(d)}$$

Die einzelnen Wahrscheinlichkeiten P (d | C_j) und P (C_j) lassen sich dann relativ einfach berechnen. Praktischer Weise entfällt die Berechnung der Wahrscheinlichkeit P(d), wie weiter unten erläutert.
Die einzelnen Wahrscheinlichkeitswerte werden anhand der Dokumente in der Trainingsdatenmenge ermittelt.
P (d | C_j) ist die Wahrscheinlichkeit, daß sich das Dokument d in der Klasse C_j befindet, wenn man nur die Klasse C_j betrachtet. Diese Wahrscheinlichkeit läßt sich als Schätzwert aus den Trainingsdatensätzen wie folgt ermitteln.

Der Inhalt des Dokumentes d wird durch seine Feature Werte repräsentiert. Diese können, wie im Kapitel 3.1.2. erläutert, durch verschiedene Methoden ermittelt werden. Nehmen wir an, das Dokument d wird durch die in ihm vorkommenden Worte repräsentiert. Die Wahrscheinlichkeit P (d | C_j) läßt sich dann durch die Einzelwahrscheinlichkeiten der Worte w_i in d berechnen.

$$P(d|C_j) = P(w_1|C_j) \bullet P(w_2|C_j) \bullet P(w_3|C_j) \bullet \ldots \bullet P(w_n|C_j)$$

P (w_i | C_j) ist dabei das Verhältnis der Anzahl der Vorkommen von w_i in den Trainingsbeispielen von C_j zu der Anzahl aller auftretenden Worte in den Trainingsbeispielen der Klasse C_j.
Am einfachsten verdeutlichen läßt sich dies anhand eines Beispiels:
Nehmen wir an, in der Klasse mit dem Label “Reisen” befinden sich Trainingsdokumente mit insgesamt tausend verschiedenen Worten (Features). Das Wort w_1 “Türkei” taucht dort fünfzig Mal auf. Somit wäre P (w_1 | Reisen) = 50 / 1.000.

P (C_j) läßt sich ebenso einfach aus den Trainingsbeispielen berechnen. P (C_j) ist die a priori Wahrscheinlichkeit, daß ein Dokumente der Klasse C_j angehört. Dies ergibt sich aus dem Verhältnis der Anzahl der Worte in C_j zu der Anzahl der Worte der gesamten Trainingsdatenmenge.
Nehmen wir an, im obigen Beispiel befinden sich in der gesamten Trainingsdatenmenge insgesamt 20.000 Worte. Für die Wahrscheinlichkeit P (Reisen) würde dann gelten: P (Reisen) = 1.000 / 20.000.

Die Wahrscheinlichkeit P(d) braucht nicht berechnet zu werden, da sich der Wert für die verschiedenen Klassen nicht verändert, sondern konstant bleibt. Daher kann man ihn außer acht lassen. Also lautet die neue Formel zur Ermittlung der richtigen Klasse für ein Dokument d:

$$\underset{C_j}{\operatorname{argmax}}\ P(C_j|d) = P(d|C_j)\ \ P(C_j)$$

Vor- und Nachteile von Naive Bayes

Die obige Vereinfachung zur Berechnung von P (d | C_j) enthält eine Annahme, die diesem Klassifikationsverfahren die Bezeichnung "naive" eingebracht hat. Zur Berechnung von P (d | C_j) wird angenommen, daß die Featurewerte, also die Worte des Dokumentes, unabhängig voneinander im Dokument vorkommen. Diese *konditionale Unabhängigkeitsannahme* ist problematisch, da sie offensichtlich in natürlichsprachlichen Texten nicht der Wirklichkeit entspricht. Beispielsweise wird das Wort "Grüßen" in einem Text sehr viel häufiger zusammen mit dem Wort "freundlichen" auftreten als mit anderen Worten. Daher sind diese beiden Worte sicherlich nicht unabhängig voneinander. Obwohl die Unabhängigkeitsannahme nicht haltbar ist und die darauf begründeten Wahrscheinlichkeitswerte verfälscht sind, zeigt die Naive Bayes Methode trotzdem relativ gute Ergebnisse bei der Klassifikation (Domingos und Pazzani, 1997).

3.2.2. Decision Trees

Beim Klassifizieren mit Decison Trees (Entscheidungsbäumen) werden Klassifikationsregeln in Form von Attribut-Wert-Paaren aus den Trainingsbeispielen geschlossen. Die Attribute werden in der Erwartung ausgewählt, daß durch sie die Trainingsdatenmenge in eine möglichst homogene Teilmenge aufgespaltet werden kann. Es wird eine Baumstruktur erzeugt, indem jeder Knoten mit einem Attribut der Daten assoziiert wird. Die Zweige, die von dem jeweiligen Knoten ausgehen, werden mit den möglichen Werten der Attribute beschriftet. Jeder Knoten beinhaltet einen Test, bezogen auf die Merkmale der zu klassifizierenden Daten. Die Daten durchlaufen den Baum von der Wurzel bis zu den Blättern des Baumes und absolvieren an jedem Knoten den Test auf das jeweilige Attribut. Anhand des Tests wird entschieden, welcher Zweig des Baumes weiterverfolgt wird. Die Blätter des Baumes entsprechen den möglichen Klassen, in die die Daten eingeordnet werden sollen. Ein Datensatz durchläuft so den Baum von der Wurzel bis zu der jeweiligen Klasse, in die der Datensatz eingeordnet wird. Auf diese Weise findet eine Klassifikation der Daten statt.

Eine der populärsten Decision Tree Methoden ist das C4.5 Verfahren (Quinlan, 1993), welches auf einer Verbesserung des ID3 Algorithmus basiert.

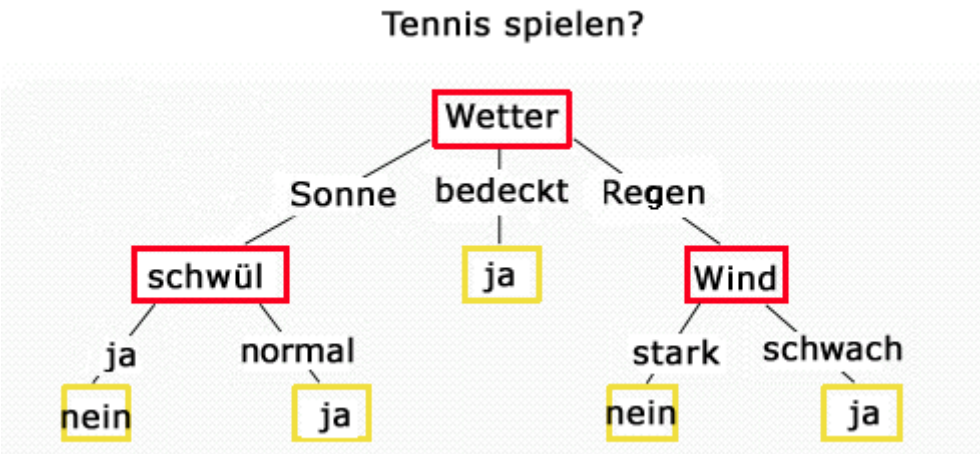

Abbildung 6: Decision Tree zur Bewertung von Wetterlagen auf ihre Eignung zum Tennis spielen.

Vor- und Nachteile der Decision Tree Methode

Ein Vorteil des Decision Tree Verfahrens ist sicherlich, die niedrige Komplexität, die es dem Menschen leicht macht, die Klassifikation einzelner Datensätze nachzuvollziehen.
Es wird jedoch relativ viel Speicherplatz für den Entscheidungsbaum gebraucht und der Verwaltungsaufwand des Baumes ist hoch.
Weiterhin zeigen Experimente, daß die Decision Tree Methode bei Bäumen mit großem Merkmalsraum sehr häufig zum overfitting neigt. Das heißt, daß die Klassifizierung für die Trainingsdatenmenge korrekt ist, jedoch bei unbekannten Daten eher schlechte Ergebnisse aufweist. (Mitchell, 1997)

3.2.3. Support Vector Machines

Die Methode der Support Vector Machines (SVM) geht auf V. Vapnik zurück, der 1974 eine Arbeit über die Prinzipien der Hyperebenentrennung veröffentlichte (Vapnik und Chervonenkis, 1974). Die Klassifikation mit Hilfe von SVM geschieht durch das Finden einer Hyperebene, die den Merkmalsraum optimal in zwei Klassen teilt. Optimal bedeutet hierbei, die beiden Klassen so zu trennen, daß auch zukünftig zu klassifizierende Daten möglichst richtig klassifiziert werden. Dies ist genau dann der Fall, wenn sich zwischen den beiden Klassen ein möglichst breiter Trennspalt befindet. Die entsprechende Hyperebene befindet sich dann genau in der Mitte dieses Spalts und wird als optimal separierende Hyperebene bezeichnet.

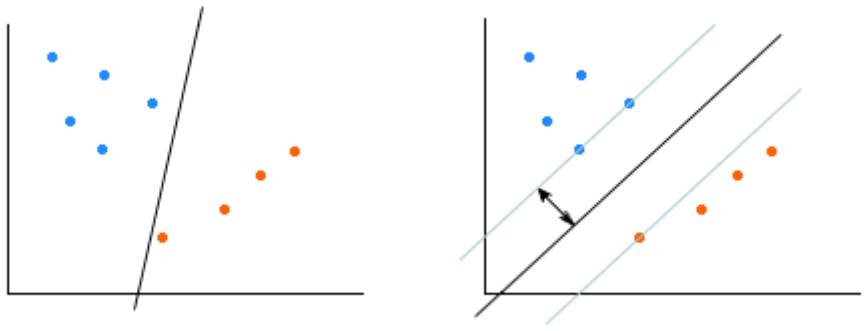

Abbildung 7:
1. Nicht optimale Hyperebene. 2. Optimal separiende Hyperebene

Die Trennspalte läßt sich mit Hilfe von nur zwei Vektoren festlegen. Diese begrenzen die Trennspalte zu beiden Seiten hin und werden Support Vectors genannt.

Das Finden der Funktion, die diese Hyperebene beschreibt, kann sehr schwierig sein, wenn die Ebene nicht linear separierbar ist. D.h. die beiden Klassen lassen sich nicht durch eine lineare Funktion trennen.

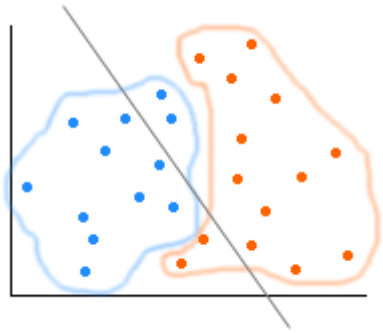

Abbildung 8: Nicht linear separierbare Klassen

Es wird daher ein Trick angewandt, der das Problem vereinfacht. Der Merkmalsraum wird mit Hilfe einer sogenannten *Kernelfunktion* in einen höherdimensionalen Raum transformiert. Dadurch wird es möglich, den Merkmalsraum mit einer linearen Funktion zu separieren. Diese Funktion zu finden, ist nun wesentlich einfacher als die des ursprünglichen Merkmalraumes.

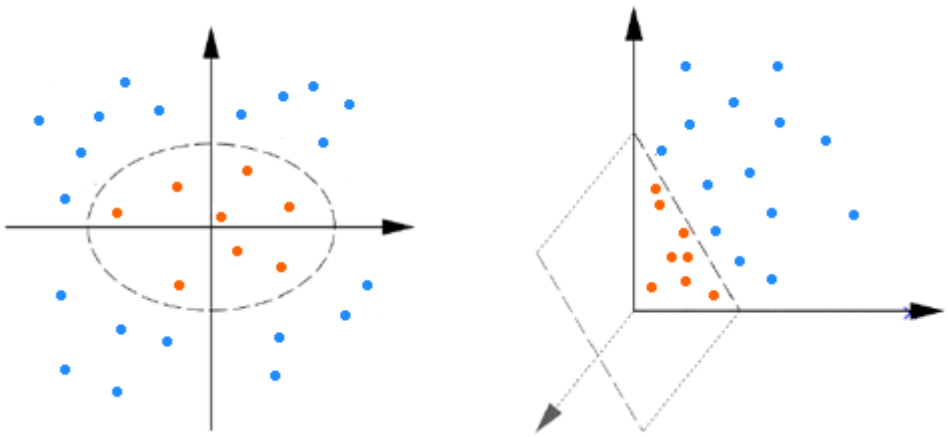

Abbildung 9: Lineare Separation durch Transformation in einen höherdimensionalen Merkmalraum.

Vor- und Nachteile von Support Vektor Machines

Ein Vorteil von SVM ist, daß sie sehr robust gegen Overfitting sind. Die Klassengrenzen, die sie erzeugen, sind linear und können über die Support-Vektoren definiert werden. Die Anzahl der Support-Vektoren ist im Verhältnis zur Größe der Trainingsmenge relativ klein.
SVM-Klassifikatoren erzielen gute Klassifikationsergebnisse, mit teilweise besseren Recall- und Precision-Werten als Naive Bayes oder C4.5 Klassifikatoren (Joachims, 1997).

Ein Nachteil ist jedoch der hohe Trainingsaufwand. Insbesondere bei nicht linear separierbaren Problemen ist der Rechenaufwand sehr hoch und das Verfahren komplex. Daher ist es besonders nachteilig, daß die SVM nicht inkrementell erweiterbar ist. Für neu dazugekommene Trainingsbeispiele muß die SVM jedesmal komplett neu erstellt werden.

Weiterhin ist mit einer SVM nur binäre Klassifikation möglich. D.h. eine SVM kann jeweils nur für zwei Klassen erstellt werden. Bei mehr als zwei Klassen steigt die Anzahl der benötigten SVM und somit auch der gesamte Trainingsaufwand.

3.2.4. Latent Semantic Indexing

Das Latent Semantic Indexing (LSI) Verfahren, auch Latent Semantic Analysis (LSA) genannt, wurde im Jahr 1990 von Scott Deerwester et al. in dem Artikel *Indexing by Latent Semantic Analysis* (Deerwester et al., 1990) veröffentlicht und ist von der Firma Telcordia Technologies (Bellcore) patentiert.
Außerhalb der Information-Retrieval Comunity ist LSI wenig bekannt und wird daher höchst selten für klassische Data-Mining Aufgaben eingesetzt. Für die Klassifizierung von Textdokumenten ist LSI jedoch ein vielversprechendes Verfahren. Es eignet sich daher auch gut für Webdokumente (Brian und Garzon, 2001). LSI zeichnet sich gegenüber anderen Verfahren unter anderem durch hohe Recall und Precision-Werte der Ergebnisse aus (Yu et al., 2003). Der Vorteil von LSI ist, daß ein Dokument nicht nur danach klassifiziert wird, ob bestimmte Worte in dem Dokument vorkommen oder nicht. Vielmehr ist es durch LSI möglich, ein Dokument einer Klasse zuzuordnen, wenn der Inhalt des Dokumentes in die Klasse paßt. Das Dokument wird auch dann korrekt einer Klasse zugeordnet, wenn das Dokument keines der Worte enthält, welche die Klasse beschreiben. Dies ist dadurch möglich, daß LSI nicht nur die Worte eines Dokumentes mit den Klassen-Bezeichnern vergleicht, sondern auch die Worte der anderen Dokumente die der Klasse angehören, miteinbezieht. Ein Dokument wird also einer Klasse zugeordnet, wenn es besonders viele Worte mit anderen Dokumenten dieser Klasse gemeinsam hat. Ein spezielles mathematisches Verfahren ermöglicht es LSI, den Inhalt eines Dokumentes maschinell zu Erfassen und einzuordnen, ohne daß das System wirklich verstehen muß, worum es in dem Dokument geht. In herkömmlichen Klassifizierungsverfahren werden meist einfach nur die Worte des Dokumentes mit den Klassen-Bezeichnern verglichen. Hier entstehen Probleme, wenn ein Wort in verschiedenen Zusammenhängen mehrere verschiedene Bedeutungen hat (Homonymie), oder wenn ein und der selbe Umstand mit verschiedenen Worten benannt werden kann (Synonymie). Es kann unter diesen Umständen häufig vorkommen, daß Texte aufgrund von bestimmten Worten einer Klasse zugeordnet werden, in die sie vom Inhalt her gar nicht hinen passen. Oder andersherum wird ein Dokument einer Klasse nicht zugeordnet, weil die entsprechenden Worte nicht in ihm vorkommen, obwohl das Dokument vom Inhalt her in die Klasse einzuordnen wäre. Durch LSI können diese Probleme vermieden werden.

Singular Value Decomposition

LSI basiert auf dem Verfahren der Singular Value Decomposition (SVD), zu deutsch Singulärwertzerlegung, einem mathematischen Verfahren, welches ursprünglich genutzt wurde, um schlecht konditionierte Gleichungssysteme[1] lösen zu können. Zudem wird es zur Lösung von Least-Squares Problemen eingesetzt. Bei dieser Form von Problemen wird zur Lösung eines linearen Gleichungssystems eine Approximation in einen niedriger-dimensionalen Raum vorgenommen (vgl. u.a. Heath, 2002). Dies wird beispielsweise für Bildkomprimierungsverfahren benutzt.

[1] Dies sind Gleichungssysteme, bei denen kleine Änderungen in den Eingabewerten zu großen Änderungen im Ergebnis führen (vgl. Deuflhard, 2002).

Mit Hilfe der Singulärwertzerlegung kann jede m x n Matrix A so in drei Matrizen $U_{(m \times m)}$, $S_{(m \times n)}$ und $V_{(n \times n)}$ zerlegt werden, daß gilt:

$$A = USV^T$$

Dabei sind die Spaltenvektoren der Matrizen U und V orthonormal, so daß gilt:

$UU^T = U^TU = I$ und $VV^T = V^TV = I$, wobei I die Einheitsmatrix ist.

Die Matrix S enthält als Diagonalelemente positive, der Größe nach geordnete Singulärwerte von A. Die Anzahl der von 0 verschiedenen Singulärwerte ist gleich dem Rang der Matrix A.

Weiterhin gilt:

Die Spalten von U sind die Eigenvektoren von AA^T.

Die Spalten von V sind die Eigenvektoren von A^TA.

Die Singulärwerte sind die Wurzeln der Eigenwerte von A^TA bzw. AA^T.[1]

Um SVD auf Dokumente anwenden zu können benutzt LSI das Vector-Space-Modell für die Darstellung von Dokumenten. Wie bereits im vorherigen Kapitel beschrieben, werden die Dokumente durch Feature-Extraktion in Vektordarstellung überführt.

Die Dimension des Vektorraumes, der durch diese Vektoren aufgespannt wird, entspricht der Anzahl der Terme, die durch die Vektoren repräsentiert werden. Nimmt man zur Anschauung einen dreidimensionalen Raum an, befinden sich in diesem Vektorraum einander ähnliche Dokumente nah beieinander, während einander unähnliche Dokumenten weiter auseinander liegen. Es bilden sich so Cluster von einander ähnelnden Dokumenten. Alle Vektoren werden zusammengefaßt in der sogenannten Term-Dokument-Matrix. Sie enthält alle Dokumente des Trainingsdatensatzes sowie alle bereits klassifizierten Dokumente. In dieser Matrix bilden die Vektordarstellungen der Dokumente die Spalten der Matrix. In den Zeilen läßt sich jeweils ablesen, welcher Term wie oft innerhalb eines Dokumentes vorkommt. Auf diese Matrix wird nun die Singulärwertzerlegung angewandt.

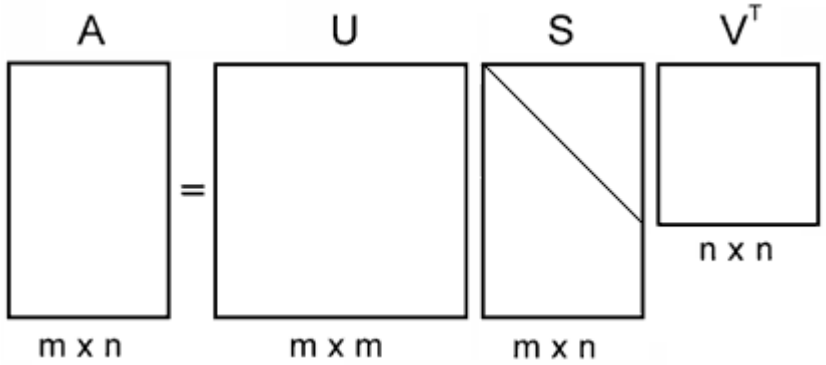

Abbildung 10: Schematische Darstellung der Singulärwertzerlegung

In gewisser Weise können in dieser Zerlegung die Zeilen der Matrix U als eine Repräsentation der Terme in Form von Termvektoren angesehen werden und die Spalten der Matrix V^T als eine Repräsentation der Dokumente in Form von Dokumenten-Vektoren. Die einzelnen Werte bilden die Koordinaten, über die sich die Dokumente und Terme im Vektorraum finden lassen. Über die Multiplikation mit den in der Matrix S abgebildeten Singulärwerten, findet eine Skalierung der Achsen des Vektorraumes statt.

[1] A^TA und AA^T haben die gleichen Eigenwerte.

