Diplomarbeit

Einsatz eines
intelligenten, lernenden Agenten
für das World Wide Web

Thorsten Joachims

Diplomarbeit

am Fachbereich Informatik

der Universität Dortmund

4. Dezember 1996

Betreuer:

Prof. Dr. Katharina Morik

Prof. Dr. Norbert Fuhr

Zusammenfassung

Motiviert durch WebWatcher, einen Tour-Guide-Agenten für das World Wide Web (WWW), wird die Frage untersucht, wie das von WebWatcher benötigte Wissen mit Methoden des maschinellen Lernens automatisch gelernt werden kann. WebWatchers Aufgabe ist es, Benutzer beim Browsing auf dem WWW zu begleiten und jeweils vielversprechende Hyperlinks auf der aktuellen Seite hervorzuheben. Zwei Informationsquellen, aus denen Wissen hierfür gelernt werden kann, werden aufgezeigt. Zum einen wird gezeigt, wie mit Methoden der Textkategorisierung aus Touren von vorangegangenen Benutzern gelernt werden kann. Zum anderen wird das Problem mit Methoden des Reinforcement Learning modelliert. Durch Analyse der Hypertextstruktur und der Dokumentinhalte werden dem Benutzer Pfade durch das WWW vorgeschlagen, die möglichst viel relevante Information enthalten. Beide Ansätze werden experimentell evaluiert. Es zeigt sich, daß der Textkategorisierungsansatz eine höhere Performanz erzielt, wenn genügend Benutzerinteraktionen als Trainingsbeispiele zur Verfügung stehen. Ansonsten ist der Reinforcement-Learning-Ansatz von Vorteil, da er nicht auf solche Trainingbeispiele angewiesen ist, sondern direkt aus den Dokumenten und der Hypertextstruktur lernt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung und Überblick

Mit dem Internet und speziell dem World Wide Web (WWW) eröffnen sich neue Möglichkeiten der Informationsbeschaffung. Während in konventionellen Medien Information vielfach mit Raum verbunden war, ist der Ort, an dem sich Information auf dem WWW befindet, für den Benutzer transparent. Ein Besucher eines Museums auf dem WWW kann problemlos eine Tour durch alle Werke seines Lieblingskünstlers machen. Wollte er alle Plätze aufsuchen, an denen dessen Werke ausgestellt sind, müßte er wahrscheinlich durch die ganze Welt reisen. Das World Wide Web bietet also Information unabhängig vom Ort an, wo sich diese befindet. Hierdurch steht einem Benutzer zu jedem Zeitpunkt eine Fülle von Daten zur Verfügung.

Mit diesem Vorteil ergeben sich aber auch Probleme. Die Datenmenge auf dem WWW macht es schwer, relevante Information aus der Datenflut zu extrahieren. Dies ist insbesondere dann der Fall, wenn der Benutzer seinen Informationsbedarf nicht genügend präzise formulieren kann - oder wenn der konkrete Informationsbedarf noch gar nicht klar definiert ist und sich erst im Laufe der Suche ergibt. Angenommen ein Benutzer betritt das World Wide Web der Carnegie Mellon University mit dem Interesse, es in Bezug auf "experimentelle Forschung über intelligente Agenten" zu explorieren. Diese Interessenbeschreibung ist sehr generell und bedarf der Verfeinerung. Was ist dem Benutzer lieber, Publikationen oder Projektinformationen? Gehört maschinelles Lernen für ihn auch dazu? Ist er an Projekten oder eher an den involvierten Personen interessiert? Ohne Wissen über den Inhalt der Dokumentkollektion wird es dem Besucher schwerfallen, die Beschreibung seines Interesses soweit zu operationalisieren, wie es für erfolgreiche Anfragen an ein konventionelles Text-Retrieval-System notwendig wäre. Hjerppe ([Hjerppe, 1986], Seite 221) formuliert dies als

> ... the need to describe that which you do not know in order to find it.

Bevor der Besucher herausgefunden hat, daß die Carnegie Mellon University z. B. Forschung im Bereich von autonomen Robotern durchführt, wird er wahrscheinlich "lernende Roboter" nicht bei der Beschreibung seines generellen Interesses an "intelligenten Agenten" angeben.

In der realen Welt wird explorative Suche dieser Art oft durch räumliche Strukturen unterstützt. In einer Bibliothek z. B. sind Bücher nicht zufällig verstreut, sondern ähnliche Bücher befinden sich meist im gleichen Regal. Gibt es auf dem WWW vergleichbare Strukturen? Obwohl das WWW von dem Ort abstrahiert, an dem Information gespeichert ist, existiert dennoch ein virtueller Raum, durch den sich der Benutzer bewegen kann. Er gibt dem WWW eine Struktur, die zur Exploration genutzt werden kann. Dieser hochdimensionale, virtuelle Raum ist durch die Hypertextstruktur gegeben. Wie in der realen Welt gibt es auf dem WWW also Orte und Verbindungen zwischen Orten.

Wenn man sich an einem fremden Ort zurechtfinden will, greift man gerne auf die Hilfe eines Fremdenführers (Tour Guides) zurück. Ähnlich wie ein Tour Guide in der realen Welt, ist WebWatcher [Armstrong et al., 1995] ein vereinfachter Tour Guide für das World Wide Web. WebWatcher begleitet den Benutzer auf Touren durch das WWW und unterstützt ihn mit seinem Wissen über die Domäne und mit seinem Wissen über die Interessen des Benutzers. Im Gegensatz zu anfragebasierten Text-Retrieval-Systemen auf dem WWW (wie z. B. Lycos oder Altavista) macht WebWatcher von der Hypertextstruktur Gebrauch. WebWatchers Touren orientieren sich an dieser Struktur, die wertvolle Information über die Relationen zwischen Dokumenten liefert.

Zu Beginn der Tour teilt der Benutzer WebWatcher sein Interesse mit - sozusagen das Thema der Tour. Diese Information benutzt das System während der Tour, um Vorschläge zu machen, welchen Hyperlinks auf der aktuellen Seite der Benutzer als nächstes folgen soll. WebWatcher bietet also eine auf das Suchinteresse des Benutzers abgestimmte Sicht auf Hypertextdokumente. Weiterhin kann der Benutzer mit WebWatcher kommunizieren und WebWatchers Performanz durch positive bzw. negative Rückmeldung kommentieren. WebWatcher ist ein sog. *learning apprentice* [Mitchell et al., 1985] [Mitchell et al., 1994], d. h. das System lernt durch Beobachtung der Aktionen von Benutzern. Auf diese Weise lernt WebWatcher über die Hypertextkollektion, auf der das System eingesetzt ist, und über die Interessen, die vorangegangene Benutzer hatten.

Motiviert durch WebWatcher wird in dieser Arbeit die Frage untersucht, inwieweit das von WebWatcher benötigte Wissen zum Geben von Touren automatisch mit Methoden des maschinellen Lernens gelernt werden kann. Eine automatische Aneignung dieses Wissens ist für den praktischen Einsatz notwendig, da durch die ständige Veränderung des WWW eine manuelle Wisseneingabe zu erheblichem Wartungsaufwand führen würde. In dieser Arbeit werden verschiedene Ansätze betrachtet, die es WebWatcher erlauben, sich das benötigte Wissen automatisch anzueignen. Zwei Informationsquellen, aus denen ein Tour Guide lernen kann, werden aufgezeigt: Die Hypertextstruktur der Domäne sowie die Touren, die WebWatcher bereits gegeben hat. Hieraus ergeben sich die folgenden zwei Fragen, die in dieser Arbeit beantwortet werden sollen:

Frage 1: Wie kann WebWatcher Wissen für die Tourenplanung durch Beobachtung von Benutzern lernen? Aus den bereits gegebenen Touren kann WebWatcher lernen, welche Hyperlinks auf der aktuellen Seite für das jeweilige Interesse des Benutzers relevant sind. Diese Hyperlinks können dann einem neuen Benutzer als vielversprechende Fortsetzungen der Tour vorgeschlagen werden. Es handelt sich bei diesem Problem um eine überwachte Lernaufgabe. Es wird gezeigt, daß dieses Problem als ein Klassifikationsproblem von Text modelliert werden kann. In Abschnitt 3 werden verschiedene Methoden zur Textkategorisierung vorgestellt, die teilweise bereits in ähnlicher Form bei anderen WWW-Agenten [Pazzani et al., 1996] [Balabanovic und Shoham, 1995] [Lieberman, 1995] verwandt wurden. Die Methoden werden experimentell verglichen.

Frage 2: Wie kann Wissen für die Tourenplanung aus Dokumenten und der Hypertextstruktur gelernt werden? Hierzu wird ein neuer Ansatz vorgestellt, der auf dem Prinzip des Reinforcement Learning (siehe z. B. [Barto et al., 1995]) beruht und bei dem von der Hypertextstruktur der Domäne gelernt wird. Im Gegensatz zu dem auf Textkategorisierung beruhenden Ansatz ist diese Methode nicht auf vom Benutzer generierte Trainingsbeispiele angewiesen. Dadurch ist sie gerade dann besonders geeignet, wenn solche Daten nicht zur Verfügung stehen, wie z. B. zu Beginn des Einsatzes. Reinforcement Learning ist eine Methode, um es z. B. einem Roboter [Lin, 1991] zu ermöglichen, Handlungssequenzen zur Lösung einer Aufgabe zu lernen. In der Domäne eines Tour Guides sind die zu lernenden Handlungssequenzen Touren, die aus Folgen von Hyperlinks bestehen. Die Struktur des WWW und der Inhalt von Dokumenten werden bei diesem Ansatz als Informationsquelle benutzt, um Navigationsinformation über Hyperlinks zu lernen. Die Idee bei diesem Ansatz ist es, eine Evaluationsfunktion zu lernen. Diese Evaluationsfunktion schätzt für jedes Hyperlink, wieviel relevante Information der Benutzer in Bezug auf sein Interesse finden wird, wenn er seine Tour mit dem betreffenden Hyperlink fortsetzt. Die Methode wird anhand der von WebWatcher gesammelten Touren evaluiert und mit dem Textkategorisierungsansatz verglichen.

2 WebWatcher

Durch WebWatcher [Armstrong et al., 1995] wird das in dieser Arbeit untersuchte Lernproblem motiviert. WebWatcher ist ein Tour Guide Agent, der Benutzer beim Browsing auf dem World Wide Web unterstützt. Das System begleitet Benutzer durch das WWW und gibt interaktiv Hilfe. Mit Methoden des maschinellen Lernens lernt WebWatcher das nötige Wissen zum Geben von Touren automatisch.

Im folgenden werden Aufbau und Arbeitsweise von WebWatcher anhand eines Szenarios beschrieben (Abschnitt 2.1). Der Einsatzbereich von WebWatcher wird definiert und mit anderen Systemen in Beziehung gesetzt (Abschnitt 2.2). Weiterhin wird das Problem des Lernens von Touren, das in dieser Arbeit behandelt wird, vorgestellt. Das Problem wird formal definiert (Abschnitt 2.4) und es wird die Methode vorgestellt, die WebWatcher momentan zum Lernen benutzt (Abschnitt 2.5). Diese Lernmethode wird als Basis für den Vergleich mit den in dieser Arbeit (Abschnitte 3 und 4) vorgestellten Ansätzen dienen. Abschnitt 2.6 präsentiert Ergebnisse und Statistiken von über 5000 Touren, die WebWatcher gegeben hat.

2.1 WebWatchers Funktionalität: Ein Szenario

Das folgende Beispiel soll WebWatchers Arbeitsweise anhand einer typischen Tour verdeutlichen. Wir betreten das World Wide Web der School of Computer Science (SCS) der Carnegie Mellon University auf der sog. Front-Door-Seite (Abbildung 1). Auf dieser Seite ist eine Instanz von WebWatcher installiert, die wir aufrufen, indem wir dem Hyperlink “The WebWatcher Tour Guide” im unteren Teil der Seite folgen. Wir gelangen zu der Seite “Welcome to WebWatcher”, die in Abbildung 2 dargestellt ist. Auf dieser Seite werden wir gebeten, eine kurze Beschreibung unseres derzeitigen Interesses einzugeben. In diesem Beispiel geben wir als Interessenbeschreibung “intelligent agents” ein und beginnen die Tour, indem wir auf den Knopf “get started” drücken. Die Tour startet auf der Seite, von der wir das System betreten haben. In unserem Falle ist das die oben erwähnte SCS-Front-Door-Seite. Allerdings bewegen wir uns jetzt nicht mehr allein durch das World Wide Web, sondern wir befinden uns in Begleitung von WebWatcher (Abbildung 3). Von nun an wird uns das System auf jeder Sequenz von Hyperlinks begleiten, der wir von dieser Seite aus folgen.

Daß WebWatcher uns begleitet, sehen wir an den Zusätzen, die das System zu der eigentlichen Seite hinzufügt. Die SCS-Front-Door-Seite mit WebWatchers Zusätzen ist in Abbildung 3 dargestellt. Unter anderem fügt WebWatcher die folgenden Elemente zu der ursprünglichen Seite hinzu:

- Eine Menüleiste wird oberhalb der ursprünglichen Seite eingefügt. Durch diese Menüleiste können wir durch Kommandos mit WebWatcher kommunizieren und positive oder negative Rückmeldung geben.