Sie legen somit die Lage der Dokumente und Terme im Raum fest. (Deerwester et al., 1990)
Anhand dieser Singulärwertzerlegung wird nun die Term-Dokument-Matrix durch eine neue Matrix A_k (m x n) approximiert. Dies geschieht, indem nur die größten k Singulärwerte zur Berechnung von A_k verwendet werden. Damit erhält man:

$A_k (mxn) = U_k (mxk) \times S_k (kxk) \times V_k^T (kxn)$

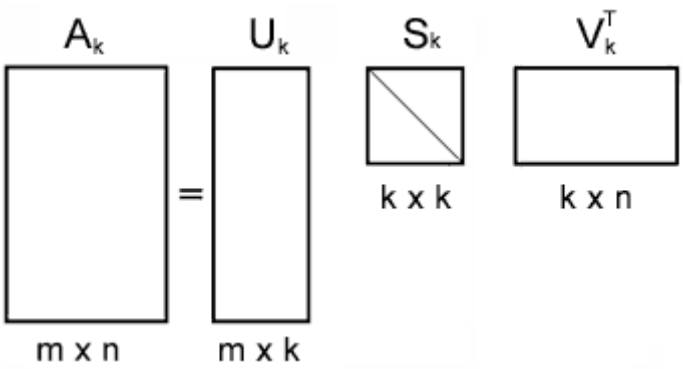

Abbildung 11: Approximierung von A durch Verwendung von k Singulärwerten

Dies wird auch als *truncated SVD* bezeichnet. Die Approximierung von A durch A_k entspricht einer Dimensionsreduzierung im Vektorraummodell. Ähneln sich zwei Dokumente, so basieren sie auf dem gleichen inhaltlichen Konzept. Durch die Dimensionsreduktion entstehen Linearkombinationen der Terme, die solche inhaltlichen Konzepte darstellen. Es werden so die latenten inhaltlichen Informationen (Latent Semantics) der Dokumente maschinell auswertbar gemacht. Daher der Name des Verfahrens: *Latent Semantic Indexing*. Dieser Kern-Schritt des LSI-Verfahrens ermöglicht die besagte Klassifizierung, ausgehend vom Inhalt der Dokumente und unabhängig vom Auftreten eines einzelnen Keywords in einem Dokument.

Ein Beispiel zur Dimensionsreduktion

Das Vorgehen der Dimensionsreduktion läßt sich am einfachsten anhand eines Beispiels in einem Vektorraum mit kleinerer Dimension zeigen.
Angenommen es gibt drei Dokumente a, b und c, repräsentiert durch jeweils zwei Features, also einem zweidimensionalen Vektor in einem zweidimensionalen Vektorraum:

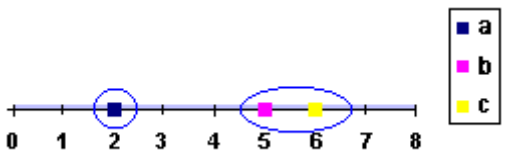

Abbildung 12: Beispiel in einem zweidimensionalen Vektorraum

Die Dokumente b und c liegen, wie man leicht erkennen kann, dicht beieinander, so daß sie einen gemeinsamen Cluster bilden, während das Dokument a etwas abseits liegt und einen eigenen Cluster bildet. Ziel von LSI ist es nun, die Dimension zu reduzieren und dabei die relative Anordnung der Dokumente zueinander möglichst zu erhalten. In

unserem Fall führen wir eine Dimensionsreduzierung auf Dimension d=1 durch. Dadurch ergibt sich Folgendes:

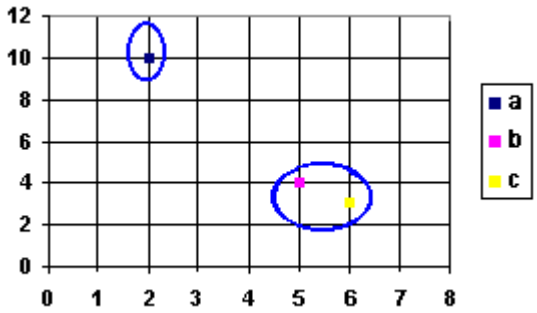

Abbildung 13: Beispiel Dimensionsreduktion auf eindimensionalen Vektorraum

Die Dimension wurde nun um eins reduziert, die Cluster sind jedoch erhalten geblieben. D. h. bei der Dimensionsreduktion sind keine relevanten Informationen verloren gegangen.
Nehmen wir z.B. an, die zwei Dimensionen würden jeweils für die Worte "Eier" und "Frühstück" stehen. Dann würde das Dokument a das Wort "Eier" 10 Mal enthalten und das Wort "Frühstück" nur einmal. Entsprechend enthalten die Dokumente b und c das Wort "Frühstück" fünf, bzw. sechs Mal, aber das Wort "Eier" nur vier bzw. drei Mal. Durch die Dimensionsreduktion bleiben nur die wichtigsten semantischen Inhalte erhalten, die unwichtigen Terme fallen weg. Also wurde das Dokument a auf "Eier" reduziert und die Dokumente b und c auf "Frühstück". Da die Dokumente b und c sich gemeinsam auf "Frühstück" beziehen, befinden sie sich in dem selben Cluster, während Dokument a sich alleine in dem "Eier"-Cluster befindet.
In Matrizen und Vektoren ausgedrückt ergibt das obige Beispiel folgendes:

Dokument **c** = $\begin{pmatrix} 6 \\ 3 \end{pmatrix}$ Dokument **b** = $\begin{pmatrix} 5 \\ 4 \end{pmatrix}$ Dokument **a** = $\begin{pmatrix} 1 \\ 10 \end{pmatrix}$

Term-Dokument-Matrix $\mathbf{A} = \begin{pmatrix} 6 & 5 & 1 \\ 3 & 4 & 10 \end{pmatrix}$

Die Singulärwertzerlegung an sich ist ein relativ kompliziertes Verfahren. Die Details sollen im Rahmen dieser Arbeit nicht weiter erläutert werden, da hier nur das Ergebnis von Bedeutung ist.[1] Es ergibt sich:

$$\underset{\mathbf{U}}{} \qquad \underset{\mathbf{S}}{} \qquad \underset{\mathbf{V}^T}{}$$

$$A = \begin{pmatrix} 0{,}4750 & 0{,}8799 \\ 0{,}8799 & -0{,}4750 \end{pmatrix} \times \begin{pmatrix} 12{,}284 & 0 \\ 0 & 6{,}008 \end{pmatrix} \times \begin{pmatrix} 0{,}4468 & 0{,}4798 & 0{,}7549 \\ 0{,}6415 & 0{,}4160 & -0{,}6441 \end{pmatrix}$$

[1] Eine detailierte Beschreibung des Verfahrens findet sich ua in (Golub und van Loan, 1989).

Nach der Dimensionsreduzierung auf k = 1 ergibt sich:

$$\mathbf{U}_k \qquad \mathbf{S}_k \qquad \mathbf{V}_k^T$$

$$A_k = \begin{pmatrix} 0{,}4750 \\ 0{,}8799 \end{pmatrix} \times \begin{pmatrix} 12{,}284 \end{pmatrix} \times \begin{pmatrix} 0{,}4468 & 0{,}4798 & 0{,}7549 \end{pmatrix}$$

$$= \begin{pmatrix} 2.6070 & 2{,}7999 & 4{,}4051 \\ 4{,}8292 & 5{,}1866 & 8{,}1601 \end{pmatrix}$$

Obwohl die neue Matrix A_k aus vollkommen anderen Werten besteht als die ursprüngliche Term-Dokument-Matrix A, enthält sie immer noch alle relevanten Informationen über die Anordnung der drei Dokumente im Raum. Diese Informationen lassen sich jedoch nicht mehr so einfach ablesen, wie dies in der Ursprünglichen Matrix A möglich war. Wie diese Informationen aus den drei Matrizen U_k, S_k und V_k^T gewonnen werden können, wird im folgenden Abschnitt erläutert.

Klassifizierung mit Hilfe von LSI

Um mit Hilfe von LSI ein Dokument klassifizieren zu können, muß der Inhalt zunächst, wie vorher beschrieben, in Vektordarstellung gebracht werden. Nehmen wir wieder das vorherige Beispiel als Grundlage, dann bilden die bereits klassifizierten Dokumente a b und c die Trainingsdatenmenge. Wir wollen beispielsweise ein neues Dokument q, welches das Wort “Eier” 0 mal aber das Wort “Frühstück” 12 mal enthält, klassifizieren.

$$\text{Dokument } \mathbf{q} = \begin{pmatrix} 0 \\ 12 \end{pmatrix}$$

Um das neue Dokument zu klassifizieren, muß es, wie die anderen Dokumente auch, in den dimensionsreduzierten Vektorraum gebracht werden. Nach Berry et al. (1995) geschieht dies durch das sogenannte Folding-In mit Hilfe der Formel:

$q' = q^T U_k S_k^{-1}$

q‘ ist dann ein Dokument-Vektor, der der Dokument-Matrix V_k^T als neue Spalte hinzugefügt wird. Die Formel ergibt sich durch Umstellung der ursprünglichen Formel wie folgt:

$A_k = U_k S_k V^T$

$V^T = A_k U_k S_k^{-1}$

$q' = q^T U_k S_k^{-1}$

In unserem Beispiel ist

$$q' = \begin{pmatrix} 0 & 12 \end{pmatrix} \times \begin{pmatrix} 0{,}4750 \\ 0{,}8799 \end{pmatrix} \times \begin{pmatrix} 0{,}08141 \end{pmatrix} = \begin{pmatrix} 0{,}85959 \end{pmatrix}$$

Dieser Vektor kann nun direkt mit den Dokument-Vektoren der Matrix V_k^T auf Ähnlichkeit verglichen werden. Da in diesem Fall die Vektoren nur eindimensional sind, können direkt die Werte miteinander verglichen werden. Es ergibt sich, daß q’ = 0,8595

dem Wert 0,7549 am nächsten kommt. Da dieser Wert für das Dokument a steht wird unser Dokument q in die selbe Klasse wie Dokument a eingeordnet. Da es in Dokument a wie auch in Dokument q hauptsächlich um Frühstück und nicht um Eier geht, ist diese Klassifizierung also richtig.

Generell wird für die Berechnung der Ähnlichkeit der Dokument-Vektoren die schon erwähnte Cosinus-Distanzfunktion verwendet. Dabei ist die Ähnlichkeit zwischen zwei Vektoren a und q der Cosinus des Winkels α(a,q). Dieser wird nach Berry et al. (1999) wie folgt berechnet:

$$\cos\theta = \frac{a^T q}{\|a\|_2 \, \|q\|_2}$$

wobei die Euklidische Vektornorm $\|x\|_2$ definiert ist als:

$$\sqrt{x^T x} = \sum_{i=1}^{t} x_i^2$$

für jeden reellen t-dimensionalen Vektor x.

Ein Dokument, welches klassifiziert werden soll, wird also wie oben beschrieben in den dimensionsreduzierten Vektorraum transformiert und dann mit Hilfe der Cosinus-Distanzfunktion mit allen bereits klassifizierten Dokumenten verglichen. Das Dokument wird dann in die Klasse des Dokumentes eingeordnet, welches ihm am ähnlichsten ist.

SVD Folding-In

Um der Trainingsmenge neue Dokumente hinzufügen zu können, ohne jedesmal die komplette Singulärwertzerlegung erneut durchführen zu müssen, wird das schon oben erwähnte SVD Folding-In verwendet. Neue Dokumente und Terme werden der Term-Dokument-Matrix, bzw. den Term und Dokument Matrizen U_k und V_k^T, einfach angehängt. Die Singulärwertzerlegung ist ein sehr berechnungsintensives Verfahren. Durch das Folding-In kann der Aufwand deutlich reduziert werden (vgl. u.a. Berry et al., 1995).
Um ein neues Dokument d einzufügen, wird die schon oben beschriebene Formel für die Anfragetransformation verwendet:

$d' = d^T U_k S_k^{-1}$

d' wird dann als neue Spalte der Dokument-Matrix V_k^T hinzugefügt.
Ebenso verfährt man beim Einfügen von neuen Termen. Ein neuer Termvektor t wird der Term-Matrix U_k durch die Entsprechende Formel:

$t' = t^T V_k S_k^{-1}$

als neue Zeile hinzugefügt.

Vorteile von LSI

LSI komprimiert die Feature-Vektoren der Dokumente und transformiert die semantischen Konzepte des Inhaltes in einem Raum mit niedrigerer Dimensionalität. Dies hilft Komplexität und damit Rechenzeit und Ressourcen bei der Klassifikation von Dokumenten zu sparen.
Dadurch, daß ein Dokument auf seine semantischen Konzepte reduziert wird, werden nur die charakteristischen Eigenschaften zur Klassifikation verwendet, Unwichtiges entfällt.

Homonyme können ausgehend von ihrem Kontext richtig klassifiziert werden, so daß beispielsweise keine Texte über den Zustand der Bänke in Parkanlagen in die Kategorie „Finanzen und Anlagemöglichkeiten“ eingeordnet wird, nur weil die Worte „Bank“ und „Anlage“ häufig in diesen Texten vorkommen. Dadurch wird ein hoher *Precision*-Wert bei der Klassifikation erreicht.

Außerdem werden umgekehrt auch Synonyme erkannt und richtig eingeordnet. Beispielsweise werden in die Kategorie Autos Texte mit Worten wie „Fahrzeug“, „KFZ“, „Kraftwagen“, „PKW“ und „Wagen“ eingeordnet, auch wenn das Wort Auto nicht explizit in den Texten genannt wird. Dadurch werden Dokumente richtig klassifiziert, die nicht über bestimmte, für die Klasse bezeichnende Keywords verfügen, aber trotzdem inhaltlich in die Klasse passen. Anders gesagt werden Klassen nicht, wie beispielsweise bei der Naive Bayes Methode, durch bestimmte Worte und deren Häufigkeit definiert, sondern über Kombinationen dieser Worte, nämlich über die vorher erwähnten semantischen Konzepte. Dies schlägt sich in einem verbesserten *Recall*-Wert gegenüber anderen Klassifikations-Methoden nieder.

Ein Vorteil gegenüber anderen bisher vorgestellten Techniken ist außerdem die Möglichkeit, die Term-Dokument-Matrix jederzeit durch neue Trainingsbeispiele erweitern zu können. Durch die Folding-In-Methode können neue Trainingsbeispiele hinzugefügt werden, ohne alle Matrizen komplett neu zu erstellen und den gesamten Vorgang der Singulärwertzerlegung erneut durchführen zu müssen. Dieses Vorgehen ist allerdings nur mit einer begrenzten Anzahl von neuen Dokumenten möglich. Beim Folding-In in die Term-Dokument-Matrix ergeben sich minimale Unterschiede zu der Matrix, die bei der originalen Singulärwertzerlegung entstehen würde. Beim Einfügen von sehr vielen Dokumenten in die Term-Dokument-Matrix summieren sich diese Fehler, so daß die Klassifikationsgüte leidet. Bei welcher Anzahl von Dokumenten und unter welchen Umständen dieser Fehler die Klassifikation beeinträchtigt, ist ein noch offenes Problem und hängt sicherlich auch davon ab, wie repräsentativ die original Dokumente and Terme für die Klassifikation waren, (Deerwester et al., 1990).

Nachteile von LSI

Die Komplexität der Singulärwertzerlegung beträgt in etwa $O(n^3)$, wobei n die Anzahl der Terme ist. Dadurch ist das Verfahren in der Trainingsphase, also beim Erstellen des Indexes, sehr rechenintensiv.
Es wurden jedoch in der Literatur bereits Methoden vorgeschlagen, um die Performanz bei der Dimensionsreduzierung zu erhöhen. Beispiele hierfür sind die *Monte-Carlo-Methode* (Frieze et al., 1998) oder die Verwendung von *Random Projection* (Papadimitriou et al., 1998) zur Projektion der Term-Dokument-Matrix auf einen niedrigdimensionalen Unterraum.

Ein bisher offenes Problem stellt die Wahl des Wertes k bei der Singulärwertzerlegung dar. Der Wert k steht für die Anzahl der Dimensionen, auf die die Term-Dokument-Matrix reduziert werden soll.
Dieser Wert wird meist anwendungsspezifisch ausgewählt, indem evaluiert wird, bei welchem Wert das LSI-Verfahren die besten Ergebnisse erzielt. Leider gibt es bisher keinen allgemeinen Wert für k, der sich im Vorfeld festlegen ließe.

3.3. Einsatz von LSI im Rahmen dieser Arbeit

In der im Rahmen dieser Arbeit entstandenen Anwendung wird LSI als Klassifikationstechnik eingesetzt. Es existiert eine Fülle von verschiedenen Techniken zur Klassifikation von Daten. Einige dieser Techniken wurden im vorherigen Kapitel beschrieben. LSI erschien für diese Anwendung am vielversprechendsten.

Einer der wichtigsten Gründe, die für den Einsatz von LSI im Rahmen dieser Arbeit sprechen, ist, daß LSI, im Gegensatz zu den anderen hier vorgestellten Verfahren, speziell für Text-Retrieval konzipiert ist. Daher wird der Information Retrieval Ansatz der Arbeit durch den Einsatz von LSI am besten unterstützt.

LSI bietet außerdem einige spezielle Vorteile gegenüber anderen Klassifizierungsverfahren, die an dieser Stelle noch einmal zusammengefaßt werden sollen.

Klassifikationstechniken, die auf der Indizierung von einzelnen Worten oder Termen beruhen, führen oft zu sehr großen, hochdimensionalen Feature-Räumen. Die Anzahl der Dimensionen steigt mit der Anzahl der Worte und somit auch mit der Anzahl der Dokumente in der Trainingsdatenmenge. Dies geht einher mit hohem Speicherverbrauch und hoher Rechenzeit, sowohl in der Trainingsphase als auch bei der Klassifikation. Außerdem steigt die Gefahr des sogenannten *overfitting*, der Überanpassung eines Modells an eine vorgegebene Datenmenge, bei hochdimensionalen Feature-Räumen. Gleichzeitig ist jedoch auch die Genauigkeit (precision) der Klassifikationsergebnisse abhängig von der Anzahl der Trainingsbeispiele, so daß sich das Problem nicht durch eine Reduktion der Trainingsdatenmenge lösen läßt.

Um den Feature-Raum klein zu halten, ohne dabei wichtige Informationen über die Trainingsdaten zu verlieren, werden Techniken zur Dimensionsreduzierung verwandt. Eine solche Technik ist die Singulärwertzerlegung, wie sie bei LSI zum Einsatz kommt. Durch die Reduzierung der Dimensionen in der Term-Dokument-Matrix gewinnt LSI an Performanz beim Klassifikationsvorgang. Der Performanzgewinn bezieht sich zum einen auf die Schnelligkeit der Klassifikation, da die Vergleiche bei der Klassifikation mit kürzeren Term-Vektoren durchgeführt werden können. Zum andern wird auch die Klassifikationsleistung verbessert (Recall und Precision), da überflüssige Informationen aus dem Inhalt der Dokumente quasi herausgefiltert werden[1].

LSI bietet eine gute Lösung für das Synonym- bzw. Homonym-Problem innerhalb von Dokumenten. Außerdem klassifiziert LSI, wie bereits erwähnt, Dokumente ausgehend von latenten semantischen Konzepten, die im Inhalt der Dokumente vorhanden sind.

[1] Für eine ausführliche Dokumentation über die Performance von LSI in Bezug auf Precision und Recall siehe (Deerwester et al., 1990).

Dadurch müssen nicht die „richtigen“ Keywords eingeben werden, um ein bestimmtes Dokument zu finden, sondern es kann sozusagen nach Themen gesucht werden, die über eine längere textuelle Beschreibung in das System eingeben werden können.
Im Rahmen dieser Arbeit wird Dokumenten-Retrieval, genauer gesagt Webdokumenten-Retrieval, nicht über Keywords sondern, über die Ähnlichkeit von Dokumenten betrieben. Es werden nicht einzelne Keywords sondern ganze Texte zur Suche nach passenden Dokumenten eingesetzt. LSI erscheint daher für diese Aufgabe vielversprechend.

4. Bestehende Ansätze und Systeme

In diesem Kapitel sollen einige Ansätze und darauf basierende Anwendungen zur Verbesserung der Informationssuche im World Wide Web vorgestellt werden.
Die Ansätze unterstützen den Benutzer in verschiedenen Bereichen. Zum einen im Finden und Filtern der im Web vorhandenen Informationen nach den Vorgaben oder Vorlieben des Benutzers und zum anderen beim Ordnen und Strukturieren der von Suchmaschinen gelieferten Ergebnisse.

4.1. Bestehende klassifizierende Systeme

In der Vergangenheit haben sich eine beträchtliche Menge von Forschungsarbeiten mit dem Klassifizieren von Zeitungsartikeln und ähnlichen Dokumenten befaßt. Die Klassifizierungstechniken reichen von Naive Bayes über Entscheidungsbäume bis zu Support-Vektor-Maschinen (siehe u.a.: Yang und Liu, 1999; Lewis et al., 1996; Joachims, 1997)
Relativ viele Arbeiten beschäftigen sich auch mit dem automatischen Klassifizieren von Online-Nachrichten. Dabei handelt es sich größtenteils um USENET Nachrichten oder Email-Nachrichten (siehe u.a.: Hsu und Lang, 1999; Boone, 1998)

4.1.1. Clusterbasierte Systeme

Ein weiteres Forschungsfeld bildet das Clustering von Webdokumenten. Hier werden meist die Ergebnisse von Suchmaschinen nach Themen geordnet dargestellt, um die Ergebnisse übersichtlicher zu machen.
Clustering-Verfahren werden auch für die automatische Erstellung von Web-Directories und Ontologien verwendet (siehe u.a. Frommholz, 2001).