- Unterhalb der Menüleiste wird eine Liste von Hyperlinks eingefügt. Diese Hyperlinks führen zu Dokumenten, welche die Schlüsselworte “intelligent agents” enthalten, die wir zu Beginn der Tour eingegeben haben. Die Hyperlinks werden mit Hilfe eines Information-Retrieval-Systems ermittelt, welches auf dem Vektorraummodell basiert [Salton, 1991]. Das benutzte System ist eine Variante des Lycos Retrievalsystems, angewandt auf die Seiten, die Benutzer zuvor in Begleitung der WebWatcher-Instanz (hier die Instanz auf der SCS-Front-Door-Seite) besucht haben. Durch diese Beschränkung auf bereits besuchte Seiten wird die Suche fokussiert und auf die Expertise der WebWatcher-Instanz angepaßt.

- Ausgewählte Hyperlinks auf der Seite werden von WebWatcher hervorgehoben. Auf diese Weise werden Richtungen vorgeschlagen, die in Bezug auf unser Suchinteresse besonders relevant sind. WebWatcher hebt diese Hyperlinks hervor, indem es sie durch sein Logo () einrahmt. Ein solches hervorgehobenes Hyperlink ist zum Beispiel in der letzten Zeile von Abbildung 3 zu sehen. Die Hervorhebungen basieren auf Wissen, das WebWatcher auf vorangegangenen Touren gelernt hat. Die Methoden hierzu sind Gegenstand dieser Diplomarbeit und werden in den folgenden Kapiteln vorgestellt und analysiert.

In unserem Szenario folgen wir WebWatchers Vorschlag und klicken auf das Hyperlink “projects” in der letzten Zeile von Abbildung 3. Im allgemeinen kann ein Benutzer jedoch jederzeit WebWatchers Hinweise ignorieren und jedem beliebigen Hyperlink auf der Seite folgen. Jedesmal, wenn der Benutzer einem Hyperlink folgt, zeichnet WebWatcher die Bewegung auf und speichert sie als Trainingsbeispiel, um daraus für zukünftige Touren zu lernen.

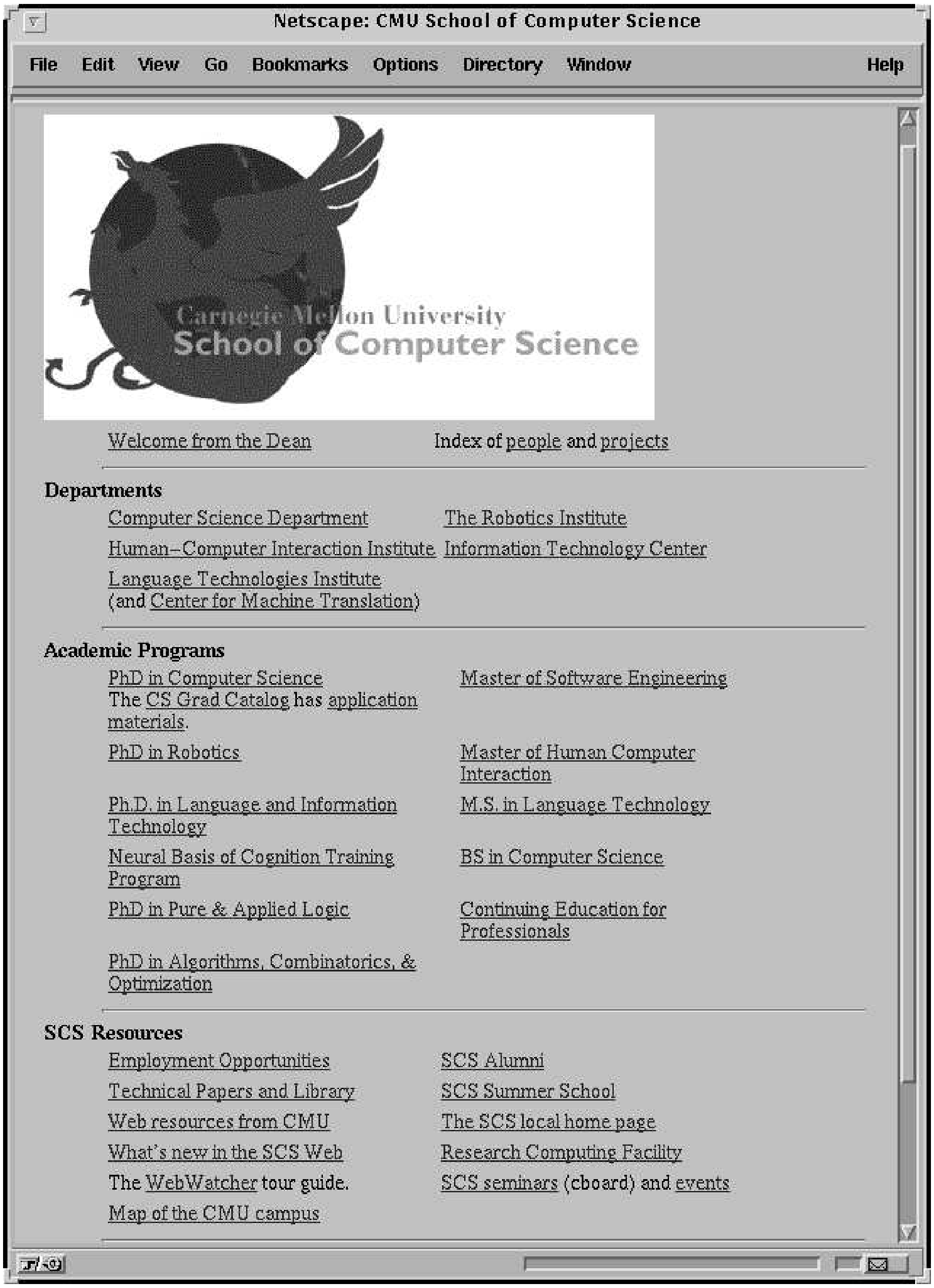

Abbildung 1: Die ursprüngliche SCS-Front-Door-Seite. Ein Hyperlink zu WebWatcher findet sich im unteren Teil der Seite.