Ein Beispiel für das Clustering von Suchmaschinenergebnissen ist die Suchmaschine *Clusty* (clusty.com) von *Vivísimo*. Die Ursprünge von Vivísimo liegen in einem Forschungsprojekt der Carnegie Mellon University in Pittsburgh. Gegründet wurde Vivísimo im Juli 2000, unter der Leitung von Raul Valdes-Perez.
Clusty ist eine Crawler-basierte Metasuchmaschine, die ihre Ergebnisse nach Themen geordnet in Clustern ausgibt. Diese Cluster können dann wie bei einem Web-Directory durch Browsen durchsucht werden. Dabei werden die Bezeichnungen der Cluster nicht von Clusty vorgegeben, wie dies bei normalen Web-Directories der Fall ist, sondern dynamisch aus den Suchergebnissen generiert.

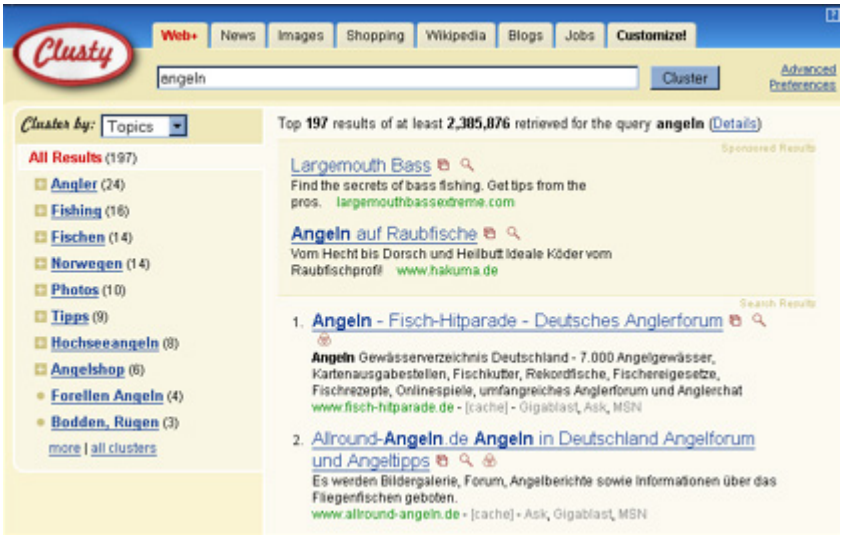

Abbildung 14: Suchergebnis der Suchmaschine Clusty

Ein weiteres Beispiel für ein solches System ist *Grokker*. Grokker ist ein Tool zur Websuche und Informationsvisualisierung. Die clientbasierte Software von Groxis Inc. (www.groxis.com) durchsucht das Web oder die Festplatte und sortiert die gefundenen Informationen in Clustern. Die Cluster werden in einer Map als farbige Kreise visualisiert. Unterkategorien werden innerhalb dieser wiederum als farbige Kreise oder Quadrate dargestellt.

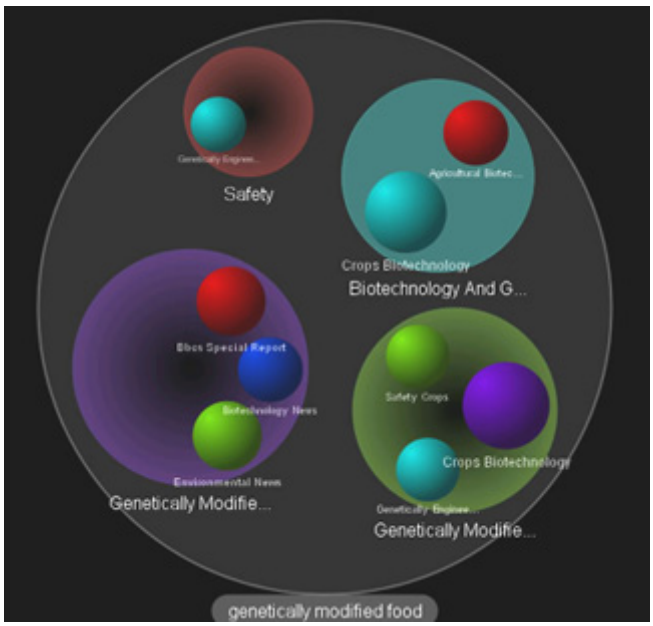

Abbildung 15: Suchergebnis der Suchmaschine Grokker

Der Vorteil der geclusterten Ergebnisse ist, daß Dokumente viel übersichtlicher dargestellt werden können, als über ein herkömmliches Rankingverfahren. So wird die Suche nach Dokumenten mit einem bestimmten Thema sehr vereinfacht.

Somit wäre es allgemein wünschenswert, Ergebnisse von Websuchen gleich in einer Struktur von Clustern darzustellen. Sind die Cluster zusätzlich permanent, so wie es bei einem Clientbasierten System möglich wäre, könnte zudem das Wiederauffinden von einmal gespeicherten Dokumenten erleichtert werden.

4.1.2. Systeme zur Klassifizierung von Webseiten

Relativ wenige Forschungsarbeiten beschäftigen sich bisher mit der Klassifizierung von Webseiten in eine vorgegebene Clusterstruktur. Zu diesen gehören beispielsweise die Arbeiten von Tsukada et al. (2001) und Mladenić (1998), bei denen es um den automatischen Aufbau und die Erweiterung von Web-Directories geht.

Systeme, die sich der Unterstützung der Suche im Web durch Klassifikation widmen, sind in den Arbeiten von Chekuri et al. (1997), Gauch et al. (2003), und Potamias et al. (2001) zu finden.

Chekuri et al. (1997) stellen ein System vor, mit dessen Hilfe die Treffgenauigkeit von Websuchergebnissen verbessert werden soll. Hier wird ein Klassifikator zum Filtern der Suchergebnisse eingesetzt.
Chekuri et al. (1997) entwickeln ein Such-Interface, welches im Grunde dem einer herkömmlichen Suchmaschine entspricht, also auf der Eingabe von Suchworten durch den Benutzer basiert, jedoch zusätzlich die Angabe eines oder mehrer Themen erlaubt. Durch die Angabe der Themen sollen die Suchergebnisse beschränkt werden, und zwar nur auf solche Dokumente, die den vorgegebenen Themen entsprechen.
Möglich wird dies durch einen Vektor-Raum basierten Klassifikator. Der Klassifikator wird mit einer Menge von im System vorgegebenen Kategorien und bereits vorklassifizierten Webseiten trainiert. Als Kategorien und Trainingsbeispiele können beispielsweise ein Web-Directory und die dort bereits klassifizierten Webseiten dienen.
Der Klassifikator ist anschließend in der Lage alle, über eine normale Keyword-basierte Suche gefundenen, Webseiten in eine der vorhandenen Kategorien einzuordnen. Diese Kategorie wird dann mit den vom Benutzer vorgegebenen Themen verglichen. Paßt das gefundene Dokument zu keinem der vom Benutzer vorgegebenen Themen, ist das Dokument nicht relevant und wird nicht in das Suchergebnis mit einbezogen.
Bei der Auswahl der Themen ist der Benutzer auf die im System vorhandenen Kategorien beschränkt.
Durch die Klassifizierung der über Keywords gefundenen Webseiten wird die Präzision der Suche erhöht, da nur Themenrelevante Dokumente im Suchergebnis dargestellt werden.

Während bei Chekuri et al. (1997) die Kategorien bereits im System vorgegeben sind, wird bei den Systemen von Gauch et al. (2003) und Potamais et al. (2001) ein individuelles Nutzerprofil aufgebaut, das es dem Nutzer erlaubt eigene Kategorien zu erstellen.

Gauch et al. (2003) stellen ein „Ontology-Based Personalized Search and Browsing“ (OBIWAN) System vor. Der Benutzer wird hier nicht nur bei der Suche im Web über eine Suchmaschine unterstützt, sondern auch direkt beim Browsen. Die Darstellung der Informationen basiert dabei auf einem persönlichen Nutzerprofil.
Das Profil des Benutzers wird aus seinem Browsingverhalten erstellt. Hierfür wird ein Klassifikator aus einer vorgegebenen Ontologie (hier wird das Web-Directory von Lycos als Ontologie verwendet) und den im Browser-Chache des Benutzers vorhandenen

Dokumenten generiert. Das Nutzerprofil besteht dann aus der, nach dem Surfverhalten des Benutzers gewichteten, Ontologie, die das Interesse des Nutzers widerspiegeln soll.
Suchmaschinenergebnisse werden über ein nach dem Nutzerprofil individuell erstelltes Ranking dargestellt. Ebenso wie bei Chekuri et al. (1997) werden irrelevante Dokumente anhand des Nutzerprofils herausgefiltert.

Das Browsing wird unterstützt, indem ganze Websites inhaltlich analysiert werden und über die vorgegebene Ontologie strukturiert werden.
Ähnlich wie bei Clusty werden die Inhalte der Webseiten einer Kategorie zugeordnet und als Hierarchie von Clustern dargestellt. Da jedoch kein Clustering stattfindet, sondern Klassifizierung eingesetzt wird, ist die Clusterhierarchie durch die Lycos-Ontologie vorgegeben.
OBIWAN bietet hierfür einen speziellen Browser, in dem, zusätzlich zu der jeweiligen Webseite, alle Kategorien der Ontologie angezeigt werden. Diese dienen so als permanent zur Verfügung gestellte Struktur und erleichtern die Suche nach bestimmten Themen.
Personalisierung findet über eine vom Nutzer selbst erstellte Ontologie statt. Hierfür werden vom Nutzer ausgewählte Kategorien über eine Abbildung mit der vorgegebenen Lycos-Ontologie verknüpft.
Für die Strukturierung der Website über die nutzerbasierte Ontologie wird ebenfalls ein Klassifikator eingesetzt, der mit vom Nutzer vorklassifizierten Dokumenten trainiert wird.

Das System von Potamias et al. (2001) bietet ebenfalls Unterstützung durch die Klassifizierung von Suchmaschinenergebnissen. Für die Klassifizierung wird ein Vektor-Raum-basierter Klassifikator verwendet.
Das Nutzerprofil wird aus vom Nutzer ausgewählten Kategorien gebildet. Die Kategorien sind dabei frei wählbar und werden über einzelne Schlüsselworte beschrieben. Die Schlüsselworte werden ebenfalls vom Nutzer ausgewählt.
Potamias et al. (2001) erwähnen zudem ein Tool, welches entwickelt wurde, um die Schlüsselworte automatisch aus vom Nutzer bereitgestellten Texten zu extrahieren. Über dieses Tool kann das System um eine automatische Nutzerprofilgenerierung erweitert werden.
Der Klassifikator wird mit vom Nutzer vorklassifizierten Trainingsdokumenten trainiert. Diese Trainingsdokumente werden jedoch zuvor an das Nutzerprofil angepaßt, indem die vom Nutzer gewählten Schlüsselworte für die jeweilige Kategorie extrahiert werden.
Eine Besonderheit des Systems ist die Ausnutzung der Struktur von HTML Dokumenten für die Klassifizierung. Zusätzlich zu den in der Trainingsphase aus den Dokumenten extrahierten Termen werden Terme aus bestimmten HTML-Tags extrahiert und speziell gewichtet.
Beispielsweise werden Terme, die im `<TITLE>`-Tag des Dokumentes vorkommen, stärker gewichtet als alle anderen Terme. Welche Tags hierbei Beachtung finden und die Gewichtung der Terme, wird vom Nutzer festgelegt und ist somit Teil des Nutzerprofils.

Auch bei diesen Systemen ließe sich ein weiterer Nutzen erzielen, wenn die Klassifikation der Suchergebnisse nicht verloren ginge sondern permanent gehalten werden könnte.

4.2. Bestehende Agentensysteme zur Informationssuche im WWW

Es existieren einige Systeme, die Informationsagenten einsetzen, um den Benutzer bei der Suche im WWW zu unterstützen. Diese Systeme nutzen häufig Techniken aus den Bereichen Data-Mining, Information Retrieval und maschinelles Lernen, um ein Nutzerprofil aufzubauen und Informationen aus dem Web zu filtern und dem Nutzer seinen Bedürfnissen nach zu präsentieren.

Eine Abgrenzung zu den im vorherigen Abschnitt vorgestellten Systemen ist jedoch nicht immer eindeutig gegeben, da der Begriff Agentensystem oft nur aufgrund des Einsatzes von maschinellem Lernen und einer Nutzerprofilbildung verwendet wird. Aufgrund dieser Eigenschaften könnte man auch einige der klassifizierenden Systeme aus dem vorherigen Abschnitt als Agentensysteme bezeichnen.

Einige Informationsagenten unterstützen den Benutzer direkt beim Browsing im Web. Beispiele hierfür sind WebWatcher (Armstrong et al., 1995), Letizia (Lieberman, 1995), WebMate (Chen und Sycara, 1998) und Syskill&Webert (Pazzani und Billsus, 1997).
Diese Systeme beobachten den Benutzer bei der Suche nach Dokumenten im WWW. WebWatcher verwendet Naive Bayes, um während des Browsens aus den besuchten Seiten ein Nutzerprofil zu erstellen. Der Benutzer muß hierbei die aufgerufenen Seiten auf ihre Relevanz hin bewerten. Mit Hilfe des Nutzerprofils ist WebWatcher dann in der Lage, dem Nutzer beim Browsen Links vorzuschlagen, die seinen Interessen entsprechen.

Letizia ist ebenfalls ein solcher Browsing-Assistent. Hier wird statt Naive Bayes das Vektorraummodell und TFIDF (term frequency/inverse document frequency)[1] verwendet, um ein Nutzerprofil zu erstellen.
WebMate und Syskill&Webert arbeiten auf ähnliche Weise und verwenden ebenfalls TFIDF bzw. Naive Bayes zur Erstellung des Nutzerprofils und zum Vorschlagen von relevanten Webinhalten.

Eine weitere Art von Informationsagenten basiert auf der Eingabe von Indextermen, um anschließend nach relevanten Informationen im WWW oder anderen verteilten Informationsquellen zu suchen. Zu diesen Systemen zählen Citeseer (Bollacker et al., 1998), FAQFinder (Burke et al., 1997), MySpiders (Pant und Menczer, 2002) und MELISSA (Brian und Garzon, 2001).
Während MySpiders Multiagentensysteme sind, deren Suchmechanismen darauf ausgerichtet sind, alle möglichen Arten von HTML-Dokumenten im WWW zu finden, ist Citeseer speziell auf die Suche von wissenschaftlichen Dokumenten, vor allem auch im PostScript und PDF-Format, zugeschnitten. FAQFinder ist dagegen auf das Finden von Antworten auf vom Benutzer gestellte Fragen spezialisiert. Hierfür werden Datenbanken mit FAQ-Files durchsucht.

Die Multiagentensysteme MySpiders und MELISSA sollen hier näher vorgestellt werden, da die verwendeten Suchmechanismen grundlegend für das in dieser Arbeit entwickelte System sind.

[1] TFIDF ist ein Verfahren, um die Relevanz eines Terms in einem Dokument zu bestimmen. Das Verfahren stützt sich auf das Vektorraummodell und die Berechnung der Termfrequenzen in Dokumenten. Ähnlichkeiten von Dokumenten können über das Cosinus-Ähnlichkeitsmaß bestimmt werden. (zu Einzelheiten siehe Salton und McGill, 1983)

4.2.1. Intelligente Suchagenten - MySpiders

MySpiders (Pant und Menczer, 2002) ist ein Multiagentensystem, basierend auf einem evolutionären Algorithmus, der eine Population von Web-Crawler-Agenten kontrolliert. Die Agenten durchsuchen autonom das WWW, ausgehend von einer Suchanfrage des Benutzers. Dabei untersuchen sie eine gefundene Webseite auf vorhandene Links und verfolgen diese weiter, sofern sie dem Suchauftrag des Benutzers entsprechen. Das Ergebnis ist eine nach Relevanz geordnete Liste von Links, die auf die gefundenen Dokumente verweisen.
Jeder Agent des MAS durchsucht die Links in seiner Nachbarschaft nach neuen Dokumenten. Dabei legt der Agent intelligentes Verhalten an den Tag, indem er einordnen kann, ob ein Dokument zu der Benutzeranfrage paßt. Außerdem ist es dem Agenten möglich, selbst autonom zu entscheiden, welche Links er verfolgen will, um relevante Dokumente zu finden.

MySpiders verwendet einen *evolutionären Algorithmus*, basierend auf *Local Selection.* Diese Begriffe werden in den folgenden Abschnitten erläutert.

Evolutionäre Algorithmen

Evolutionäre Algorithmen nehmen sich die darwinistische Evolutionstheorie zum Vorbild. Diese arbeitet nach dem Prinzip der *natürlichen Auslese* oder auch *Survival of the Fittest*.
Für einen evolutionären Algorithmus wird eine Anfangspopulation von einzelnen Individuen erstellt, die jeweils mit ihrer eigenen Strategie ausgestattet sind, um eine ihnen gestellte Aufgabe zu lösen. Diese Population durchläuft dann zyklisch die folgenden Schritte:

1. Die einzelnen Individuen führen jeweils die notwendigen Aktionen ihrer Strategie zum Lösen der Aufgabe durch. Danach wird ausgewertet, wie gut die Individuen jeweils beim Lösen der Aufgabe abgeschnitten haben. Das Ergebnis der Auswertung wird als *Fitneß* des Individuums bezeichnet.
2. Die besten Individuen werden zur Reproduktion ausgewählt. Dies bezeichnet man auch als *Selektion*.
3. Die Reproduktion findet statt, indem Kopien der ausgewählten Individuen erstellt und diese anschließend modifiziert werden. Die Modifikation der Nachkommen nennt man *Mutation*.
4. Im letzen Schritt wird aus den neu erschaffenen Individuen eine neue Generation erstellt und der Population hinzugefügt.

Der Algorithmus terminiert unter einer bestimmten, vorher festgelegten Bedingung, z.B. wenn die Population zu homogen wird, so daß kein evolutionärer Fortschritt mehr stattfinden kann. Dies bezeichnet man auch als *Konvergenz*.
Für nähere Informationen zu Evolutionären Algorithmen siehe u.a. (Goldberg, 1989).

Evolutionäre Agenten

Alle Agenten in MySpiders werden mit einem Attribut „Energie“ ausgestattet. Die Energie verändert sich im Laufe des Lebens eines Agenten je nach Relevanz der gefundenen Dokumente. Ein Agent, der ein relevantes Dokument gefunden hat, wird mit einer Energiezufuhr belohnt. Dabei wird nur der Agent belohnt, der ein Dokument zum ersten Mal besucht. Weitere Besuche des selben Dokumentes lohnen sich also nicht. Dadurch werden Deadlocks, die durch zyklisches Aufrufen von Dokumenten entstehen können, vermieden. Außerdem wird die Breite der Suche so erhöht. Um dies realisieren zu können werden alle gefundenen Dokumente in einem Cache gespeichert. Die einzige Interaktion zwischen den Agenten geschieht durch das Teilen dieser Ressource. Darüber hinaus findet keinerlei Kommunikation oder Interaktion zwischen den Agenten statt.

Wird bei einem Agenten ein bestimmter Energie-Level θ erreicht, findet eine Reproduktion des Agenten statt. Ebenso sterben Agenten, wenn der Energie-Level unter ein Minimum ω fällt. Die Nachkommen eines Agenten merken sich die Merkmale des Dokumentes, welches zu einer Reproduktion geführt hat, und bauen in ihre Suchanfrage Worte ein, die häufig in diesem Dokument vorkamen. Hierdurch entsteht eine gewisse Mutation der Agenten.

Die original Suchanfrage wird wie gesagt vom User vorgegeben. Die MySpiders-Agenten bekommen ihre Start-URLs über die Ergebnisse einer normalen Suchmaschine, die mit der User-Anfrage gefüttert wurde. Jeder Agent analysiert das Dokument seiner Start-URL und die Links, die sich in dem Dokument befinden. Um die Links zu analysieren betrachtet der Agent die Worte, die vor und nach dem Link und in der Linkbeschreibung auftauchen.
Durch ein neuronales Netz, in dem die Häufigkeit der Worte um den Link herum gespeichert wird, wird für alle Links in dem Dokument ein Relevanz-Ranking erstellt. Danach wird der nächste zu verfolgende Link von dem Agenten ausgewählt.
Nachdem ein Dokument besucht wurde, wird dessen Inhalt mit der Benutzeranfrage verglichen und das neuronale Netz mit den Ergebnissen aktualisiert. Der Agent speichert außerdem seine aktuelle Position und einen Back-Link, um Sackgassen zu umgehen.
Eine schematische Darstellung eines MySpider-Agenten ist in Abbildung16 zu sehen.

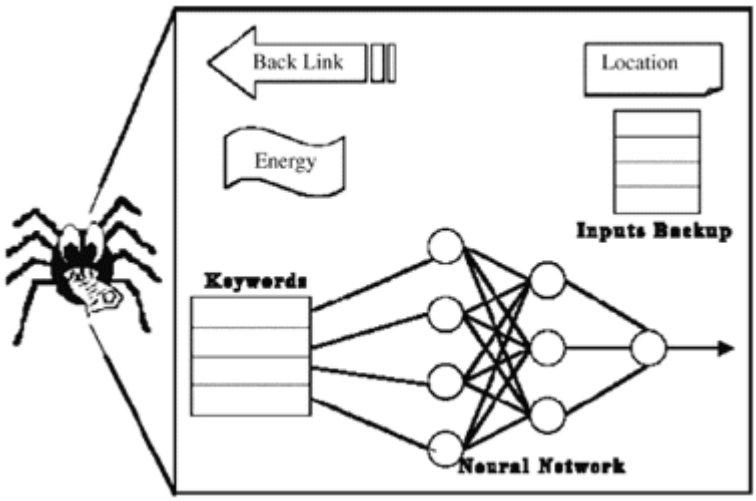

Abbildung 16: Darstellung eines MySpiders-Agenten (Pant und Menczer, 2002)

Local Selection

In Evolutionären Algorithmen, die globale Selektionsmechanismen verwenden, werden Individuen für die Reproduktion ihrer Fitneß nach ausgewählt. Dafür werden die Fitneßwerte aller Agenten untereinander verglichen. Die Werte, für die Energie-Level θ und ω, bei denen Reproduktion oder Tod eintritt, würden bei einem solchen Algorithmus von den Energiewerten der anderen Agenten abhängen.
Bei einem Local Selection Algorithmus sind die Werte θ und ω eines Agenten unabhängig vom Rest der Population. Dafür hängt der Energie-Bonus, den ein Agent als Belohnung bekommt, von den Ressourcen der Umwelt ab. Diese Ressourcen teilen sich alle Agenten, so daß hierdurch eine gewisse indirekte Interaktion zwischen den Agenten stattfindet. Wie bereits erwähnt, ist dies die einzige Interaktion, die zwischen den MySpiders Agenten stattfindet.

Der Vorteil eines evolutionären Algorithmus, der auf der Local Selection Methode basiert, liegt darin, daß die Agenten nicht ständig unter einander verglichen werden müssen. Es ist keine zentrale Instanz erforderlich, die Daten über die Energie, bzw. Fitneß aller Agenten sammelt. Die Agenten müssen auch untereinander keine Informationen über ihre Fitneß austauschen. Dadurch wird die Interaktion zwischen den Agenten auf ein Minimum reduziert. Dies ist ein wichtiges Kriterium für verteilte Anwendungen, wie Multiagenten Systeme. Es bedeutet, daß die Agenten weitgehend unabhängig voneinander, parallel arbeiten und sich asynchron vermehren können. Dadurch werden Flaschenhälse vermieden.

Durch das Teilen der Ressourcen wird Nischenbildung gefördert. Dadurch bleibt die Diversität der Population erhalten. Das Prinzip der Nischenbildung bedeutet eine Abwertung von Individuen, die in einer dichtbesiedelten Region existieren. D.h. die Fitneß der Agenten sinkt, wenn ihr Gebiet zu dicht besiedelt wird.
Das hat den Effekt, daß nicht alle Agenten immer wieder die gleichen Dokumente entdecken, sondern nach neuen Dokumenten an anderen Orten suchen, wenn Dokumente bereits von andern Agenten gefunden wurde.

In Abbildung 17 wird die Idee eines Local Selection Algorithmus als Pseudocode dargestellt.

```
initialize population of agents, each with energy E_0
while there are alive agents
   for each agent i
      1. input: sense environment
      2. output: compute action a
      3. update energy:
         E_envt ← E_envt − benefit(a)
         E_i ← E_i + benefit(a) − cost(a)
      4. optional learning
      5. selection:
         if (E_i > θ)
            reproduce(i)
            E_offspring ← E_i / 2
            E_i ← E_i / 2
         else if (E_i ≤ ω)
            die(i)
         end
   end
   E_envt ← E_envt + E_replenish
end
```

Abbildung 17: Pseudocode eines Local Selection basierten Algorithmus (Pant und Menczer, 2002)

In der MySpiders Implementation werden für die Energie-Level zur Selektion die folgenden konstanten Werte verwendet:

$\theta = 2E_0 > 0$, wobei E_0 der Anfangsenergiewert aller Agenten ist.
$\omega = 0$

Energie wird durch die Kosten einer Aktion verbraucht und über die Umwelt erzeugt. Bei der Reproduktion wird Energie an die Nachkommen weitergegeben. Sobald in einem Agenten keine Energie mehr vorhanden ist, stirbt er.

MySpiders Architektur

Das MySpiders System besteht aus fünf Subsystemen:

- das User-Interface
- der Manager
- gemeinsam genutzte Objekte, wie z.B. der Chache
- Utilities für das Information Retrieval (parsing, stemming, etc.)
- die Spider-Agenten

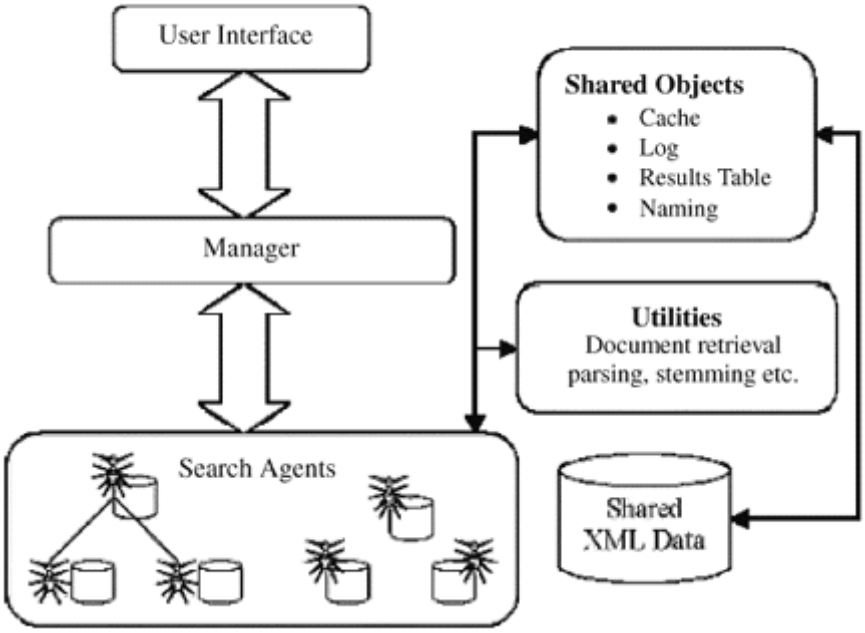

Abbildung 18: MySpiders Architektur (Pant und Menczer, 2002)

Um eine Trennung von User-Interface und Agenten zu erreichen, wird eine 3-Schichten-Architektur verwendet. Dafür wird der Manager als Middle-Tier zwischen Agenten und User-Interface eingesetzt. Dies ermöglicht die einfache Konfiguration und Erweiterung der Anwendung. Der Manager bietet eine Schnittstelle, die es dem User-Interface erlaubt, die Agenten zu starten und zu stoppen, einzelne Parameter der Suche zu verändern und den Suchstatus anzufordern und abzubilden.

MySpiders ist eine Java-Anwendung und als Java-Applet realisiert. Somit kann es direkt im Browser des Benutzers gestartet werden.

4.2.2. Websuche mit LSI - MELISSA

MELLISA (Moblie Electronic LSA[1] Internet Server Search Agent) ist ein Multiagentensystem, bestehend aus mobilen Informationsagenten, die LSI für die Suche im Web einsetzen. Es wurde von Hal Brian und Max Garzon an der University of Memphis entwickelt (Brian und Garzon, 2001).

MELISSA begegnet den genannten Problemen bei der Websuche mit mobilen Agenten. Mobile Agenten sind Programme, die auf entfernte Hostrechner migrieren können, um dort die Aufträge ihre Benutzer zu erledigen. Für die Implementation von MELISSA wurde das IBM Aglet Software Development Kit (ASDK) verwendet. IBM Aglets sind mobile Agenten, die eigenständig, d.h. in einem eigenen Thread, auf Aglet-fähigen Hostrechnern arbeiten. Auf dem eigenen wie auch auf fremden Hosts besteht die Möglichkeit der Interaktion mit fremden Diensten und Datenbanken.

[1] LSA (Latent Semantic Analysis) ist gleichbedeutend mit LSI

MELISSA besitzt eine Master-Slave Architektur, bestehend aus einem Master-Agenten, der die Schnittstelle zum Benutzer beinhaltet, und mehreren Slaves, die auf verschiedene Hosts gesendet werden. Der Benutzer gibt eine Anfrage in Form von Suchworten in die Benutzerschnittstelle ein. Der Master-Agent gibt diese Anfrage an die Slave-Agenten weiter und schickt sie zu verschiedenen Web-Servern. Nach der Migration auf den fremden Host durchsuchen die Slave-Agenten den Web-Server ihres Hosts nach Webseiten, die der Anfrage des Benutzers entsprechen. Jeder Slave-Agent erstellt eine Liste der gefundenen relevanten Webseiten und schickt diese zurück an den Master-Agenten.

Für die Suche kommt in jedem Slave-Agenten ein LSI-Modul zum Einsatz. Die Benutzeranfrage wird per LSI mit den Web-Dokumenten auf dem jeweiligen Server verglichen. Hierfür wird die Benutzeranfrage als Dokument betrachtet und in eine m x 1 Matrix umgewandelt. m ist hierbei die Anzahl der verschiedenen Suchworte der Anfrage. Aus den Webdokumenten, die der Agent auf dem jeweiligen Server findet wird eine Term-Dokument-Matrix erstellt. Die Singulärwertzerlegung und Dimensionsreduzierung erfolgt wie in Kapitel 3.2.4 beschrieben. Die neue Dimension wird in den MELISSA Agenten auf k=2 festgelegt. Die Benutzeranfrage wird dann mit dem vom Agenten erstellten Index verglichen.

Brian und Garzon (2001) geben LSI als eine sehr effektive Methode für die Suche im WWW an. Es habe sich gezeigt, daß LSI der menschlichen Beurteilung über die Bedeutung und Ähnlichkeit von Worten in natürlichsprachlichen Texten sehr nahe kommt.
Als Quelle hierfür geben Brian und Garzon ein Experiment mit LSI und einem TOEFL-Test (Test of English as Foreign Language) an. Der TOEFL-Test ist ein standardisierter Test, der die grundlegenden englischen Sprachkenntnisse eines Kandidaten prüft. Hier schnitt LSI bei der Beurteilung von englischen Synonymen ebenso gut ab wie ein durchschnittlicher menschlicher Kandidat (Landauer und Dumais, 1997).
In einem anderen Experiment ging es um die Fähigkeit der LSI-Methode, Wissen aus Texten zu extrahieren. Für ein Multiple-Choice-Examen wurde LSI mit einem einführenden Psychologiebuch trainiert. LSI schnitt bei dem Test so gut ab, daß es das Examen bestanden hätte (Landauer und Dumais, 1997).

Aus diesen Versuchen folgern Brian und Garzon, daß es mit LSI möglich ist, die menschliche Intelligenz zu simulieren, die nötig ist, um den Inhalt von Webseiten zu erkennen und auf Relevanz zu prüfen.

Um diese These zu untermauern, wurde das MELISSA-System auf seine Performanz getestet und mit einer aktuellen Suchmaschine verglichen. Für den Vergleich wurde Alta Vista ausgewählt, da sie zur Zeit des Experimentes als eine der besten Suchmaschinen in Bezug auf Schnelligkeit sowie Größe und Aktualität ihres Indexes galt.

Bei dem Vergleich wurden zum einen die Antwortzeiten der beiden Systeme sowie zum anderen die Treffgenauigkeit bei den Ergebnissen bewertet.
In Bezug auf die Antwortzeiten schnitt MELISSA erwartungsgemäß schlechter ab als Alta Vista. Während Alta Vista nur 0,5 Sekunden für die Ausgabe der Ergebnisse brauchte, lag der Wert bei MELISSA, mit mehr als 10 Sekunden durchschnittlicher Antwortszeit, deutlich höher. Dies liegt daran, daß herkömmliche Suchmaschinen durch ihren schon vorhandenen Index einen zeitlichen Vorsprung gegenüber realtime-basierten Systemen haben.

Ein deutlicher Vorteil von MELISSA gegenüber Alta Vista ergab sich jedoch bei dem Vergleich der Suchergebnisse in Bezug auf die Treffgenauigkeit. Hierfür wurden die Ergebnisse beider Systeme von Hand auf Relevanz geprüft. Es wurde das Ranking der Ergebnis-Dokumente sowie auch die Anzahl der wirklich relevanten Dokumente im Vergleich zu falsch identifizierten Dokumenten verglichen. Beim Ranking schnitt MELISSA im Durchschnitt um 35% besser ab als Alta Vista. Bei der Prüfung der Ergebnisse auf Relevanz erzielte MELISSA sogar einen um 147% besseren Wert als Alta Vista.

4.3. Fazit

Die clusterbasierten Systeme zeigen, daß die Darstellung von Suchergebnissen in Clustern sehr viel übersichtlicher ist, als die Ausgabe einer langen Ergebnisliste.
Der Vorgang des Klassifizierens von Suchergebnissen in Cluster ist einem Filtern nach Themen gleichzusetzen, wie die Arbeit von Chekuri et al. (1997) zeigt. Das Finden von Dokumenten zu einem bestimmten Thema kann durch diesen Filtervorgang erheblich erleichtert werden.

Agenten können einen Benutzer in dieser Hinsicht unterstützen, indem sie Dokumente nach Themen klassifizieren, die den Interessen des Benutzers entsprechen. Hierfür ist es sinnvoll ein Nutzerprofil zu erstellen.
Einige Agentensysteme versuchen, ein Nutzerprofil automatisch, durch die Beobachtung des Benutzerverhaltens, zu generieren. Hierbei kommen jedoch leicht Fehleinschätzungen zu stande, die zu Aktionen des Agenten führen können, die der Nutzer nicht nachvollziehen kann.
Daher ist es nötig, die Erstellung des Nutzerprofils so transparent wie möglich zu gestalten. Dies ist möglich, wenn beispielsweise nicht Dokumente aus dem Browsercache zur Profilgenierung verwendet werden, sondern der Nutzer die Dokumente selbst auswählen kann.

Klassifikation findet meist über Vektorraum- oder Naive-Bayes-basierte Klassifikatoren statt. Für die Konstruktion sind meist alleine Worthäufigkeiten in den Dokumenten ausschlaggebend. Bei diesem sogenannten Keyword-Matching werden die semantischen Zusammenhänge von Worten in Texten außer Acht gelassen. Verfahren wie LSI, die semantische Korrelationen mit einbeziehen, können hier Vorteile bringen.

Beim Vergleich mit der Suchmaschine Alta Vista zeigte das LSI-basierte System MELISSA klare Vorteile. Das besonders gute Abschneiden von MELISSA bei der Relevanz der Suchergebnisse demonstriert das Leistungsvermögen des LSI-Verfahrens bei der Suche von Texten.
Im Ergebnis von MELISSA wurden sehr viele irrelevante Webseiten, die Alta Vista als Ergebnis geliefert hatte, herausgefiltert. Brian und Garzon sehen hier einen klaren Vorteil von LSI gegenüber Verfahren, die über Keyword-Matching arbeiten.

Zur Verbesserung der Klassifizierung können auch andere Merkmale von Dokumenten als der Inhalt verwendet werden. Hier bieten sich bei HTML-Dokumenten vor allem HTML-Tags an, wie bei dem System von Potamias et al. (2001) umgesetzt. MySpiders verwendet zudem Linkstrukturen, um Voraussagen über Dokumente zu treffen.

Für die Klassifizierung von wissenschaftlichen Publikationen bieten sich außerdem auch Strukturen innerhalb der Texten an, wie Titel, Abstracts und Zitate. Zitate können hier wie Links verwendet werden um die Publikationen zu ermitteln, die sich aufeinander beziehen (siehe u.a. Salton, 1971). Eingesetzt wird dies beispielsweise in der Literaturdatenbank Citeseer (Giles, 1998).

Durch den Einsatz eines Multiagentensystems zur Websuche können ebenfalls Vorteile im Vergleich zur Suche mit normalen Suchmaschinen gewonnen werden. Herkömmliche Suchmaschinen versuchen, das WWW durchsuchbar zu machen, indem sie das ständig wachsende und sich verändernde Web nach neuen Inhalten durchforsten und diese in einem Index verfügbar machen. Dieses Vorgehen führt aufgrund der genannten Eigenschaften des Webs zu einem Aktualitäts- und Skallierbarkeitsproblem. Der Anspruch, durch den Index quasi ein aktuelles Abbild des WWW zu erstellen, kann daher kaum zum Erfolg führen.
Hier setzt die Idee von MySpiders und MELISSA an. Diese Systeme kombinieren Index-basierte Suchmaschinen mit lokalen intelligenten Suchagenten. Diese können ihre Suche in Echtzeit durchführen und verhindern so das Auftauchen von Broken Links im Suchergebnis.
Zudem werden Dokumente gefunden, die von einer Index-basierten Suchmaschine nicht gefunden oder indexiert wurden.

Der Nachteil von Echtzeitbasierten Systemen ist der erhöhte Zeitbedarf für die Suche. Die verbesserten Suchergebnisse können jedoch nach Meinung von Brian und Garzon (2001) die schlechteren Antwortzeiten kompensieren.

Mobile Agenten bringen bei der Realtime-Suche einen klaren Zeitvorteil gegenüber anderen echtzeitbasierten Architekturen. Die Kommunikationskosten für die Anfrage beim Host und das Empfangen jedes einzelnen Web-Dokumentes entfällt. Der Aufwand der Suche läßt sich so auf den eigentlichen Suchvorgang, also das Vergleichen der Benutzeranfrage mit den Dokumenten, beschränken. Dies bewirkt eine Beschleunigung des gesamten Vorgangs. Zusätzlich entsteht nur minimaler Netzwerkverkehr, so daß mobile Such-Agenten die Netzlast im WWW nicht übermäßig vergrößern.
Der Nachteil ist hierbei jedoch, daß alle Hosts die Ausführung der mobilen Agenten zulassen müssen. Eventuell muß hierfür eine spezielle Software auf dem Host installiert sein.

5. Umsetzung eines beispielhaften Prototypen

Ziel dieser Arbeit ist der Entwurf und die beispielhafte Umsetzung eines Softwaresystems, daß Informationssuchende bei der Recherche nach wissenschaftlichen Publikationen im WWW unterstützt.
In diesem Kapitel sollen die Konzeption und die Implementation des Systems *DocAssistant*, das im Rahmen dieser Arbeit entwickelt wurde, beschrieben werden.

5.1. DocAssistant

DocAssistant ist ein Multiagentensystem, das Informationsagenten im Auftrag des Benutzers einsetzt, um wissenschaftliche Publikationen im WWW zu finden und diese ihrem Thema nach zu klassifizieren. DocAssistant verfügt über ein Nutzerprofil, um individuell auf den Benutzer zugeschnittene Unterstützung bei der Recherche zu bieten.
Für die Suche nach Publikationen im Web wird ein evolutionäres Multiagentensystem nach dem Vorbild von MySpiders (siehe Kap. 4.2.1.) eingesetzt. DocAssistant verwendet LSI für die Relevanzanalyse, wie es in Kapitel 4.2.2 anhand des Systems MELISSA beschrieben wurde. Zusätzlich wird LSI verwendet, um die gefundenen Dokumente in einer Clusterstruktur zu klassifizieren und zu archivieren.

5.2. Konzeption

Um wissenschaftliche Informationssuchende bei der Dokumentenrecherche im WWW zu unterstützen, wurde in Kapitel 2.3. das Vorgehen bei einer solchen Recherche analysiert. Es wurden zwei Bereiche des gesamten Vorganges ausgewählt, die mit Hilfe von maschinellem Lernen und dem Einsatz von Agenten unterstützt werden können.
Der erste Bereich ist das Ausfindigmachen von relevanten Publikationen durch die manuelle Suche (browsen) oder den Einsatz von Suchmaschinen. Unterstützung soll hier durch die Automatisierung des Suchprozesses gegeben werden. Dies geschieht in DocAssistant durch den Einsatz eines Multiagentensystems.
Der zweite Aspekt, der unterstützt werden soll, ist die Archivierung von Dokumenten. Hier wird die Klassifikationstechnik LSI verwendet, um die gefundenen Dokumente ihrem Inhalt nach zu klassifizieren und in einer Clusterstruktur abzulegen.

5.2.1. Automatisierung des Suchprozesses

DocAssistant verwendet Informationsagenten in Form eines Multiagentensystems, um den Suchprozeß zu automatisieren. In diesem Abschnitt wird aufgezeigt, warum Agenten für diese Aufgabe besonders gut geeignet sind und auf welche Weise sie in DocAssistant eingesetzt werden.

Einsatz von Agenten für die Wissensgewinnung im Web

Die Wissensgewinnung im Web unterscheidet sich, wie bereits gesagt, in einigen Punkten zur Wissensgewinnung in Datenbanken.

Data-Mining	Web-Mining
Datenobjekte mit homogenen Attributen	Dokumente können sehr unterschiedlich sein (verschiedene Formate, unterschiedliche Sprachen)
Viele unabhängige Daten	Dokumente sind durch Links vernetzt
Datenmenge wächst eher langsam	Web wächst ständig und ändert sich sehr schnell
Datenmenge begrenzt	Riesige Datenmenge nicht vergleichbar mit herkömmlichen Datenbanken

Um den Anforderungen, die die Eigenschaften und die Struktur des Webs an ein Softwaresystem stellen, gerecht zu werden, wurden für die Automatisierung des Suchprozesses in DocAssistant Agenten eingesetzt.

Agenten sind besonders gut für die Unterstützung des Suchprozesses geeignet, da sie sich an die Gegebenheiten des WWW und die Anforderungen des Benutzers anpassen können. Intelligente Agenten können selbständig das Format und den Inhalt von Dokumenten erkennen und ihre Vorgehensweisen daran anpassen. Zudem sind Agenten autonom. Sie durchwandern das Web selbständig und ohne das Zutun des Benutzers.
Multiagentensysteme bieten außerdem ein hohe Maß an Modularität und Skalierbarkeit. Durch die riesige Datenmenge, das schnelle Wachstum und die unvorhersehbaren Veränderungen im WWW, ist es nötig, spezielle möglichst modulare Komponenten einzusetzen, die jeweils darauf spezialisiert sind, einen bestimmten Aspekt eines Problems zu lösen. Multiagentensysteme bieten diese Eigenschaften und sind daher für die Anforderungen, die das WWW an ein Softwaresystem stellt, besonders geeignet.
Das WWW hat die Struktur eins riesigen Netzes. Die Modularität des Agentensystems bietet die Möglichkeit sich dynamisch an diese Struktur anzupassen. Die Agenten sind in der Lage, die Links, über die die Webseiten vernetzt sind, zu verfolgen und werden so bei der Suche der Struktur des Webs gerecht.