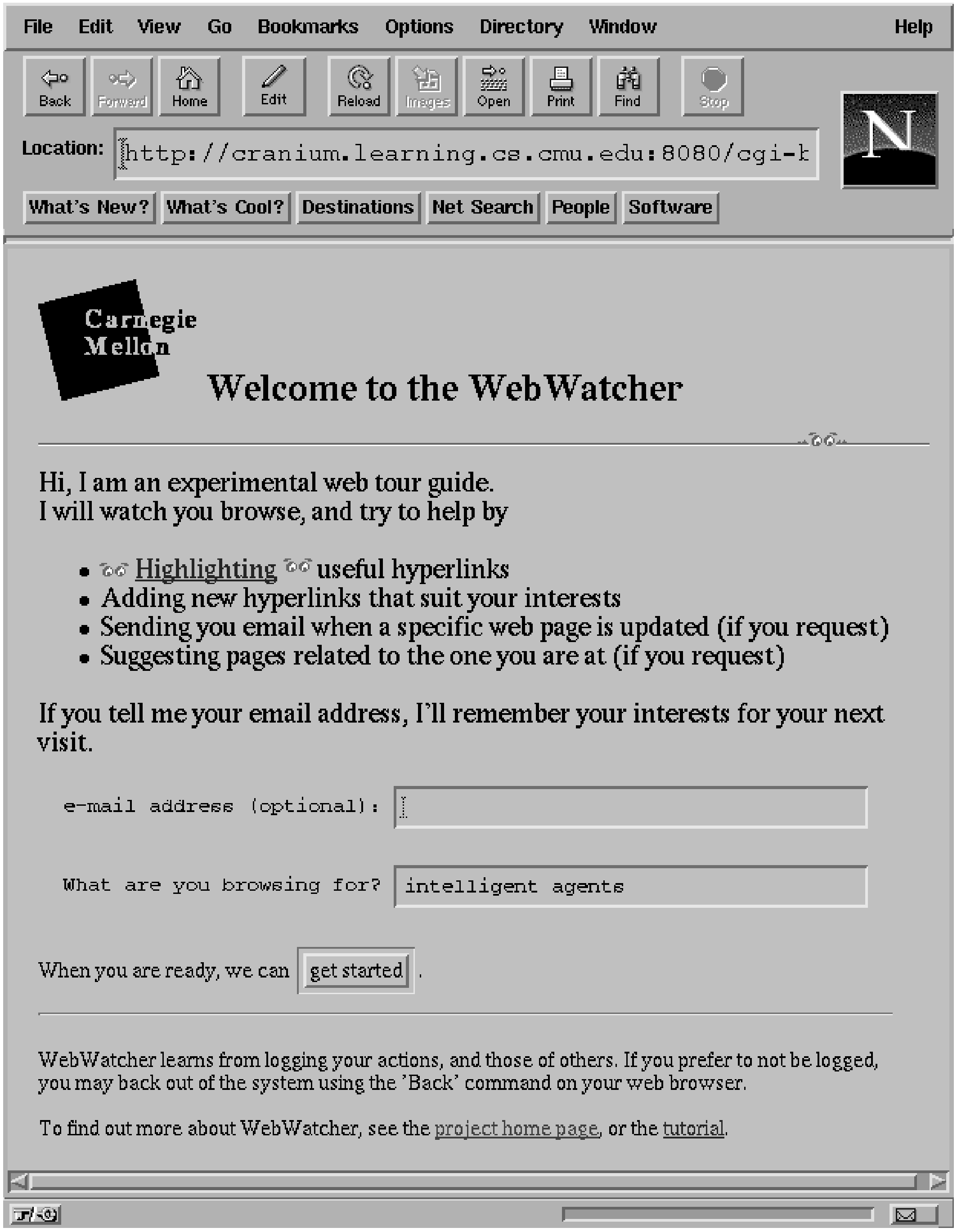

Abbildung 2: WebWatcher ist aufgerufen.

Abbildung 3: Die SCS-Front-Door-Seite mit WebWatchers Zusätzen.

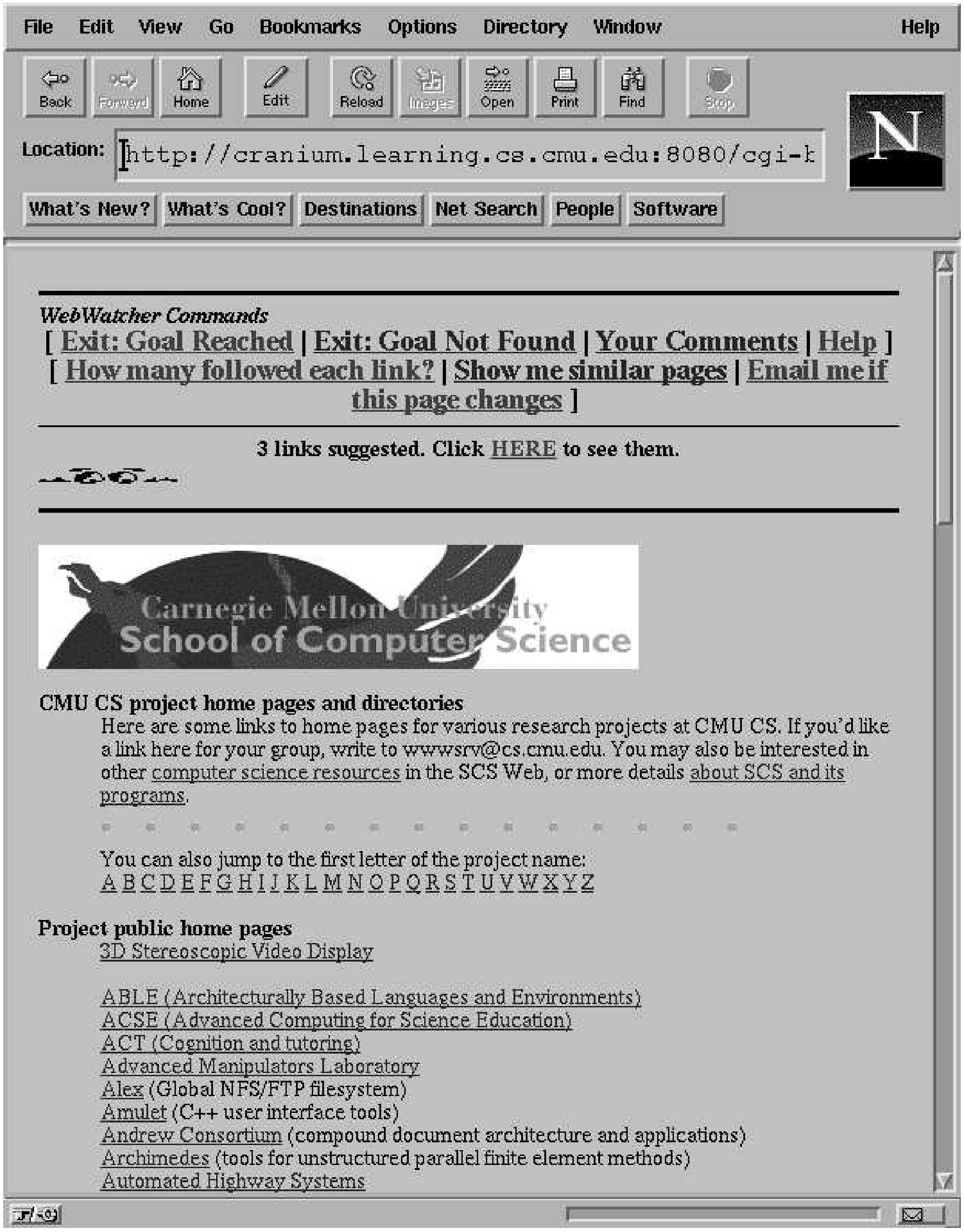

Abbildung 4: Eine Liste von Projekten an der CMU.