Unterstützung des Suchprozesses

Durch den Einsatz von Informationsagenten ist es möglich den gesamten Suchvorgang zu automatisieren, ohne daß der Benutzer in den Suchprozeß eingreifen muß. Hierfür ist es jedoch notwendig, zuvor die Bedürfnisse des Benutzers zu ermitteln. Dies geschieht durch die Erstellung eines Nutzerprofils.

Das Nutzerprofil besteht in DocAssistant aus bereits vorhandenen und vom Nutzer nach Themen geordneten PDF- oder HTML-Dokumenten. Die vom Nutzer auf diese Weise

vorklassifizierten Dokumente dienen als Trainingsbeispiele für den in DocAssistant integrierten LSI-basierten Klassifikator. Über diesen Klassifikator prüft DocAssistant bei der Suche jedes gefundene Webdokument auf Relevanz.

Wissenschaftliche Publikationen werden immer häufiger neben der Veröffentlichung in Fachzeitschriften auch frei zugänglich im World Wide Web veröffentlicht. Als Dateiformat hat sich hier das PDF-Format etabliert. Daher werden in DocAssistant neben HTML-Dokumenten auch Dokumente im PDF-Format verarbeitet. Für die Archivierung sind ausschließlich PDF-Dokumente vorgesehen. Es wäre jedoch auch denkbar weitere Formate wie das Microsoft-Word-Format oder das PostScript-Format zu verwenden.

Ausgehend von einer Startseite im Web sucht DocAssistant mit Hilfe von Informationsagenten autonom nach PDF-Dokumenten, die zu den im Nutzerprofil angelegten Themen passen. Hierfür wird eine Population von evolutionären Agenten nach dem Vorbild von MySpiders erzeugt. Die DocAssistant-Agenten verfolgen die Links einer vorgegebenen Startseite und gelangen so zu neuen Webdokumenten und neuen Links, die dann wiederum weiter verfolgt werden.
Wie bei MySpiders erhält jeder Agent eine Anfangs Energie, die sich im Laufe seines Lebens erhöhen oder verringern kann. Die Agenten bewerteten die Relevanz der jeweils besuchten Webseite durch einen Vergleich mit den Dokumenten des Nutzerprofils. Ist ein Dokument für eine der vom Nutzer vorgegebenen Kategorien relevant, so wird der Agenten mit einer Energiezufuhr belohnt. Das Aufsuchen von Dokumenten kostet dagegen Energie. Sinkt die Energie eines Agenten auf null, stirbt er. Anders als bei MySpiders vermehrt sich ein Agent nicht nur bei einem bestimmten Energie-Level sondern solange er überhaupt Energie zur Verfügung hat. Dieses Prinzip wird im Abschnitt 5.2.4. näher erläutert.

Auf diese Weise durchsuchen die Agenten das WWW Link für Link. Die Population wächst vor allem an Orten an denen sich mehrere inhaltlich relevante Webseiten befinden und verringert sich in Regionen mit wenig relevanten Inhalten.
Stößt ein Agent bei der Suche auf ein PDF-Dokument, so wird es ebenfalls auf Relevanz geprüft. Paßt das Dokument inhaltlich zu den Dokumenten einer bestimmten vom Nutzer angelegten Kategorie, wird es als neues Dokument in der entsprechenden Kategorie abgelegt.

5.2.2. Archivierung von Dokumenten

In DocAssistant wird Klassifizierung mit Hilfe von LSI eingesetzt, um Dokumente während der Suche auf Relevanz zu prüfen. Anders als bei den in Kapitel 4 vorgestellten Systemen wird Klassifizierung in DocAssistant jedoch auch verwendet, um die gefundenen Dokumente zu archivieren. Über den LSI-Klassifikator werden neue Dokumente in vom Nutzer angelegte Kategorien eingeordnet. Die Klassifikation der Dokumente bei der Suche wird auf diese Weise also permanent gemacht.

Die Untersuchung der Systeme in Kapitel 4 hat gezeigt, daß Cluster eine geeignete Struktur darstellen, um Dokumente zu ordnen und nach Themen zu sortieren. Die permanente Speicherung von Dokumenten in einer Clusterstruktur bietet daher eine gute Möglichkeit, Dokumente geordnet zu archivieren.
In den meisten Betriebssystemen werden für die Strukturierung und Archivierung von Dateien Verzeichnisse verwendet, die der Benutzer in Form von Ordnern anlegen und

verwalten kann. DocAssistant verwendet diese Ordnerstruktur des Betriebssystems, um ein Nutzerprofil anzulegen. Dies bietet sich an, da Funktionen wie das Hinzufügen, Löschen oder Verschieben von Dokumenten innerhalb der Ordner bereits durch das Betriebssystem gewährleistet sind. Dies vereinfacht die Implementation. Zudem erhält es dem Benutzer den gewohnten Umgang mit Dokumenten und ermöglicht eine einfache Bedienbarkeit.

Um ein Benutzerprofil zu erstellen, legt der Benutzer Themenfelder fest, nach denen DocAssistant suchen und Dokumente auswählen soll. Für jedes Thema wird ein Ordner im Dateisystem angelegt. Bereits vorhandene Publikationen werden in den entsprechenden Ordnern gespeichert. Dabei ist die Bezeichnung der Ordner nebensächlich und vom Benutzer frei wählbar. Ausschlaggebend sind die Inhalte der Ordner, da DocClassifier neue Dokumente anhand der in den Ordnern befindlichen Dokumente klassifiziert. DocAssistant analysiert den Inhalt der gefundenen Dokumente und speichert sie thematisch geordnet in der vom Nutzer erstellten Clusterstruktur. Für die inhaltliche Analyse der Dokumente wird der LSI-basierte Klassifikator verwendet.

Für die Speicherung der gefundenen Dokumente wird eine Kopie der im Nutzerprofil vorhandenen Kategorien angelegt. Um neue Dokumente von den bereits vorhandenen zu unterscheiden, werden neue Dokumente zunächst in dieser Kopie der Clusterstruktur gespeichert. Auf diese Weise läßt sich nach der Suche auch auf einen Blick erkennen, ob überhaupt neue Dokumente von DocAssistant gefunden wurden. Der Nutzer kann anschließend auswählen, welche Dokumente er behalten möchte. Diese kann er durch Kopieren oder Verschieben der Dateien einfach in das Nutzerprofil übernehmen. Der Klassifikator kann für jedes neue Dokument durch ein LSI-Folding-In (siehe Kapitel 3.2.4.) aktualisiert werden.

5.2.3. Wissensgewinnung im Web mit DocAssistant

In Kapitel 2.1. wurde deutlich, das die Informationssuche im WWW nicht nur aus dem Einsatz eines Suchdienstes, also in diesem Fall Suchagenten, besteht, sondern daß die Informationssuche analog zum Knowledge Discovery in Datenbanken als ein Prozeß der Wissensgewinnung, mit mehreren Teilschritten gesehen werden kann.
In diesem Abschnitt soll daher die Unterstützung der Dokumentenrecherche durch DocAssistant anhand des in Kapitel 2.1.5 vorgestellten Prozesses der Wissensgewinnung im WWW dargestellt werden.

Als erster Schritt des Prozesses wurde die **Untersuchung des Anwendungsbereiches** genannt. Dieser Schritt wurde bereits in Kapitel 2.2 durchgeführt. Es ergaben sich die eingangs erwähnten Bereiche des Recherchevorgangs, die durch DocAssistant unterstützt werden sollen.

Schritt zwei beinhaltet das **Web Information Retrieval.** Hier findet die Suche nach relevanten Dokumenten im WWW statt. Dieser Vorgang besteht in DocAssistant aus zwei Teilschritten:

- Die Angabe der Start-URL beziehungsweise die Auswahl der Start-Links für die DocAssistant-Agenten.
  Dies geschieht über das Suchergebnis einer Suchmaschine, die Vorauswahl einer

Startseite durch ein Suchsystem wie Citeseer oder die direkte Auswahl einer Startseite durch den Benutzer.
- Der Einsatz der DocAssistant-Agenten. Die Agenten verfolgen die Links, die sie in ihrer Umgebung finden, und spüren so neue Dokumente auf.

Im dritten Schritt findet das **Information Extraction und Pre-processing** statt. Zu diesem Schritt gehört die Extraktion der Feature-Werte aus den Dokumenten, um die Klassifizierung durchführen zu können. Außerdem wird die Anzahl der Features durch den Einsatz von Stopwords reduziert. Die extrahierten Feature-Werte werden dann in Form einer Term-Dokument-Matrix gespeichert.

In Schritt vier findet dann das **Web-Mining** durch die Anwendung des Web-Mining-Modells auf die Daten statt. In diesem Fall wird ein Klassifizierungsmodell auf die von den DocAssistant-Agenten gefunden Webdokumente angewandt, um ihre Relevanz für den Benutzer zu bestimmen. Das Modell wird über das LSI-Verfahren durch das Training mit den vom Benutzer vorklassifizierten Dokumenten generiert. Durch den so entstandenen LSI-Klassifikator ist jeder DocAssistant-Agent damit in der Lage zu entscheiden, ob ein Dokument für den Suchenden relevant ist und in welche Kategorie es einzuordnen ist.

Im letzten Schritt erfolgt die **Interpretation und Evaluation** der gewonnenen Daten. Diese werden dem Benutzer in einer leicht verständlichen und faßbaren Form präsentiert. Das bedeutet, daß die aus dem Klassifizierungsvorgang als relevant eingestuften Dokumente in die entsprechenden Kategorien aus dem Nutzerprofil eingeordnet und gespeichert werden. Anschließend ist eine Anpassung des Klassifikators an die neuen Dokumente möglich.

5.2.4. DocAssistant Architektur

DocAssistant ist ein clientbasiertes Multiagentensystem das im wesentlichen aus drei Komponenten besteht. Die beiden Hauptkomponenten sind das Multiagentensystem, bestehend aus vielen einzelnen Informationsagenten, und das LSI-Modul. Eine dritte Komponente bildet das Nutzerprofil, welches der Nutzer über eine Clusterstruktur anlegt und verwaltet.

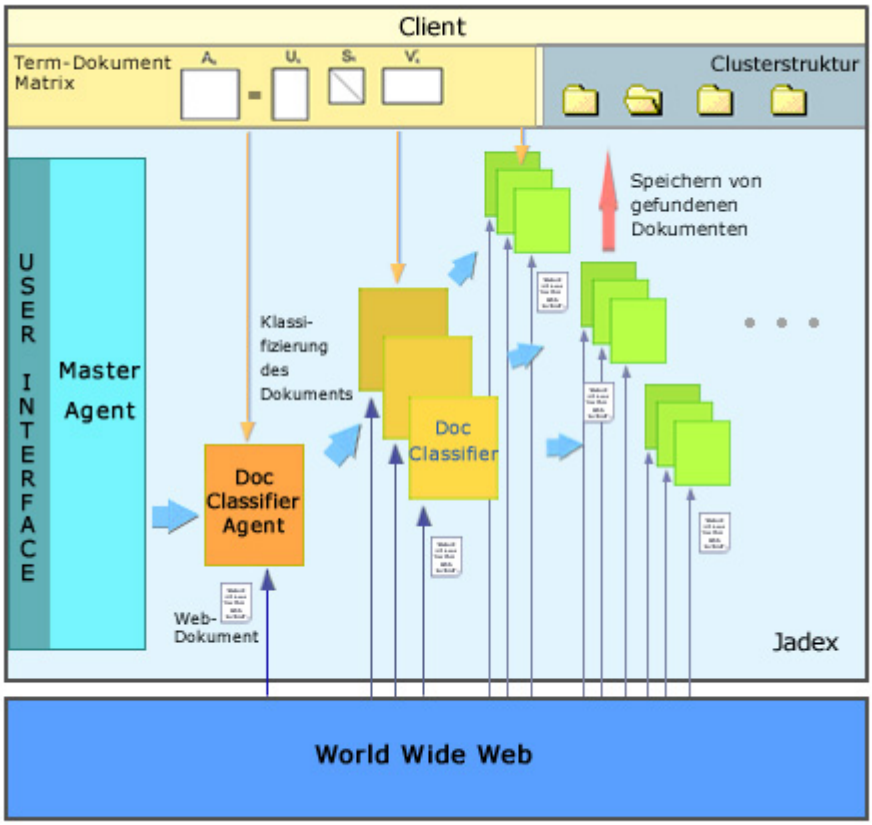

Abbildung 19: DocAssistant Architektur

Das Multiagentensystem

Das in DocAssistant eingesetzte Multiagentensystem orientiert sich an der Architektur des Systems MySpiders von Pant und Menczer (2002), welches in Kapitel 4.2.1. vorgestellt wurde.
Das Multiagentensystem besteht aus einem Manager-Agenten und mehreren Informationsagenten, den DocClassifier-Agenten.

Der **Manager-Agent** beinhaltet das Benutzerinterface und eine grafische Oberfläche, über die die DocClassifier-Agenten dargestellt werden können.
Über das Benutzerinterface können wesentliche Merkmale der Anwendung festgelegt werden, wie die Start-URL und die maximal erlaubte Anzahl von Agenten im System.

Bei Eingabe einer Start-URL über das Benutzerinterface startet der Manager-Agent einen **DocClassifier-Agenten** mit dem Auftrag, die geforderte Webseite aufzurufen, auf Relevanz zu Prüfen und die Links der Seite zu extrahieren.
Der DocClassifier-Agent startet nun wiederum für jeden gefundenen Link einen neuen DocClassifier-Agenten und gibt diesen den jeweiligen Link als Start-URL mit.
Dieser Vorgang wiederholt sich nun für jeden weiteren DocClassifier-Agenten. Die Lebensdauer und die Reproduktion der Agenten sind in DocAssistant etwas anders umgesetzt als in MySpiders. Der Lebenszyklus der DocClassifier-Agenten wird in dem entsprechenden Abschnitt genauer behandelt

Jeder DocClassifier arbeitet autonom und ist nur von seinem jeweiligen Erzeuger abhängig. Der Master-Agent ist weder zentrale Instanz des Systems noch greift er aktiv in die Vorgänge der DocClassifier-Agenten ein.

Der Manager-Agent

Die Aufgabe des Manager-Agenten ist es eine grafische Benutzeroberfläche bereitzustellen, über die die DocClassifier dargestellt werden können. Die DocClassifier agieren in dieser Konstellation als Clients, die an den Manager die Anforderung stellen, sie zu einem bestimmten Zeitpunkt darzustellen oder wieder zu löschen. Die DocClassifier werden in einer Hierarchie von Eltern-Kind-Beziehungen dargestellt (siehe Abbildung20).

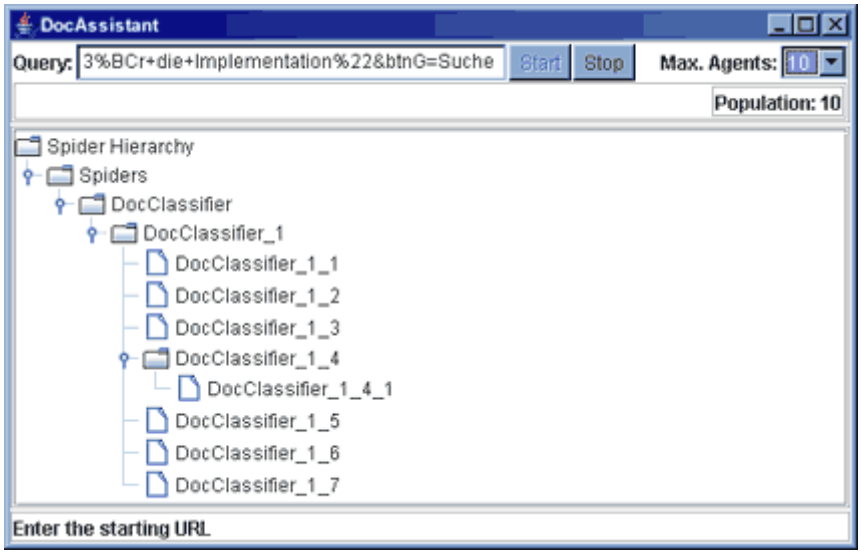

Abbildung 20: DocAssistant Benutzerinterface

Zudem kontrolliert der Manager wichtige Rahmenbedingungen für die Agentenpopulation. Das wichtigste Merkmal, welches der Manager vorgibt, ist die Start-URL, die über das Benutzerinterface durch den Benutzer festgelegt wird.
Bei der Wahl dieser URL ist der Benutzer völlig frei. Sinnvolle Adressen wären jedoch Webseiten in deren Umgebung wissenschaftliche Publikationen vermutet werden. Wie in Kapitel 2.2.3. angegeben, können dies Webseiten von Hochschulen, wissenschaftlichen Instituten, Homepages von bereits bekannten Autoren, Seiten wissenschaftlicher Tagungen oder Ergebnisseiten von wissenschaftlich orientierten Suchmaschinen sein. Auf wissenschaftliche Literatur spezialisierte Suchmaschinen und Literaturdatenbanken sind unter anderem, Scirus (www.scirus.com), Google Scholar (scholar.google.com), die Elektronische Zeitschriftenbibliothek Regensburg (rzblx1.uni-regensburg.de/ezeit/) und Citeseer (citeseer.ist.psu.edu).
Außerdem kann über das Benutzerinterface die maximale Anzahl der DocClassifier, die während der Suche erzeugt werden sollen, festgelegt werden.

Lebenszyklus der DocClassifier-Agenten

Um den Rahmen dieser Diplomarbeit nicht zu sprengen und den Implementationsaufwand zu beschränken, wurden einige Änderungen in Bezug auf die Umsetzung der Informationsagenten im Vergleich zu MySpiders vorgenommen. Die Informationsagenten in MySpiders besuchen im Laufe ihres Lebens mehrere Webdokumente nacheinander, solange bis ihnen die Energie ausgeht. Im Unterschied dazu ruft ein DocClassifier-Agent jeweils nur ein einziges Dokument auf. Links können in DocAssistant daher nur durch Reproduktion verfolgt werden.

In MySpiders wird die Auswahl der zu verfolgenden Links in einem Informationsagenten über ein neuronales Netz gesteuert. Für den Verlauf der Suche wäre der Einsatz eines neuronalen Netzes zwar vorteilhaft, jedoch würden die Analyse und der Einsatz von neuronalen Netzen über das Kerngebiet dieser Arbeit hinausgehen. Es wurde daher eine möglichst einfache Umsetzung der Informationsagenten angestrebt. In DocAssistant werden, statt der Auswahl eines bestimmten Links, eine bestimmte Anzahl von Links auf der jeweiligen Webseite ausgewählt und für jeden dieser Links ein neuer DocClassifier erzeugt. Diese rufen dann die weiter zu verfolgenden Links auf.
Die neu erzeugten DocClassifier erben die Energie ihrer Eltern. Die Reproduktion kostet jedoch einen Energiepunkt, so daß die Nachkommen jeweils einen Energiepunkt weniger zur Verfügung haben. Auf diese Weise ist das Aufrufen einer Webseite, ebenso wie in MySpiders, mit Energiekosten verbunden.
Reproduktion findet in MySpiders bei einem bestimmten Energiepegel statt. In DocAssistant dagegen findet Reproduktion auf jeder Webseite statt, solange der Agent einen Energielevel von über Null besitzt. Sinkt die Energie auf Null findet keine Reproduktion mehr statt und der Agent stirbt.

Diese Veränderung hat zusätzlich den Effekt, daß die Agentenpopulation im Vergleich zu MySpiders schneller wächst und sich dadurch die Breite der Suche in DocAssistant erhöht. Zudem ist es nicht mehr nötig eine Back-Link zu speichern. Da ein Agent bei einer Sackgasse nicht wieder zum Ausgangsdokument zurückkehren muß, werden Übertragungskosten eingespart. Der Nachteil ist jedoch, daß das Finden eines relevanten Dokumentes in einer Sackgasse nicht belohnt wird, da sich der Agent dort nicht mehr vermehren kann.

Der Einsatz von LSI

Wie im vorherigen Kapitel erläutert wird aus den vom Benutzer bereitgestellten und vorklassifizierten Dokumenten ein LSI-Klassifikator erstellt. Der Aufbau des Klassifikators erfolgt wie in Kapitel 3.2.4. beschrieben.

Jeder DocClassifier verwendet eine Kopie des über das Nutzerprofil erstellten Klassifikators zur Klassifizierung des jeweils von ihm aufgerufenen Dokumentes.
Der Klassifikator besteht aus der dimensionsreduzierten Term-Dokument-Matrix bzw. aus den Matrizen U, S und V^T (vlg. Kapitel 3.2.4.). Jeder DocClassifier berechnet nun den Dokumentenvektor für das von ihm aufgerufene Dokument und vergleicht ihn mit Hilfe der Cosinus-Distanzfunktion mit den Vektoren der bereits vorhandenen Dokumente. Ist das gefundene Dokument einem oder mehreren der vorhandenen Dokumente besonders ähnlich, überprüft der Agent in welcher Kategorie diese gespeichert sind und legt das neue Dokument ebenfalls in der entsprechenden Kategorie ab.

5.3. Implementation

In diesem Kapitel soll nun die Implementation von DocAssistant beschrieben werden.