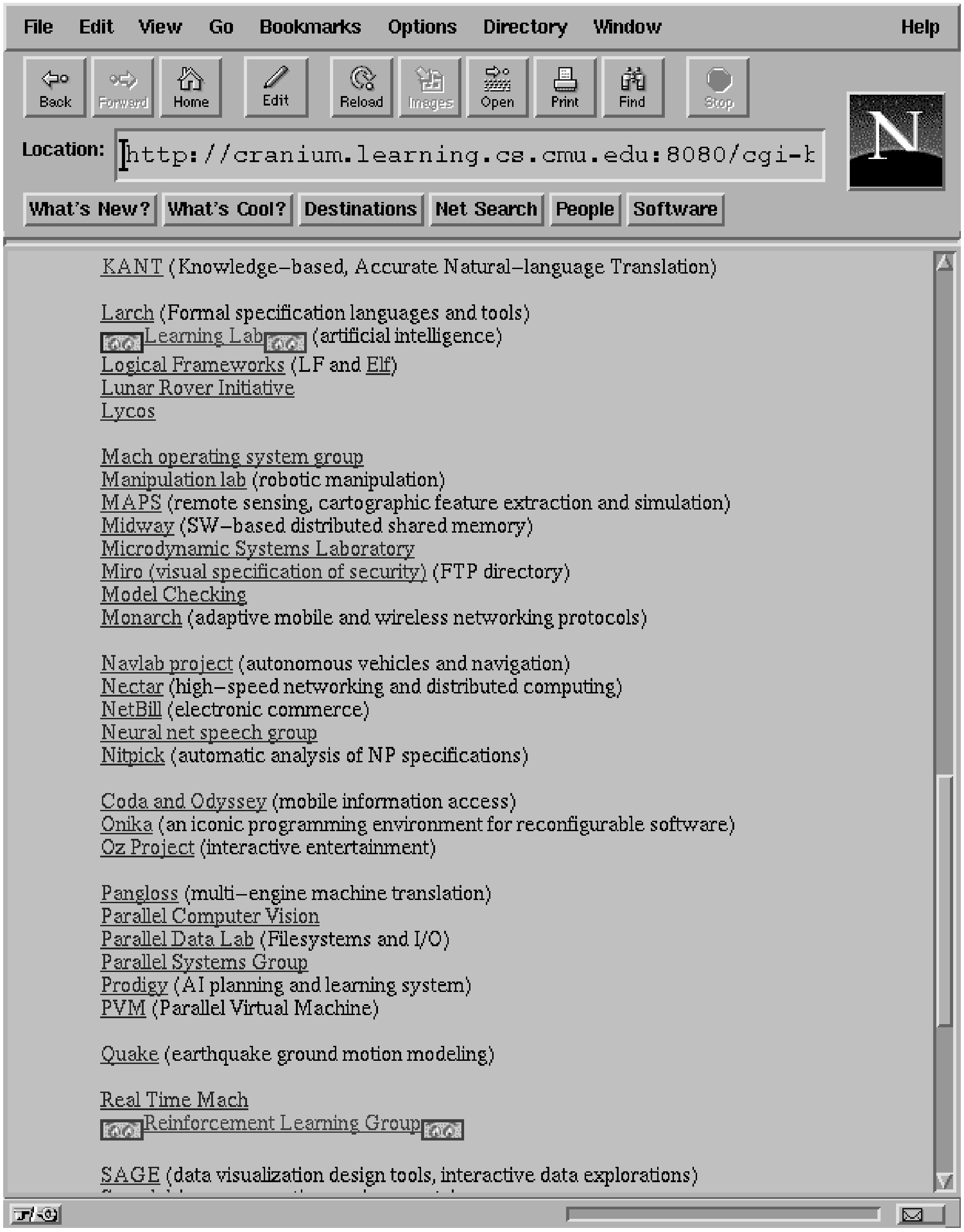

Abbildung 5: Weiter unten in der Liste von Projekten.

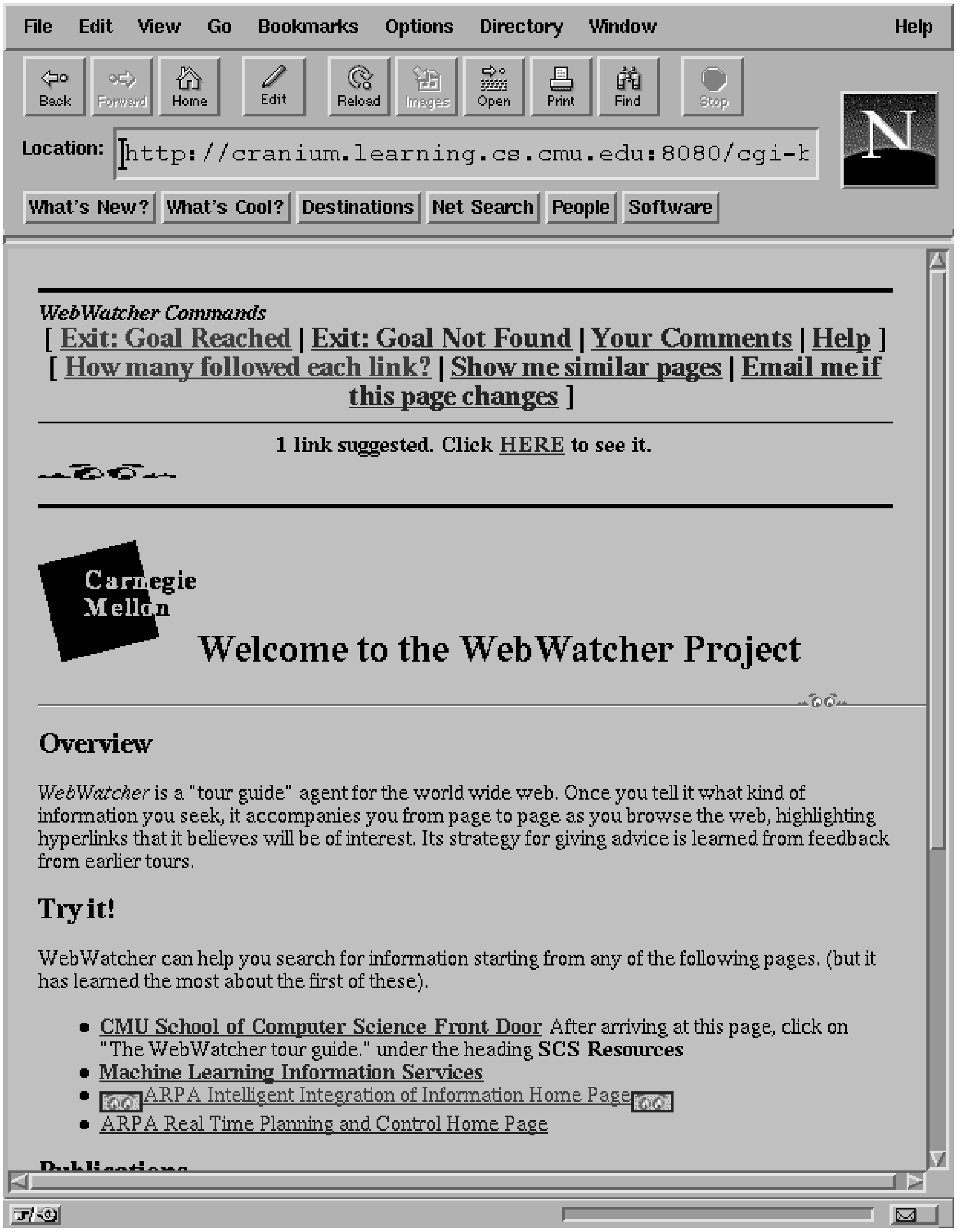

Abbildung 6: Der Benutzer kommt an der Home Page des WebWatcher Projektes an.

Die Seite, zu der wir nun gelangen, ist in Abbildung 4 dargestellt. Sie enthält eine lange Liste von Hyperlinks, die zu Home-Pages von Projekten an der Carnegie Mellon University führen. Wieder fügt WebWatcher die Menüleiste in die ursprüngliche Seite ein und zeigt uns so an, daß wir uns immer noch in Begleitung befinden. Auf dieser Seite hebt WebWatcher drei Hyperlinks hervor, die es für relevant in Bezug auf unser Interesse an "intelligent agents" hält. Indem wir auf das Hyperlink "HERE" im oberen Viertel der Seite (Abbildung 4) klicken, gelangen wir direkt zu WebWatchers erstem Hinweis (Abbildung 5). Wenn wir auf die Augen rechts oder links eines hervorgehobenen Hyperlinks klicken, gelangen wir zum jeweils nächsten bzw. vorangehenden Hinweis.

Auf unserer Tour entscheiden wir uns nun, WebWatchers drittem Hinweis (nicht in der Abbildung sichtbar) zu folgen, der uns zu der Seite führt, die Information über das WebWatcher Projekt enthält (Abbildung 6). Wieder hebt WebWatcher Hyperlinks hervor.

In dem Fall, daß eine Seite uns ganz besonders interessiert, können wir dieses WebWatcher mitteilen, indem wir die Funktion "Show me similar pages" in der Menüleiste auf der betreffenden Seite auswählen. WebWatcher zeigt uns dann eine Liste von Dokumenten, die ähnlich zu der aktuellen Seite sind. Das verwendete Ähnlichkeitsmaß macht von der Hypertext Struktur, in welche die aktuelle Seite eingebettet ist, Gebrauch. Im Gegensatz zu herkömmlichen Maßen geht der Text auf der Seite nicht in die Berechnung der Ähnlichkeit ein. Statt dessen werden die Hyperlinks betrachtet, die zu der Seite hinführen. Es wird die Annahme gemacht, daß jedes Hyperlink eine Rolle (oder Relation) darstellt, in der sich die Seite befindet. Zwei Dokumente sind ähnlich, wenn sie möglichst viele Rollen gemein haben. Eine genaue Beschreibung des Ähnlichkeitsmaßes findet sich in [Joachims et al., 1995].

Zwei weitere Kommandos können über die Menüleiste aufgerufen werden. Wenn wir auf das Hyperlink "how many followed each link?" klicken, fügt WebWatcher neben jedem Hyperlink auf der Seite ein, wie oft Benutzer in der Vergangenheit dem jeweiligen Hyperlink gefolgt sind.

Durch "email me if this page changes" kann die aktuelle Seite markiert werden. Alle markierten Seiten werden von WebWatcher in regelmäßigen Abständen überprüft. Wann immer sich eine der Seiten ändert, erhalten wir eine Email, die uns über die Änderungen informiert.

WebWatchers Mechanismus erlaubt es dem System prinzipiell, uns zu jeder beliebigen Seite auf dem World Wide Web zu begleiten. Um eine Tour zu beenden, können die zwei Kommandos "Exit: Goal reached" oder "Exit: Goal not found" in der Menüleiste benutzt werden. Durch diese Funktionen kann der Benutzer WebWatcher darüber Rückmeldung geben, ob relevante Information in Bezug auf sein Interesse gefunden wurde.

2.2 Was unterscheidet WebWatcher von anderen Information-Retrieval-Systemen auf dem WWW?

Um die Unterschiede zwischen WebWatcher und anderen Systemen verstehen zu können, ist es hilfreich, zuerst die verschiedenen Arten von Informationssuche zu strukturieren. Informationssuche ist ein allgemeiner Oberbegriff für eine Reihe von Zielen und Methoden. Meadow [Meadow, 1992] unterscheidet z. B. die folgenden vier Arten von Informationssuche, die jeweils eine unterschiedliche Aufgabenstellung darstellen und unterschiedliche Lösungsstrategien benötigen.

1. *known-item search*: Das gesuchte Objekt ist bekannt und kann vom Suchenden eindeutig beschrieben werden.

2. *specific-information search*: Der Benutzer hat einen konkret definierten Informationsbedarf (z. B. eine Frage), den er formulieren kann.

3. *general-information search*: Die Art des Informationsbedarfs ist eher generell und läßt sich nur schwer operational formulieren. Er ergibt sich zu einem großen Teil erst aus dem Inhalt der Dokumentkollektion.

4. *search to explore the database*: Ungerichtete Informationssuche, die sich nur durch die (möglicherweise latenten) Interessen des Benutzers definiert.

Ähnliche Kategorisierungen nimmt Bates [Bates, 1986] vor. Sie ordnet Suchstrategien in den Dimensionen aktiv/passiv und zielgerichtet/ungerichtet ein. Auch Ellis [Ellis, 1990], Lucarella und Zanzi [Lucarella und Zanzi, 1993] und Belkin et. al. [Belkin et al., 1993] kommen zu ähnlichen Charakterisierungen. Alle Ansätze haben den Aspekt gemeinsam, daß es Informationssuchen gibt, bei denen das Suchziel zu Beginn der Suche nicht genau formulierbar ist und sich erst durch den Suchprozeß genauer definiert.