5.3.1. Implementation des Multiagentensystems

DocAssistant entspricht in seiner Architektur dem typischen Multiagentensystem von Sycara (1998). Die DocClassifier-Agenten arbeiten selbständig und unabhängig voneinander und erfüllen jeweils ihre spezielle Aufgabe. Dabei findet unter den Agenten keinerlei Kommunikation oder Koordination statt. Das Verhalten eines einzelnen Agenten wird mehrheitlich durch seine Umgebung bestimmt. Trotzdem führen die Leistungen der einzelnen Agenten zur Lösung des vom Benutzer gestellten Problems, nämlich im Web relevante Publikationen zu finden. Die Intelligenz, die zum Lösen dieser Aufgabe benötigt wird ist, eine emergente Eigenschaft des Systems. Emergente Eigenschaften sind häufig in Populationen von einfachen Individuen zu finden, wie in (Di Marzo Serugendo, 2006) dargestellt.

Für die Implementation von DocAssistant wurde Jadex als Plattform und Rahmenwerk gewählt. Ein Vorteil von Jadex ist die Möglichkeit, Java bei der Implementation einzusetzen. Auf diese Weise war es möglich das LSI-Modul quasi unabhängig von den Agenten zu implementieren. Die Agenten haben Zugriff auf das Modul über normale Java-Objekte. Zudem konnten Bibliotheken, wie beispielsweise für die Singulärwertzerlegung oder das Einlesen von PDF-Dokumenten einfach in das System integriert werden.

Agenten werden in Jadex über XML-Files (Agent-Definition-Files) definiert. Hier werden das Wissen, die Ziele und die Pläne, über die der Agent verfügt, aufgeführt. Die Implementation dieser Bausteine erfolgt über normale Java-Klassen.

Als Beispiel sei hier das Agent-Definition-File des Manager-Agenten angegeben:

```
<agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="http://jadex.sourceforge.net/
jadex.xsd"
      name="Manager"
      package="docclassifier.jadex">

<beliefs>
      <!-- starting URL -->
      <belief name="url" class="String">
            <fact>""</fact>
      </belief>
</beliefs>


<plans>
      <!-- Plan to create the GUI, and update when the agent
population changes. -->
      <plan name="guiManager" initial="true">
            <body>new GUIManagerPlan()</body>
            <waitqueue>
                  <messageevent ref="update_gui"/>
            </waitqueue>
```

```
      </plan>
      <!-- Plan to start the first DocClassifier with the starting
URL -->
      <plan name="agentManager" initial="true">
            <body>new AgentManagerPlan()</body>
      </plan>
</plans>
```

Der Manager verfügt über zwei Pläne, den „GUIManagerPlan“ für die Verwaltung der grafischen Benutzeroberfläche und den „AgentManagerPlan“, für das Starten eines DocClassifier-Agenten. Sein Wissen besteht aus der Start-URL, die anfangs aus einem leeren String besteht und über das Benutzerinterface gesetzt werden kann.

Kommunikation findet in Jadex über Nachrichten statt. Interaktion zwischen den Agenten findet, wie gesagt, nur über die Reproduktion statt. Ausnahme bildet die Interaktion der DocClassifier mit der GUI. Dies geschieht über den Manager-Agenten. Nach der Erzeugung sendet der DocClassifier eine Nachricht an den Manager mit dem Inhalt “update”. Dies veranlasst den Manager die Hierarchie der im System vorhandenen Agenten um diesen DocClassifier-Agenten zu erweitern und die veränderte Hierarchie darzustellen. Stirbt ein DocClassifier, sendet er als letzte Handlung ebenfalls eine Nachricht an den Manager mit dem Auftrag ihn wieder aus der Hierarchie zu löschen.
Zudem ist der Manager in der Lage Agenten zu veranlassen “Selbstmord” zu begehen, wenn zu viele Agenten im System vorhanden sind. Dann sendet er als Antwort auf die Update-Anfrage den Hinweis, daß bereits zu viele Agenten existieren und der Agent deshalb nicht mehr dargestellt wird. Daraufhin beendet der DocClassifier alle Aktionen und stirbt. Es findet dann auch keine Reproduktion des Agenten mehr statt.

5.3.2. Implementation des LSI-Moduls

Für die Erstellung der Term-Dokument-Matrix müssen, wie in Kapitel 3.1.2. dargestellt, zunächst die Feature-Werte aus den Webdokumenten extrahiert werden. Als Feature-Werte wurden die einzelnen Worte der Inhalte der Webdokumente gewählt. Hier sind für die beiden Formate PDF und HTML, die DocAssistant verarbeiten kann, zwei verschiedene Vorgehensweisen nötig.
Für die Extraktion des Textes eines PDF-Dokumentes wurde die Open Source Java-Bibliothek *PDFBox* (www.pdfbox.org) eingesetzt. Mit PDFBox ist es möglich den Text eines PDF-Dokumentes als Java-String zu extrahieren. Dieser kann dann über eine einfache Methode in einzelne Worte aufgespalten werden.
Um den Text eines HTML-Dokumentes zu extrahieren, ist es nötig einen HTML-Parser einzusetzen. Für die Implementation in DocAssistant wurde der *HTMLParserWrapper* von T. Loton (vgl. Loton, 2002) verwendet, der auf dem Java-Paket *javax.swing.text.html* und den darin enthaltenen Klassen *HTMLEditiorKit.ParserCallback* und *parser.ParserDelegator* basiert.

Um die Featuremenge zu reduzieren, wurde eine Liste von Stopwords verwendet, mit deren Hilfe Worte wie a, about, but, then, this, where, which aus den extrahierten Texten herausgefiltert wurden. Außerdem wurde zusätzlich ein Word-Stemming-Algorithmus eingesetzt[1]. Da sowohl der Stemmer als auch die Liste der Stopwords auf der englischen

[1] Hier wurde der Paice/Husk Stemmer verwendet. Der Paice/Husk Stemmer wurde am Computing Department der Universität Lancaster entwickelt (vgl. Paice, 1990). Die original Pascal-Implementation wurde von Chris O'Neill (Student der Universität Lancaster) in Java übersetzt.

Sprache basieren, können von DocAssistant nur Dokumente, die in Englisch verfaßt sind, klassifiziert werden. Um auch Texte in anderen Sprachen erfassen zu können, müssen entweder Stopwords und Stemmer in den entsprechenden Sprachen eingebunden werden oder ganz auf diese verzichtet werden. Ohne eine Reduktion der Featuremenge erhöhen sich jedoch die Kosten für die Singulärwertzerlegung drastisch, so daß in DocAssistant nicht auf diese Mittel verzichtet wurde.
Als weitere Maßnahme zur Reduktion der Featuremenge wurden, wie in (Deerwester et al., 1990) vorgeschlagen, nur Worte in die Featuremenge mit aufgenommen, die in mehr als einem Dokument vorkommen. Wie Deerwester et al. (1990) darlegen, haben solche Worte wenig oder gar keinen Einfluß auf die Vorgänge des LSI-Verfahrens.

Die so erhaltenen Features werden in DocAssistant über die Bags-of-Words-Methode in Vektordarstellung überführt (siehe Kapitel 3.1.2.). Anschließend wird eine Term-Dokument-Matrix über alle vorhandenen Dokumente erzeugt. Diese wird dann wie in Kapitel 3.2.4. beschrieben mit Hilfe der Singulärwertzerlegung dimensionsreduziert. Für die Implementation der Singulärwertzerlegung in Java wurde die Open Source Bibliothek *Colt* (hoschek.home.cern.ch/hoschek/colt/) verwendet. Als Wert für k wurde eine Dimensionsreduktion um 40% gewählt.

Für die Klassifizierung verfügt jeder DocClassifier-Agent über ein LSI-Modul. Dies besteht aus Kopien der drei reduzierten Matrizen U, S und V^T, einer Liste aller vorhandenen Dokumente, einer Liste aller Features, einer Stopwords-Liste und einer Liste mit Stemming-Regeln. Somit ist der Agent in der Lage, das jeweils aufgerufene Webdokument, wie in Kapitel 3.2.4. beschrieben, zu klassifizieren.

5.4. Funktionsweise des DocAssistant-Systems

In diesem Abschnitt soll die Funktionsweise des DocAssistant-Systems erläutert werden. Nach einer theoretischen Betrachtung des Suchvorgangs wird anhand eines Beispiels der Einsatz des DocAssistant-Systems demonstriert.

In Abbildung 21 ist der theoretische Ablauf der Informationssuche dargestellt. Zu sehen ist ein Teil des WWWs mit mehren untereinander verlinkten HTML- und PDF-Dokumenten. (Dies könnte beispielsweise die Homepage eines Autors mehrerer wissenschaftlicher Publikationen sein.) Zu Beginn ist nur ein DocClassifier-Agent vorhanden. Dieser ist mit einer Anfangsenergie von 3 ausgestattet. Für den Link den er auf der Seite findet startet er einen neuen DocClassifier-Agenten, der als Anfangsenergie den Wert 2 erhält. Die von diesem aufgerufene Seite wird als nicht relevant eingestuft. Für jeden Link, den der Agent findet, startet er nun einen weiteren DocClassifier, die jeweils mit der Anfangsenergie 1 starten. In einem Zweig des Baumes befindet sich jedoch eine Webseite mit relevantem Inhalt. Daher erhöht sich die Energie des Agenten, der diesen Zweig verfolgt wieder auf 2. Auf der nächsten Seite folgt eine weitere relevante Seite, daher bleibt die Energie des dort gestarteten Agenten konstant und er kann sich weiter vermehren.
Auf diese Weise gelangen die Agenten in die beiden unteren Zweige des Baumes, in denen sich die für den Nutzer relevanten PDF-Dokumente befinden. (Dies sind die in der Abbildung markierten Bereiche B und C). Der Zweig, in dem sich mehrere nicht relevante PDF-Dokumente befinden, wird nicht von den Agenten besucht (in der Abbildung Bereich A), ohne daß eine Koordination zwischen den Agenten stattfindet.

Ausgehend von der Annahme, daß inhaltlich miteinander verwandte Webseiten untereinander verlinkt sind, wählt die emergente Intelligenz des Systems den Zweig des Baumes aus, in dem sich die relevanten Dokumente befinden. Dies geschieht allein dadurch, daß sich die Agenten in diesem Zweig des Baumes weiter vermehren. Der Agent in dem nicht relevanten Teil des Baumes produziert keine Nachkommen. Er stirbt, bevor er die Dokumente im Bereich A überhaupt erreicht.

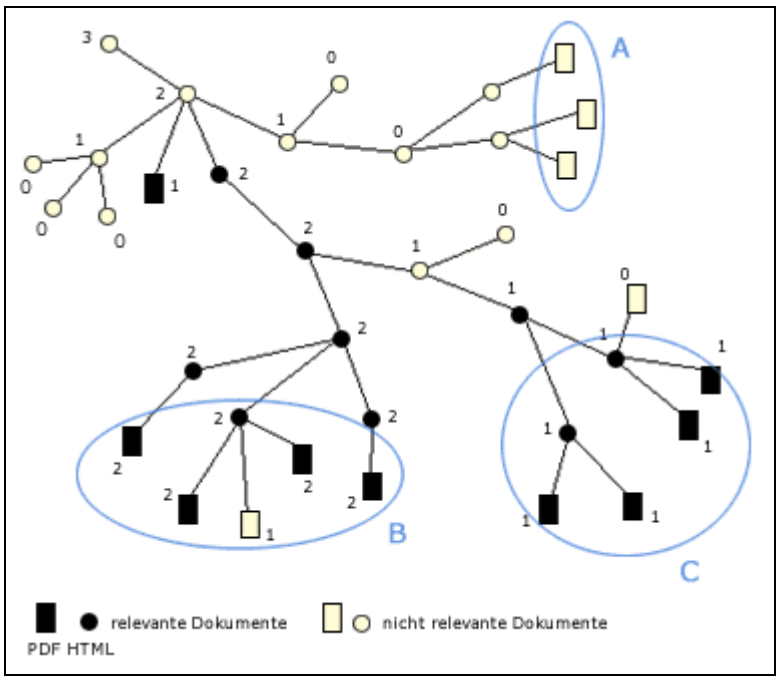

Abbildung 22: Darstellung der Informationssuche in DocAssistant

Im Folgenden soll der Ablauf eines Suchvorgangs anhand eines konkreten Beispiels dargestellt werden.

Für die Suche nach wissenschaftlichen Publikationen zu den Themen: Agenten, Data-Mining, Klassifikation, LSI und Web-Mining wurden seitens des Benutzers bereits fünf Ordner in seinem Profil angelegt (siehe Abbildung 23).

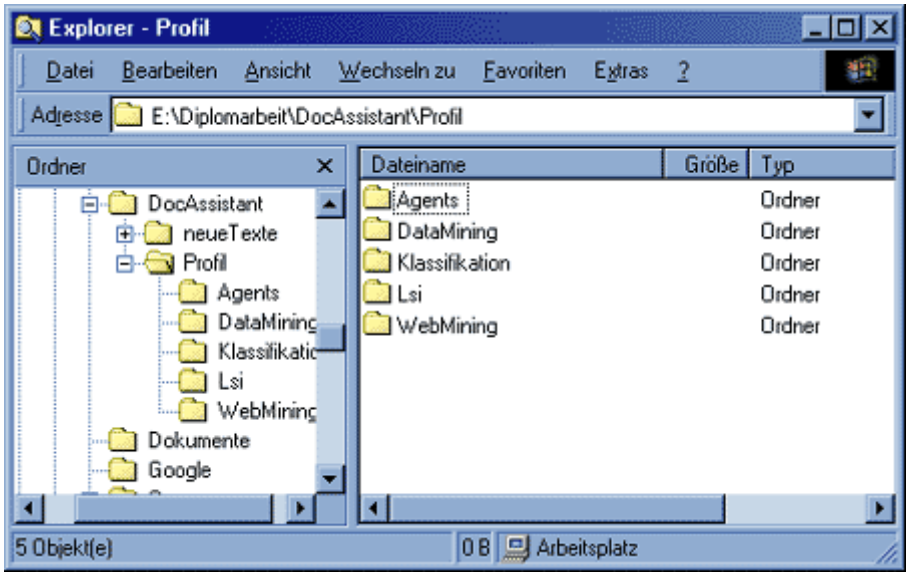

Abbildung 23: Darstellung eines Nutzerprofils

In jede Kategorie wurden jeweils zum Thema passende Publikationen in Form von HTML- und PDF-Dokumenten gespeichert (Abbildung 24).

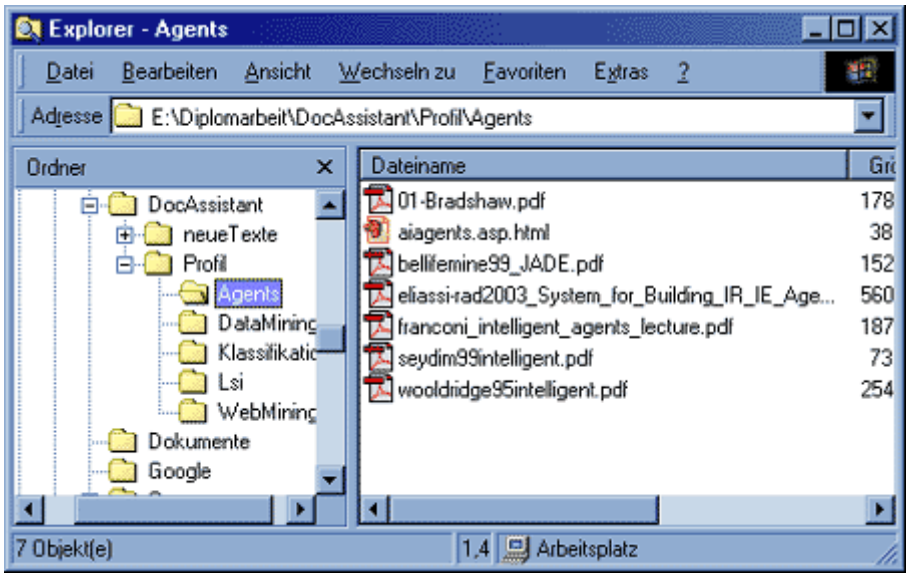

Abbildung 24: Trainingsdokumente im Profil des Nutzers

Anhand dieser Trainingsdokumente wird von DocAssistant ein LSI-Klassifikator erstellt. Nach dem Start der DocAssistant-Anwendung gibt der Benutzer eine Start-URL in das Query-Feld von DocAssistant ein.

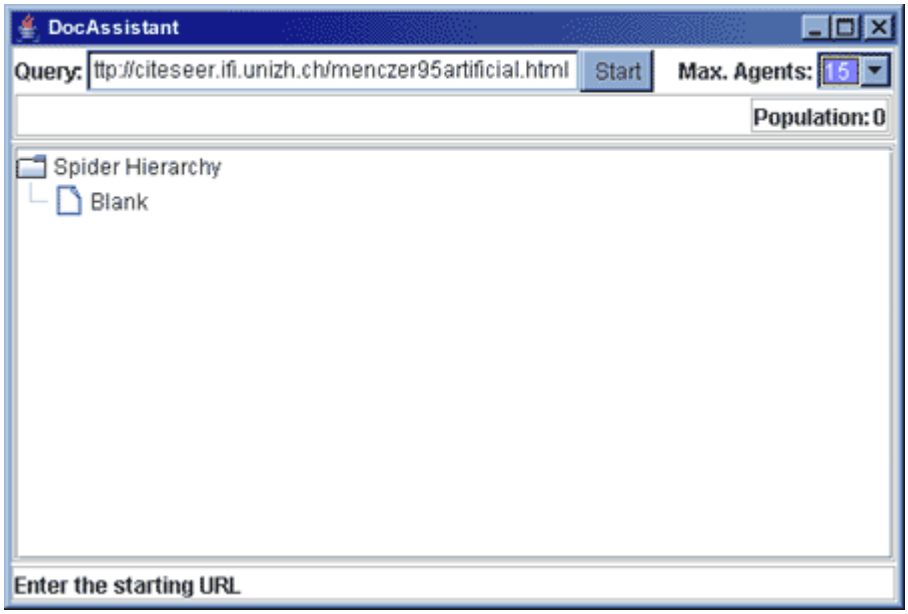

Abbildung 25: DocAssistant Start

In diesem Fall ist es die Seite einer Publikation zum Thema Informationsagenten, die in der Literaturdatenbank Citeseer gefunden wurde.

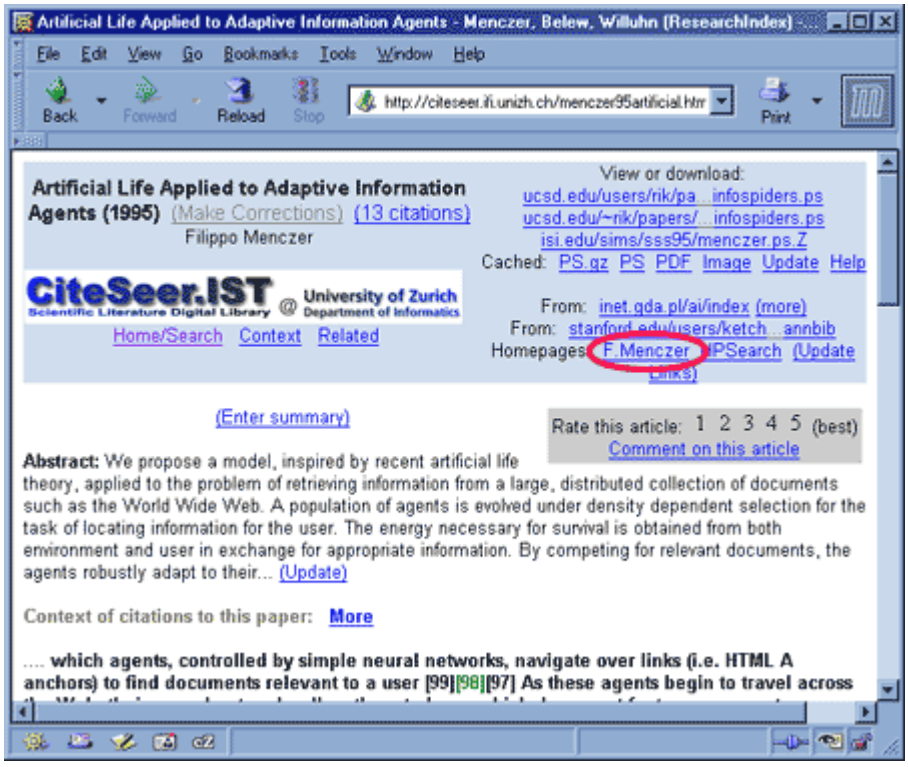

Abbildung 26: Seite einer Publikation in Citeseer

DocAssistant startet nun seine DocClassifier-Agenten, die die Links der Seite weiterverfolgen. Durch einen Klick auf einen der DocClassifier im Benutzerinterface kann der Benutzer jederzeit Details zu dem ausgewählten Agenten aufrufen. In einem

zusätzlichen Fenster werden ihm Informationen über Status des Agenten, seine Energie und die jeweils besuchte URL angezeigt.