Klassische Text-Retrieval-Systeme, insbesondere die zur Zeit auf dem WWW verfügbaren wie Lycos [Lycos, 1996], Altavista [Altavista, 1996] oder Webcrawler [Webcrawler, 1996] [Pinkerton, 1994], adressieren mehr die Probleme der known-item search und der specific-information search. Der Benutzer formuliert eine Anfrage und erhält eine Liste von potentiell relevanten Dokumenten.

An diesem Aufbau ändern auch erweiterte Text-Retrieval-Systeme nichts, welche die Relevanzschätzung nicht nur anhand des Dokumenttextes durchführen, sondern beim Retrieval aus Hypertext die durch Hyperlinks gegebene Information einbeziehen. Verschiedene Retrieval-Modelle wurden vorgeschlagen, die Hyperlinks in die Relevanzberechnung integrieren. Sie basieren z. B. auf Spreading Activation [Savoy, 1991] [Croft et al., 1989], sind durch Reinforcement Learning motiviert [Boyan et al., 1996] oder benutzen Bayes'sche Inferenz [Frisse und Cousins, 1989] [Lucarella und Zanzi, 1993] [Croft und Turtle, 1993]. Dunlop und Rijsbergen [Dunlop und van Rijsbergen, 1993] zeigen, wie Hyperlinks zur Indexierung von nicht textuellen Dokumenten verwendet werden können, was bei der multimedialen Natur des WWW nützlich sein kann. Trotzdem handelt es sich bei diesen Systemen in erster Linie um Werkzeuge, welche die Suche nach hinreichend genau beschreibbarer Information unterstützen.

Eine dem genau entgegengesetzte Position nehmen Systeme wie LIRA [Balabanovic und Shoham, 1995] und Syskill & Webert [Pazzani et al., 1996] ein. Diese Systeme unterstützen Punkt vier der obigen Auflistung, die "search to explore the database". Anhand eines generellen Interessenprofils, das erlernt oder vom Benutzer vorgegeben sein kann, durchsuchen diese Systeme automatisch das WWW nach potentiell für den Benutzer interessanten Dokumenten.

WebWatcher füllt die Lücke zwischen Systemen wie Lycos und Agenten wie LIRA. WebWatcher unterstützt den Benutzer bei Informationssuchen vom Typ "general-information search". Der Informationsbedarf ist nur teilweise explizit und schwer formulierbar. Er ergibt sich zu einem großen Teil aus dem Inhalt der Dokumentkollektion. Diese Art von Informationssuche fällt unter die Bezeichnung *"Browsing"*.

Hypertext eignet sich sehr gut für Informationssuche durch Browsing. Durch Hyperlinks können semantischen Relationen zwischen Dokumenten operational dargestellt werden. Durch das Folgen von Hyperlinks kann ein Benutzer auf sehr einfache Weise zu semantisch verwandten Dokumenten navigieren und eine Datenbank erforschen. Gerade unerfahrene Benutzer bevorzugen diese einfache Art der Informationssuche [Jacobson, 1991] [Campagnoni und Ehrlich, 1989], obwohl sie im allgemeinen nicht zu besseren Ergebnissen führt [Jacobson, 1991] [Wolfram et al., 1996].

Insbesondere bei großen Dokumentkollektionen ist es schwierig, allein durch Navigation in Hypertext zu seinem Suchziel zu gelangen. Oftmals sind hierzu eine Vielzahl von Schritten und Entscheidungen nötig. Dieser Nachteil kommt besonders zum Tragen, wenn bei genau beschreibbaren Suchzielen umständlich über Hyperlinks navigiert werden muß, obwohl eine Anfrage einfach und effizienter wäre. Deshalb wurden integrierte Systeme vorgeschlagen, die sowohl anfragebasierte Suche als auch das Browsing unterstützen [Belkin et al., 1993] [Frisse und Cousins, 1989] [Lucarella und Zanzi, 1993] [Dunlop und van Rijsbergen, 1993].

Verschiedene Methoden wurden vorgestellt, um den Benutzer beim Browsing zu unterstützen. Hierbei kommen vielfach graphische Darstellungen der Dokumentstruktur [Campagnoni und Ehrlich, 1989] und Fish-Eye Projektionen zum Einsatz [Furnas, 1986] [Nielsen, 1990]. Auch Metaphern aus der räumlichen Navigation werden betrachtet [Dieberger, 1996]. Andere Systeme bieten zusätzlich Thesauri an [Pollard, 1993] oder fügen zusätzliche Hyperlinks anhand einer Begriffshierarchie ein [Agosti et al., 1992]. Eine Begriffshierarchie für das WWW ist z. B. Yahoo [Yahoo!, 1996].

WebWatcher geht bei der Unterstützung des Browsing einen anderen Weg. Zusätzlich zu den bereits durch den Browser zur Verfügung gestellten Navigationsmethoden, die allerdings vielfach unzureichend und verwirrend sind [Jones und Cockburn, 1996], soll WebWatcher dem Benutzer beim Browsing aktiv assistieren. Entgegen einigen obigen Ansätzen soll der Hypertext hierzu nicht verändert werden. Die Idee ist es, basierend auf der vorgegebenen Hypertextstruktur, die Aufmerksamkeit des Benutzers auf potentiell relevante Teile der jeweiligen Seite zu fokussieren. Dies ist besonders bei unübersichtlichen und langen Hypertextdokumenten, wie sie auf dem WWW häufig vorkommen, sinnvoll. Es sei darauf hingewiesen, daß WebWatcher den Hypertext nicht grundlegend verändert. Dies hat den Vorteil, daß falsche Entscheidungen des Agenten den Benutzer nicht beschränken.

Konzeptionell ähnlich zu WebWatcher ist das System *Letizia* [Lieberman, 1995]. Wie WebWatcher begleitet das System den Benutzer beim Browsing auf dem WWW und gibt interaktiv Hilfe. Ein wesentlicher Unterschied zu WebWatcher ist, daß bei Letizia der Benutzer keine Interessenbeschreibung eingibt, sondern das Suchinteresse aus den Aktionen des Benutzers inferiert wird. Letizia benutzt darauf aufbauend eine Version des Rocchio-Algorithmus (siehe Abschnitt 3.4.1) und Breitensuche, um relevante Dokumente zu finden und dem Benutzer in einem zusätzlichen Fenster anzuzeigen.

2.3 Wie begleitet WebWatcher den Benutzer?

WebWatcher ist als Server auf einer eigenen Maschine mit Internet-Anschluß implementiert. Aus der Perspektive des Systems sieht das Szenario aus Abschnitt 2.1 folgendermaßen aus.