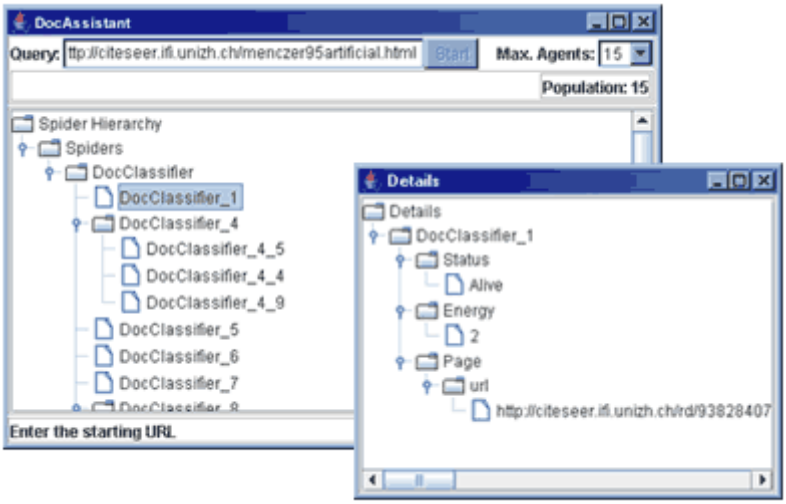

Abbildung 27: Detailansicht eines DocClassifier-Agenten

Auf der Startseite befindet sich ein Link auf die Homepage des Autors der Publikation, Filippo Menczer (in Abbildung26 markiert).
Von der Homepage führt wiederum ein Link zu weiteren Publikationen des Autors (siehe Abbildung 28).

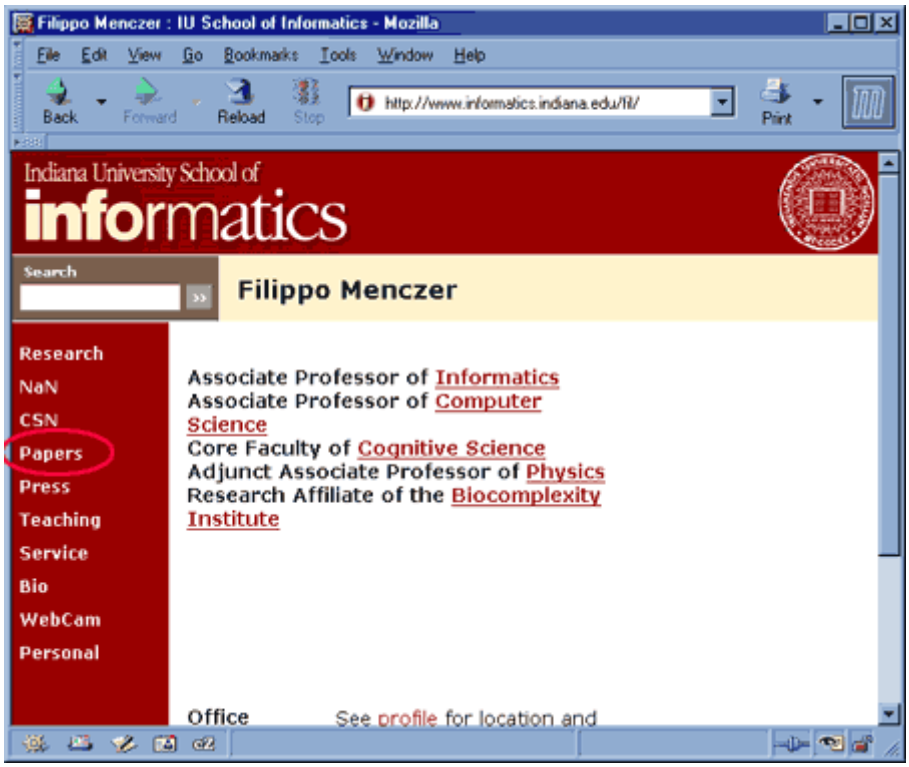

Abbildung 28: Link zu den Publikationen von Filippo Menczer

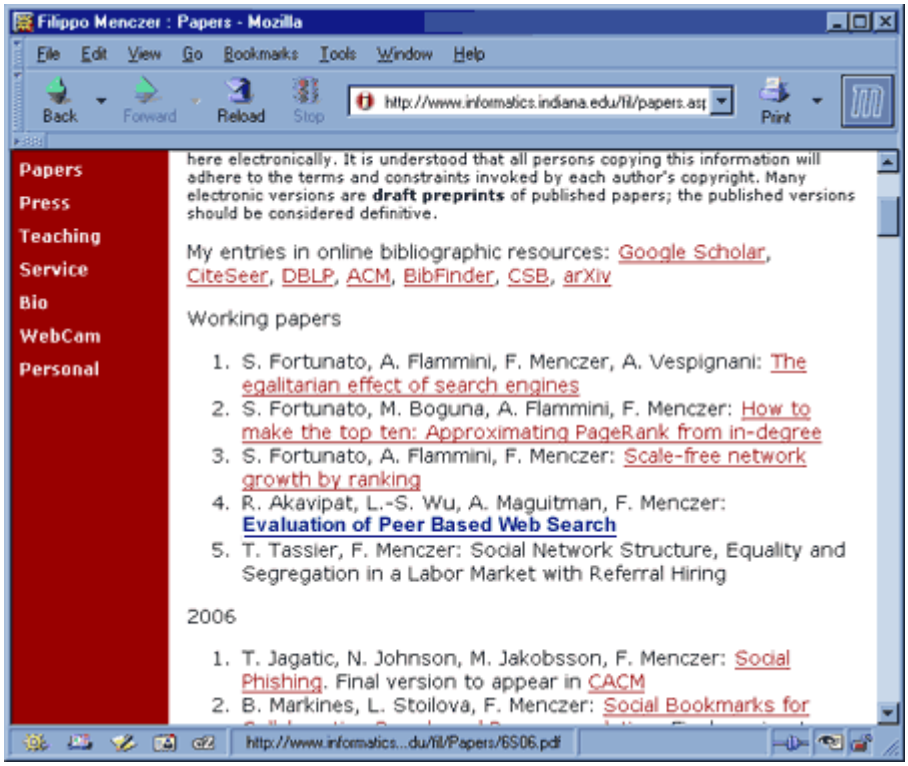

Abbildung 29: Publikationen von Filippo Menczer

In unserem Beispiel ist der Eintrag "Evaluation of Peer Based Web Search" für den Benutzer interessant, da dieses Dokument zu den Publikationen der Kategorie Web-Mining paßt, die der Nutzer in seinem Profil angelegt hat.
DocAssistant erkennt nun, wie beschrieben, anhand der im Nutzerprofil vorhanden Publikationen zum Thema Web-Mining, daß dieses Dokument für den Nutzer relevant ist und speichert es automatisch als neues Dokument in der Kategorie "Web-Mining" im Ordner "neueTexte". Zusätzlich wird eine Textdatei des selben Namens angelegt, in der die Fund-URL und der Titel der Publikation vermerkt sind.

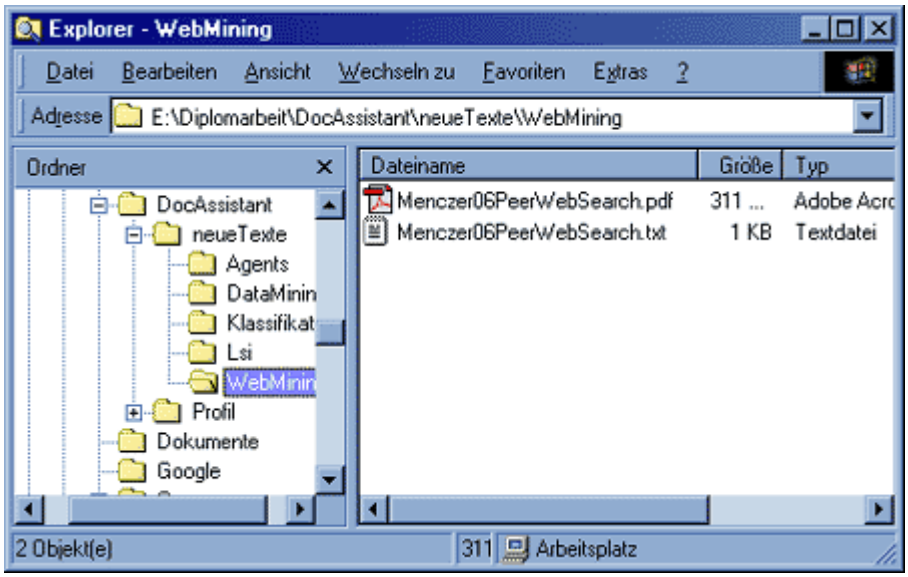

Abbildung 30: Speicherung der gefundenen Publikation als neues Dokument

5.5. Bewertung

In diesem Kapitel findet eine Bewertung des DocAssistant-Systems statt. Hierfür werden die einzelnen Bausteine des Systems gesondert betrachtet und evaluiert.

5.5.1. Bewertung des Gesamtsystems

DocAssistant verbindet die Möglichkeiten der echtzeitbasierten Websuche mit der Technologie der Klassifikation von Webdokumenten und dem Einsatz von persönlichen intelligenten Agenten.
Die Verwendung von Informationsagenten, die basierend auf dem vorgestellten evolutionären Algorithmus agieren, simulieren quasi den Suchvorgang eines Informationssuchenden, der ausgehend von einem bestimmten Ort im WWW, möglichst alle relevanten Informationen in dessen Umgebung ausfindig machen möchte. Durch die Personalisierung ist es den Informationsagenten möglich die Relevanz der einzelnen Dokumente zu beurteilen. Daher konnte durch das eingesetzte Multiagentensystem der Suchprozeß in DocAssistant vollständig automatisiert werden.
Zur Verbesserung der Darstellung von Suchergebnissen wurde die Klassifizierung der Ergebnisdokumente vorgeschlagen. In DocAssistant wird dies außerdem genutzt, um die Ergebnisdokumente nach Themen geordnet für den späteren Gebrauch zu speichern, also zu archivieren.

DocAssistant erleichtert die Suche nach Publikationen im WWW, da es den Benutzer die von ihm gesuchten Informationen anhand von Beispielen spezifizieren läßt. Es ist nicht notwendig die gewünschten Informationen auf komplizierte Art und Weise zu umschreiben, wie es beispielsweise die Keyword-bezogene Suche erfordert. Ein Beispiel zu geben ist die denkbar einfachste Methode seine Wünsche zu formulieren. Für die Agenten sind Beispiele die einfachste Art die Wünsche des Benutzers zu erlernen.

5.5.2. Validierung des LSI-Modells

Der in DocAssistant implementierte LSI-Klassifikationsmechanismus wurde für die Klassifikation von wissenschaftlichen Publikationen einem Test unterzogen.
Hierfür wurden 62 wissenschaftliche Texte aus den Bereichen: Agenten, Data-Mining, Klassifikation, LSI und Web-Mining aus dem WWW ausgewählt. Der DocAssistant-Klassifikator wurde mit 18 zufällig aus dieser Datenmenge ausgewählten Publikationen trainiert. Die übrigen 44 Dokumente wurden zum Testen von Hand in die verschiedenen Kategorien eingeordnet und anschließend durch den DocAssistant-Klassifikator klassifiziert.

Es wurden mehrere Testdurchläufe mit unterschiedlichen Werten für k, also die Anzahl der Dimensionen in der reduzierten Term-Dokument-Matrix (siehe Kapitel 3.2.4.), durchgeführt. Die höchste Fehlerrate ergab sich bei einer starken Dimensionsreduktion der Matrix (Reduktion der Dimensionen um 70%). Hier ordnete der DocAssistant-Klassifikator 8 von 44 Dokumenten falsch ein. (Dies entspricht einer Fehlerrate von18,2%). Das beste Ergebnis konnte bei einer Dimensionsreduktion von 40% erzielt werden. Hier Betrug die Anzahl der Fehlklassifikationen nur 3 von 44 also 6,8%. Bei einer zu niedrigen

Wahl von k stieg die Fehlerrate dagegen wieder an. Bei einer Dimensionsreduktion der TD-Matrix um nur 20% betrug die Fehlerrate 13,6%.

Insgesamt wird bei der richtigen Wahl von k eine relativ geringe Fehlerrate erzielt. Auffällig ist dieses Ergebnis in Relation zu der sehr kleinen Anzahl von Trainingsdokumenten. Trotz einer minimalen Anzahl von Trainingsbeispielen in einigen Klassen wurde die Mehrheit der Testdokumente richtig klassifiziert. In der Kategorie „Klassifikation“ waren beispielsweise nur 3 Trainingsdokumente vorhanden. Trotzdem traten in dieser Kategorie bei der Klassifizierung der Testdokumente keine Fehler auf. Da LSI auf der Basis der Ähnlichkeit von Dokumenten arbeitet können auch mit sehr kleinen Trainingsdatenmengen gute Ergebnisse erzielt werden.
Dieses Ergebnis deckt sich mit den Erfahrungen, die bei der Evaluation des Systems MELISSA gemacht wurden (siehe Kapitel 4.2.2.). In dieser Untersuchung wurde allerdings, wie auch bei der Evaluation von MELISSA, nur eine beschränkte Menge von Testdokumenten verwendet. In der hier eingesetzten Testmenge befanden sich nur 9 Dokumente, die für keine der Klassen relevant waren. Es wurden von DocAssistant alle Dokumente bis auf eines als nicht relevant erkannt. Dies ist ein Hinweis auf einen guten Precision-Wert. Im World Wide Web ist die Masse der nicht relevanten Dokumente im Vergleich zu der Zahl der relevanten aber sehr viel höher. Die Klassifikationsleistung von LSI wurde, wie in Kapitel 4.2.2. erwähnt, in mehreren Experimenten evaluiert. Jedoch wurde keine Aussage über die Leistungen eines solchen LSI-basierten Klassifikators für die Suche im WWW getroffen. Um die Eignung des hier verwandten LSI-Modells für die Suche im Web abschließend beurteilen zu können, müßte ein Test mit einer sehr viel größeren Menge von nicht relevanten Dokumenten durchgeführt werden. Da eine solche Testung mit großem Aufwand verbunden ist und daher über den Rahmen dieser Arbeit hinausgehen würde, wurde auf die Durchführung in dieser Arbeit verzichtet.

5.5.3. Bewertung des Multi Agenten Systems

Wie bereits erwähnt sind Multiagentensysteme gut für den Einsatz in verteilten Systemen geeignet. Sie eignen sich daher auch besonders für Anwendungen im WWW.
In DocAssistant arbeiten die Informationsagenten unabhängig voneinander. Die Suche kann auf diese Weise parallel durchgeführt werden. Dadurch wird ein Zeitvorteil gewonnen.
Durch die Unabhängigkeit der einzelnen Agenten ist das System zusätzlich sehr robust. Tritt in einem Agenten ein Fehler auf, z.B. weil ein Dokument nicht mehr vorhanden ist oder nicht gelesen werden kann, beeinträchtigt dies die Abläufe in den anderen Agenten nicht und die Suche kann unbeeinträchtigt fortgesetzt werden.
Ein weiterer Vorteil des Multiagentensystems ist, daß es sich automatisch der Anzahl der jeweils vorhandenen Links anpaßt. Sind keine Links vorhanden, werden auch keine neuen Agenten gestartet.
Nachteilig ist die Unabhängigkeit der einzelnen Agenten jedoch beim Einsatz des LSI-Verfahrens. Für die Klassifikation braucht jeder Agent eine Kopie der LSI-Matrizen. Zudem muß bei der Klassifizierung jedesmal eine Multiplikation der Matrizen mit den Term-Vektoren des aufgerufenen Dokumentes durchgeführt werden. Wenn viele dieser Multiplikationen gleichzeitig erfolgen, was bei unabhängig voneinander arbeitenden Agenten nicht vermieden werden kann, führt dies zu einer sehr großen Speicherauslastung. Daher hängt es von der verfügbaren Systemleistung ab, wieviele Agenten zur gleichen Zeit in DocAssistant aktiv sein können. Eine Möglichkeit die Obergrenze für diese Zahl zu erhöhen, wäre jeden DocClassifier, der seine Aufgaben, also

die Klassifizierung des aufgerufenen Dokumentes und eine eventuelle Reproduktion, beendet hat, sterben zu lassen, auch wenn der Agent noch über Energie verfügt. Dies darf allerdings nur geschehen, wenn keine Nachkommen des Agenten vorhanden sind. Zur Zeit werden in DocAssistant alle Agenten am Leben gelassen, deren Energie mehr als 0 beträgt, um während der Suche nachvollziehen zu können, welche Dokumente an welchen Orten von welchen Agenten gefunden wurden.

5.5.4. Erweiterungen und Empfehlungen

Das Nutzerprofil ist in DocClassifier sehr einfach, nur über die Publikationen des Benutzers realisiert. Andere Agentensysteme betreiben hier deutlich mehr Aufwand, um aus den Aktionen des Benutzers auf seine Wünsche und Vorlieben zu schließen. Wie in Kapitel 4.3. erläutert, wurde in DocAssistant aus Gründen der Transparenz, eine möglichst einfache und vom Nutzer selbst beeinflußbare Form des Nutzerprofils gewählt.
Es könnten jedoch für die Suche zusätzlich noch weitere vom Nutzer angegebene Attribute berücksichtigt werden. Zum Beispiel könnten Worte, denen eine Besondere Bedeutung in einer Klasse zukommt, bei der Klassifikation besonders stark gewichtet werden. Beispielsweise könnte ein Benutzer angeben, daß möglichst nur Texte eines bestimmten Autors gefunden werden sollen, oder Texte in denen die Bezeichnung „Web-Mining“ explizit vorkommt.
Für die Erkennung von wissenschaftlichen Texten könnten weiter Verbesserungen erzielt werden, wenn die Struktur der Dokumente bei der Suche Beachtung finden würde. In Kapitel 4.3. wurde dies bereits vorgeschlagen. Bei HTML-Dokumenten könnte dies über HTML-Tags geschehen. Bei PDF-Dokumenten könnten spezielle Abschnitte wie die Überschrift, die Namen der Autoren oder die Kurzfassung des Textes in die Bewertung des Inhaltes gesondert einbezogen werden. Besondere Mechanismen zur Auswahl dieser Features werden bereits in Citeseer (vgl. Giles, 1998) eingesetzt. Der hier vorgestellte Prototyp könnte sicherlich von Suchmöglichkeiten, wie sie bei Citeseer gegeben sind, profitieren.

Um die Auswahl der passenden Kategorie bei der Klassifizierung durch DocAssistant für den Benutzer noch transparenter zu machen, wäre es möglich, dem Benutzer anzuzeigen, welchem der im Nutzerprofil befindlichen Dokumente das gefundene Dokument am ähnlichsten ist.

Die Vereinfachungen, die in der Implementation der Informationsagenten und des evolutionären Algorithmus gegenüber MySpiders gemacht wurden, bewirken, wie in Kapitel 5.2.4 dargestellt, daß die Agentenpopulation schneller wächst. Dies ist einerseits positiv zu bewerten, da eine größere Population einen Zeitvorteil bei der Suche bedeutet. Gleichzeitig wird auch die Breite der Suche erhöht. Andererseits steigt mit der Anzahl der Agenten auch der Ressourcenverbrauch im System. Da LSI erhöhte Anforderungen an die Systemressourcen stellt, ist hier abzuwägen, welche Form des evolutionären Algorithmus in DocAssistant günstiger ist.

In DocClassifier wurde der Mechanismus zur Auswahl der Links durch die DocClassifier-Agenten gegenüber MySpiders stark vereinfacht. Die Auswahl einer festen Anzahl von Links in der Reihenfolge ihres Vorkommens erwies sich jedoch in der Praxis als nicht besonders praktikabel. Dieses Vorgehen hat zur Folge, daß die richtige Auswahl der Links dem Zufall überlassen wird.
Der Lösungsansatz einfach alle Links einer Seite zu verfolgen, ist ebenfalls nicht realisierbar, da hier die Anzahl der Agenten schnell exponentiell ansteigen kann. Hier

sollte daher nach dem Vorbild von MySpiders zusätzlich eine Bewertungsfunktion für die Links in den Informationsagenten vorhanden sein.
Da mit Hilfe von DocAssistant möglichst wissenschaftliche Publikationen gefunden werden sollen, wäre es hier naheliegend, die Agenten nach Stichworten wie „Papers“, „Publications“, „Research“ usw. in den Linkbeschreibungen Ausschau halten zu lassen, um so die richtigen Links auszuwählen.

Durch DocAssistant wird der Suchvorgang weitestgehend automatisiert. Um eine geeignete Startseite für die Suche zu finden, muß jedoch durch den Benutzer eine Start-URL festgelegt werden. Dieser Schritt ist in dem erarbeiteten Prototypen noch nicht automatisiert, sondern muß vom Benutzer manuell vorgenommen werden. Es wäre jedoch denkbar, diesen Schritt ebenfalls in das System zu integrieren und somit zu automatisieren. Hier müßte evaluiert werden, ob sich dadurch Vorteile in Form einer Vereinfachung für den Benutzer ergeben oder eher Nachteile im Sinne einer Einschränkung der Flexibilität.

6. Zusammenfassung und Ausblick

Ziel dieser Arbeit war es ein System zu entwerfen und prototypisch umzusetzen, das wissenschaftliche Informationssuchende bei der Recherche nach Publikationen im World Wide Web unterstützt.
Für die Konzeption des Systems wurde zunächst der Anwendungsbereich, also die Informationssuche im WWW, untersucht. Hierfür wurde der Prozeß der Wissensgewinnung im World Wide Web analysiert und mit dem klassischen Knowledge Discovery in Databases (KDD) verglichen. Anhand der aufgezeigten Parallelen wurde analog zum KDD ein Prozeß für das Knowledge Discovery on the World Wide Web (KDW) entwickelt.
Anschließend wurden die Anforderungen, die wissenschaftliche Informationssuchende an eine Recherche im Web stellen, erörtert. Hierfür wurden verschiedene Suchdienste, die Informationssuchenden zur Verfügung stehen, beschrieben und die Probleme, die bei der Suche entstehen können, aufgezeigt.

Insbesondere ergaben sich Probleme beim Einsatz von Indexbasierten Suchdiensten. Diese Suchdienste versuchen das WWW durchsuchbar zu machen, indem sie das Web ständig nach neuen Dokumenten durchforsten und diese in einem Index für den Suchenden verfügbar machen. Da das WWW jedoch beständig wächst und die Inhalte sich sehr schnell ändern, führt dieses Vorgehen in Bezug auf den Index zu einem Aktualitäts- und Skallierbarkeitsproblem.
Einen Nachteil für den Benutzer weist außerdem die Keyword-basierte Suche auf. Hier wird vom Benutzer erwartet, daß er die „richtigen“ Suchworte kennt, um zu dem gewünschten Suchergebnis zu gelangen. Es ist also nötig, die gesuchten Informationen mit einigen wenigen Schlüsselworten zu beschreiben. Als eine Möglichkeit zur Vereinfachung dieses Vorgangs wurde ein Suchmechanismus vorgeschlagen, der statt einzelner Suchworte, die Angabe ganzer Texte erlaubt. Auf diese Weise können beim Suchenden bereits lokal vorhandene Publikationen für die Suche nach ähnlichen Veröffentlichungen verwendet werden.
Weitere Probleme ergaben sich bei der Darstellung der Suchergebnisse von herkömmlichen Suchmaschinen. Diese werden dem Benutzer meist in Form von Ergebnislisten präsentiert, die bei einem umfangreichen Suchergebnis schnell unübersichtlich werden. Hinsichtlich dieses Problems erwiesen sich nach Themen geordnete, geclusterte Suchergebnisse als vorteilhaft.

Aus der Analyse der Anforderungen, die wissenschaftliche Informationssuchende an eine Recherche im Web stellen, ergaben sich zwei Teilbereiche, die durch das in dieser Arbeit entwickelte System unterstützt werden sollen.
Der erste Teilbereich ist das Suchen und Auffinden von relevanten Publikationen im WWW. Von wissenschaftlichen Informationssuchenden werden hier möglichst aktuelle und qualitativ hochwertige Ergebnisse erwartet. Zudem soll die Suche nach einem bestimmten Thema ein Ergebnis liefern, das möglichst alle relevanten Publikationen umfasst.
Als zweiter Teilbereich für die Unterstützung ergab sich die Archivierung der Suchergebnisse.

In der Untersuchung des Anwendungsbereiches wurden die Anforderungen herausgestellt, die für ein System zur Recherche im Web von Bedeutung sind. Anhand dieser Anforderungen wurde der Prototyp *DocAssistant* entwickelt.
Für das Auffinden von wissenschaftlichen Publikationen im WWW wurde eine Automatisierung des Suchvorganges angestrebt. Hierfür wurden intelligente Informationsagenten eingesetzt, die insbesondere eine persönliche, individuell auf den Benutzer zugeschnittene Unterstützung bei der Suche bieten.

DocAssistant kann als Erweiterung und Ergänzung eines herkömmlichen Suchdienstes beispielsweise einer indexbasierten Suchmaschine eingesetzt werden, da die Suchergebnisse einer Suchmaschine wiederum als Ausgangspunkt für eine Suche mit DocAssistant verwendet werden können.
Um dem Problem der mangelnden Aktualität der Suchergebnisse von Suchmaschinen zu begegnen, wurde in DocAssistant ein echtzeitbasierter Suchmechanismus implementiert. Zu diesem Zweck wurde ein Multiagentensystem eingesetzt, das aus vielen unabhängig voneinander arbeitenden Informationsagenten besteht. Durch die Anwendung eines evolutionären Algorithmus, spüren die Agenten Bereiche im WWW auf, in denen sich für den Benutzer relevante Publikationen befinden. Durch die Unabhängigkeit der Agenten wird zudem eine hohe Skalierbarkeit des Systems erreicht. Auf diese Weise können schnell sehr viele Publikationen, die sich in der Umgebung der Agenten befinden, ausfindig gemacht werden.
Um die Nachteile der Keyword-basierten Suche zu umgehen, wurde DocAssistant mit einem Suchverfahren versehen, das auf der Ähnlichkeit von Dokumenten basiert. Statt eine Suchanfrage in Form von Keywords zu stellen, gibt der Benutzer Beispiele für Publikationen, nach denen gesucht werden soll, an. Diese Suchmethode wird in DocAssistant durch den Einsatz eines auf Latent Semantic Indexing (LSI) basierenden Klassifikationsmodells realisiert. Anhand des LSI-Modells ist es den Agenten möglich, die im Web gefundenen Dokumente auf Relevanz zu prüfen. Das LSI-Verfahren zeichnet sich hierbei gegenüber anderen Klassifikationsverfahren durch besondere Vorteile bei der Erkennung der Inhalte von Texten aus

Für die Unterstützung der Archivierung wurde eine Clusterstruktur eingesetzt, in die im Web gefundene Publikationen ihrem Thema nach eingeordnet werden. Die Clusterstruktur wird durch den Benutzer in Form von Kategorien, die jeweils verschiedene Themen repräsentieren, erstellt. Die von den DocAssistant-Agenten gefundenen Publikationen werden automatisch in die entsprechenden Kategorien eingeordnet und gespeichert. Im Unterschied zu anderen clusterbasierten Systemen wird in DocAssistant die Klassifizierung der Dokumente, durch das Speichern in den Kategorien, permanent gemacht. Da die Kategorien durch den Benutzer selbst erstellt werden, ergibt sich ein einfaches und für den Benutzer transparentes System, um Dokumente automatisch zu archivieren. Zudem werden die gefundenen Dokumente dem Benutzer in einer übersichtlichen Art und Weise präsentiert.

In der Evaluierung des DocAssistant-Systems erwies sich die Suche anhand von Dokumentenähnlichkeiten über LSI als vielversprechende Methode. Dieser Ansatz läßt den Benutzer seine Suchanfrage anhand von Beispielen formulieren. Dies ist vorteilhaft, da das Erläutern und Lernen anhand von Beispielen einfacher ist als das Verfassen und Deuten von komplexen Beschreibungen.
Wie sich in der Evaluierung des LSI-Klassifikators zeigte, ist LSI gut geeignet, um anhand der Beispiele des Benutzers zu lernen und so über einen Vergleich die Relevanz von Dokumenten zu beurteilen. Als Nachteile des LSI-Verfahrens sind der hohe

Ressourcenbedarf für die Singulärwertzerlegung und die übrigen Matrix-Operationen zu nennen. In Kombination mit einem Multiagentensystem, das aus vielen Einheiten mit jeweil einem LSI-Modul besteht können sich hier Probleme ergeben.

Um den für wissenschaftliche Informationssuchende wichtigen Aspekt der Qualität der Suchergebnisse zu gewährleisten, wurden für die Auswahl der Start-URL in DocAssistant die Suchergebnisse einer wissenschaftlichen Literaturdatenbank wie Citeseer vorgeschlagen. Literaturdatenbanken, die auf wissenschaftliche Publikationen spezialisiert sind, messen die Qualität von Publikationen meist über die Anzahl der Zitate, die auf diese verweisen. Eine solche Bewertungsfunktion könnte zusätzlich auch in DocAssistant als ein weiteres Relevanzkriterium Einsatz finden.

Zukünftige Verbesserungen des Systems könnten auch über die Nutzung der strukturellen Eigenschaften von Dokumenten für die Relevanzanlyse bei der Suche realisiert werden. Diese Eigenschaften können beispielsweise HTML-Tags in HTML-Dokumenten oder Überschriften und Abstracts in PDF-Dokumenten sein. Dadurch können wertvolle Hinweise auf den Inhalt und das Thema eines Dokumentes gewonnen werden.

Einsatzmöglichkeiten von DocAssistant wären auch in anderen Anwendungskontexten als dem wissenschaftlichen Bereich denkbar. Hierfür sollten dann dem Einsatzbereich entsprechende Dokumentattribute für die Relevanzanlyse während der Suche verwendet werden.

7. Literatur

Adriaans, P. und D. Zantinge (1996). "Data Mining." Addison Wesley Longman, Harlow.

Armstrong, R., D. Freitag, T. Joachims und T. Mitchell (1995). „Webwatcher: A learning apprentice for the world wide web." In: Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments.

Bellifemine, F., G. Rimassa und A. Poggi (1999). „JADE - A FIPA-Compliant Agent Framework." In: Proceedings of the 4th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents.

Berger, R. (2004) „Web Content Mining." http://www.informatik.hu-berlin.de/~berger/files/web_content_mining/einleitung.html, Abruf am 2004 -08-18.

Berry, M. W., S. T. Dumais und G. W. O'Brien (1995). „Using Linear Algebra for Intelligent Information Retrieval." In: SIAM Review 37:4 (1995), S. 573-595.

Berry, M. W., Z. Drmac und E. R. Jessup (1999). „Matrices, Vector Spaces, and Information Retrieval." In: SIAM Review 41:2 (1999), S. 335-362.

Bollacker K., S. Lawrence und C. L. Giles (1998). „CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications." In: Sycara, K. P., Wooldridge, M. (Hrsg.): Proceedings of the second international conference on autonomous agents, ACM Press, New York, S. 116-123.

Boone, G. (1998). „Concept Features in Re:Agent, an Intelligent Email Agent." The Second International Conference on Autonomous Agents (Agents '98), Minneapolis/St. Paul, 10-13.

Bratman, M. E. (1987). „Intentions, Plans, and Practical Reason." Harvard University Press: Cambridge, MA.

Bratman, M. E., D. J. Israel und M. E. Pollack (1988). „Plans and resource-bounded practical reasoning." In: Computational Intelligence, 4 (1988) S. 349-355.

Braubach, L., A. Pokahr, W. Lamersdorf (2005). "Jadex: A BDI Reasoning Engine." In: R. Bordini, M. Dastani, J. Dix und A. El Fallah Seghrouchni (Hrsg.): Multi-Agent Programming, Springer Science+Business Media Inc., USA, S. 149-174**.**

Brenner, W., R. Zarnekow und H. Wittig (1998): „Intelligente Softwareagenten. Grundlagen und Anwendungen." Berlin et al.: Springer.

Brian, H. D. und M. H. Garzon (2001). „MELISSA: Mobile Electronic LSA Internet Server Search Agent." In: Zhong, N. , Yao, Y., Liu, J., Oshuga, S. (Hrsg.): Web Intelligence: Research and Development, Proceedings of the 1st Asia Pacific Web Conference on Web Intelligence, LNAI 2198, Springer-Verlag, Berlin, S. 335-339.

Brooks, R. A. (1986). „A Robust Layered Control System for a Mobile Robot." In: IEEE Journal of Robotics and Automation RA-2(1):14-23.

Brooks, R. A. (1991). „How to build complete creatures rather than isolated cognitive simulators." In: K.VanLehn (Hrsg.): Architectures for Intelligence, S. 225-239, Lawrence Erlbaum Assosiates, Hillsdale, NJ.

Burke, R., K. Hammond, V. Kulyukin, S. Lytinen und S. Schoenberg (1997). „Natural Language Proessing in the FAQ Finder System: Results and Prospects." In: Working Notes AAAI Spring Symposium NLP for the WWW, Stanford University, CA.

C. Chekuri, M. H. Goldwasser, P. Raghavan und E. Upfal (1997). „Web search using automatic classificationProc. Sixth Int. World Wide Web Conference, Santa-Clara, CA.

Chen, L. und K. Sycara (1998). „WebMate: A personal agent for browsing and searching." In: Sycara, K. P., Wooldridge, M. (Hrsg.): Proceedings of the second international conference on autonomous agents. St. Paul, MN, USA: ACM Press, S.132-139.

Craven, M. und S. Slattery (2001). „Relational Learning with Statistical Predicate Invention: Better Models for Hypertext."

Degeratu, M. und F. Menczer (2000). „Complementing Search Engines with Online Web Mining Agents."

Deerwester, S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, T. K. Landauer und R. A. Harshman (1990). „Indexing by Latent Semantic Analysis." In: Journal of the American Society of Information Science, vol. 41, no. 6, S. 391-407.

Deuflhard, P. und A. Hohmann (2002). „Numerische Mathematik I, Eine algorithmisch orientierte Einführung." de Gruyter, Berlin.

Di Marzo Serugendo, G., M.-P. Gleizes, A. Karageorgos (2006). „Self-organisation and emergence in MAS: an overview." Informatica, Slovene Society Informatika, 30(1), S. 45-54.

Domingos, P und M. Pazzani (1997). „On the Optimality of the Simple Bayesian Classifier under Zero-One Loss." In: Machine Learning, 29:103-130.

Eliassi-Rad, T. und J. Shavlik (2001). „Intelligent Web Agents that Learn to Retrieve and Extract Information."

Etzioni, O. (1996). „The World Wide Web: Quagmire or Gold Mine?" In: Communications of the ACM, vol. 39, no.11, S. 65-68.

Fayyad, U., G. Piatetsky-Shapiro und P. Smyth (1996). „From Data Mining to Knowledge Discovery in Databases." AI Magazine 17(3), S. 37-54.

Franconi, E. „Discription Logics Introductory Lecture"

Frieze, A., R. Kannan und S. Vempala (1998). "Fast Monte-Carlo Algorithms for Finding Low Rank Approximations." In: Proceedings of 1998 FOCS, S. 370–378.

Frommholz, I. (2001). „Categorizing Web Documents in Hierarchical Catalogues." In: Proc 23rd Colloquium on Information Retrieval Research. Darmstadt.

Gauch, S., J. Chaffee und A. Pretschner (2003). „Ontology-based Personalized Search and Browsing." Journal of Web Intelligence and Agent Systems 1(3-4), S. 219-234.

Gao, X. und L. Sterling (1997). „Using Limited Common Sense Knowledge to Guide Knowledge Acquisition for Information Agents."

Georgeff, M. P. und A. L. Lansky (1987). „Reactive reasoning and planning." In: Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), S. 677-682, Seattle, WA.

Giles, C. L., K. Bollacker und S. Lawrence (1998). „CiteSeer: An automatic citation indexing system." In: Proceedings of the 3rd ACM Conference on Digital Libraries, Pittsburgh, PA, S. 89-98.

Godoy, D. und A. Amandi (2005). „User Profiling in Personal Information Agents: A Survey." The Knowledge Engineering Review, Cambridge University Press, Vol 00:0, 1-33.

Goldberg, D. E. (1989). „Genetic Algorithms in Search, Optimization and Machine Learning." Addison-Wesley.

Golub, G. H. und C. F. van Loan (1989). „Matrix Computations. Johns Hopkins University Press." Baltimore, MD, 2nd edition.

Hagedorn, J., N. Bissantz und P. Mertens (1997). „Data Mining (Datenmustererkennung): Stand der Forschung und Entwicklung." In: Wirtschaftsinformatik 39, S. 601-612.

Heath, M. T. (2002). „Scientific Computing: An Introductory Survey." Second Edition, McGraw-Hill, New York.

Han, J. und M. Kamber (2001). „Data Mining: Concepts and Techniques. Morgan Kaufmann.

Höppner, F., F. Klawonn, R. Kruse, T. Runkler (1999). „Fuzzy Cluster Analysis." Wiley, Chichester.

Hsu W. und S. Lang (1999). "Classification Algorithms for NETNEWS Articles." In: Proceedings of the 8th International Conference on Information and Knowledge Management, S. 114-121.

Hull, D. A. (1994). „Information Retrieval Using Statistical Classification." PhD thesis, Standford University.

Joachims, T. (1997). „Text Categorization with Support Vector Machines: Learning with Many Relevant Features." In: Nédellec, C. und Rouveirol, C. (Hrsg.): Proceedings of ECML-98, 10th European Conference on Machine Learning, number 1398, S. 137–142, Springer Verlag, Heidelberg.

Klusch, M. (2000). „Intelligente Informationsagenten für Wissensentdeckung und Data Mining im Internet." In: Hippner, H.; Küsters, U.; Meyer, M.; Wilde, K. (Hrsg.): Handbuch Data Mining im Marketing. Vieweg. Wiesbaden.

Koch, T. (1997). „Literature about search services", 30.12.1997. http://www.ub2.lu.se/desire/radar/lit-about-search-services.html, Abruf am: 2004-12-08.

Kosala, R. und H. Bockeel (2000): „Web mining research: A survey." In: SIGKDD Explorations, Vol. 2, S. 1-15.

Landauer, T.K. und S.T. Dumais (1997). „A solution to Platos problem: The Latent Semantic Analysis theory of the acquisition, induction and representation of knowledge." Psychological Review 104(2), S. 211-240.

Lewandowski, D. (2005). „Web Information Retrieval. Technologien zur Informationssuche im Internet." DGI, Frankfurt am Main.

Lewis, D. D., R. E. Schapire, J. P. Callan, und R. Papka (1996). „Training algorithms for linear text classifiers." In: Proceedings of SIGIR-96, 19th ACMInternational Conference on Research and Development in Information Retrieval (Zürich, CH, 1996), S. 298–306.

Lieberman, H. (1995) „Letizia: An agent that assists web browsing." In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), S. 924-929.

Liu, X. und W. B. Croft (2004). „Language models: Cluster-based retrieval using language models." In: Proceedings of the 27th annual international conference on Research and development in information retrieval.

Liu, T., Z. Chen, B. Zhang, W. Ma und G. Wu (2004). „Improving Text Classification using Local Latent Semantic Indexing." Fourth IEEE International Conference on Data Mining (ICDM'04) November 01 - 04, 2004 Brighton, United Kingdom.

Loton, T. (2002). „Web Content Mining with Java: Techniques for exploiting the World Wide Web." John Wiley and Sons Ltd.

Lu, S. Y. und K. S. Fu. (1978). „A sentence to sentence clustering procedure for pattern analysis." IEEE Transactions on Systems Mans and Cybernetics, 8:381–389.

Maes, P. (1994). „Agents that Reduce Work and Information Overload." In: Communications of the ACM 37 (7), 31-40.

Mitchell, T. (1996). „Machine Learning", McGraw Hill.

Mladenić, D. (1998). „Turning Yahoo into an Automatic Web-Page Classifier." In: Prade, H. (ed.) 13th European Conference on Artificial Intelligence ECAI 98, S. 473-474.

Newell, A. und H. A. Simon (1976). „Computer Science as Empirical Enquiry: Symbols and Search." Communications of the ACM, 19:113–126.

Nwana, H. S. (1996). „ Software Agents: An Overview." In: Knowledge Engineering Review, Vol. 11, No 3, pp.1-40.

Paice, C.D., (1990). „Another Stemmer", SIGIR Forum 24 (3): 56-61.

Pal, S. K., V. Talwar und P. Mitra (2002). „Web Mining in Soft Computing Framework: Relevance, State of the Art and Future Directions." In: IEEE Transactions on Neural Networks, Vol. 13(5), 2002, S.1163 - 1177.

Pant, G. und F. Menczer (2002). „MySpiders: Evolve your own intelligent Web crawlers." Autonomous Agents and Multi-Agent Systems 5(2): 221-229.

Papadimitriou, C. H., H. Tamaki, P. Raghavan und S. Vempala (1998). „Latent semantic indexing: A probabilistic analysis." In: Procedings of the ACM Conference on Principles of Database Systems (PODS), S. 159–168.

Pazzani, M. J. und D. Billsus (1997). „Learning and Revising User Profiles: The Identification of Interesting Web Sites." Machine Learning, 27 (3), S. 313-331.

Pokahr, A., L. Braubach und A. Walczak (2005). „Jadex User Guide". http://vsis-www.informatik.uni-hamburg.de/projects/jadex/, Abruf am: 2005-09-01.

Potamias G., V. Raxenidis und A. Papadakis (2001). „Personalized Classification of Web Documents." In: Procs 8th Panhellenic Conference in Informatics, Nicosia, Cyprus, vol. 2, S. 213-222.

Salton, G. (1971). „Automatic indexing using bibliographic citations." Journal of Documentation 27 (1971), 98–110.

Salton, G. und M. McGill (1983). „Introduction to modern information retrieval." New York: McGraw-Hill.

Sebastiani, F. (2002). „Machine learning in automated text categorization." ACM Computing Surveys 34(1), 1–47.

Sycara, K. (1998). „Multiagent Systems." American Association for Artificial Intelligence, 0738-4602-1998.

Tsukada, M., T. Washio und H. Motoda (2001). „Automatic Web-Page Classification by Using Machine Learning Methods." In: Zhong, N. , Yao, Y., Liu, J., Oshuga, S. (Hrsg.): Web Intelligence: Research and Development, Proceedings of the 1st Asia Pacific Web Conference on Web Intelligence, LNAI 2198, Springer-Verlag, Berlin, S. 303-313.

Vapnik, V. und A. Chervonenkis (1974). „Theory of Pattern Recognition" [russisch]. Nauka, Moscow. (Deutsche Übersetzung: Wapnik, W. und A. Tscherwonenkis (1979). „Theorie der Zeichenerkennung." Akademie-Verlag, Berlin.)

Wooldridge, W. und N. R. Jennings (1995). „Intelligent Agents." Springer-Verlag, Berlin.

Xu, J.(2003). „Design and Implementation of A Web Mining Research Support System." Diss. Proposal November 2003 Als Manuskript gedruckt.

Yang, Y. und X. Liu (1999). „A re-examination of text categorization methods." In: Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, S 42-49.

Yu, C., J. Cuadrado, M. Ceglowski und J. S. Payne (2003). „Patterns in Unstructured Data Discovery, Aggregation, and Visualization." http://javelina.cet.middlebury.edu/lsa/out/cover_page.htm, Abruf am 2005-06-01.

Abbildungsverzeichnis