Wenn WebWatcher aufgerufen wird (in unserem Beispiel war dies auf der SCS-Front-Door-Seite in Abbildung 1), erhält das System den URL der aufrufenden Seite als Argument mittels der HTTP-GET Methode. Es ist die "Rückkehraddresse", auf der die Tour beginnen soll. Für jede aufrufende Seite besteht eine virtuelle Instanz von WebWatcher. Wenn WebWatcher auf einer neuen Seite installiert wird, wird beim ersten Aufruf automatisch eine neue Instanz erzeugt. Jede Instanz von WebWatcher hat einen separaten Datenbestand und wird nicht von anderen Instanzen beeinflußt. Dadurch können sich Instanzen auf WWW Lokalitäten (z. B. die Carnegie Mellon University, bei Installation auf der Front-Door-Seite) oder Themengebiete (z. B. maschinelles Lernen, bei Installation auf einer Seite über Ressourcen zum maschinellen Lernen) spezialisieren.

Nachdem der Benutzer sein Interesse eingegeben hat, beginnt die Tour auf einer modifizierten Version der Ausgangsseite. Folgende Änderungen werden durchgeführt:

1. Die Menüleiste wird vor die eigentliche Seite gesetzt.
2. Jedes Hyperlink in der Seite wird durch ein Hyperlink ersetzt, welches

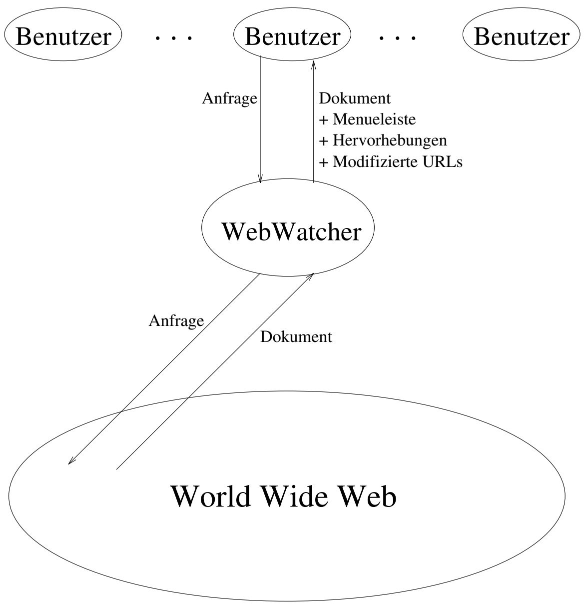

Abbildung 7: WebWatcher ist ein Interface Agent zwischen dem Benutzer und dem World Wide Web.

zurück zu WebWatcher zeigt.

3. Hyperlinks, die WebWatcher als relevant für das Interesse des Benutzers erachtet, werden durch Einfügen des Augensymbols hervorgehoben.

Nachdem die Substitutionen durchgeführt wurden, schickt WebWatcher die Seite an den Benutzer und vermerkt den Besuch der Seite in einem Protokoll der Tour. Das Protokoll enthält das vom Benutzer eingegebene Interesse, alle Hyperlinks und Seiten, die besucht wurden, und alle aus der Menüleiste aufgerufenen Aktionen. Jeder Eintrag in das Protokoll wird mit einem Zeitstempel versehen, um den zeitlichen Ablauf der Tour später nachvollziehen zu können. Während das System darauf wartet, daß der Benutzer den nächsten Schritt ausführt, werden die Seiten, auf welche die hervorgehobenen Hyperlinks zeigen, in eine Cache geladen, um Netzwartezeiten zu minimieren. Auf diese Weise ermöglicht WebWatcher dem Benutzer ein schnelleres Browsing. Wählt der Benutzer ein neues Hyperlink aus, fügt WebWatcher die Aktion dem Protokoll hinzu, holt die Seite vom WWW (falls nicht im Cache), führt die oben beschriebenen Ersetzungen durch und liefert die Seite an den Benutzer. Abbildung 7 faßt den Ablauf zusammen.

Dieser Prozeß wiederholt sich so lange, bis der Benutzer WebWatcher verläßt. WebWatcher vermerkt dann im Protokoll den Erfolg der Tour, abhängig davon, welchen der beiden Exit-Knöpfe der Benutzer gewählt hat. Auch wenn der Benutzer keinen der Exit-Knöpfe betätigt, was häufig vorkommt, wird dies vermerkt. Das Protokoll wird gespeichert und erlaubt es später, die Tour nachzuvollziehen. Der Benutzer wird vom System zu der Originalversion der Seite gebracht, auf der er WebWatcher verlassen hat. Von hier aus kann er sich alleine weiterbewegen oder eine neue Tour beginnen.

2.4 Welche Art von Wissen benötigt WebWatcher?

Im allgemeinen benötigt das System Wissen, um für einen Benutzer auf einer gegebenen Seite relevante Hyperlinks vorzuschlagen. Idealerweise wäre eine Funktion der Form

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

hierzu wünschenswert. Angenommen der Benutzer befindet sich auf einer Seite $p \in Page$. Auf dieser Seite stehen die Hyperlinks $Links(p)$ zur Verfügung, von denen eine Teilmenge als Fortsetzung der Tour vorgeschlagen werden soll. $2^{Links(p)}$ bezeichnet die Potenzmenge der Hyperlinks. Abhängig von der Interessenbeschreibung $i \in Interest$, die der Benutzer zu Beginn der Tour eingegeben hat, und Wissen über den Benutzer $u \in User$ selbst[1] bestimmt die Funktion die Hyperlinks $l \subseteq Links(p)$ zur optimalen Fortsetzung der Tour.

Die Funktion *OptimalLinks?* ist allerdings äußerst komplex. Zum einen bedarf der Begriff der "optimalen Tour" einer genaueren Definition. Zum anderen sind die Argumente der Funktion während einer Tour nicht notwendigerweise konstant. Insbesondere das Interesse des Benutzers wird sich ständig verändern. Es kann sich während der Tour iterativ entwickeln (z. B. spezialisieren) oder der Benutzer kann abgelenkt werden und plötzlich ein vollkommen neues Interesse verfolgen. In Abschnitt 4 wird gezeigt, wie *OptimalLinks?* unter vereinfachenden Annahmen direkt implementiert werden kann. Einige der Annahmen sind bereits durch das Design von WebWatcher vorgegeben. So ist es für den Benutzer lediglich zu Beginn der Tour möglich, sein Interesse mitzuteilen. Während der Tour wird das Interesse als konstant angenommen.

Zunächst wird im folgenden jedoch eine Approximation von *OptimalLinks?*, genannt *UserChoice?*, betrachtet, die für WebWatcher von großer praktischer Relevanz ist. Die Funktion *UserChoice?* läßt sich wie folgt formal darstellen:

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

Ihr Funktionswert sind die Hyperlinks, denen ein Benutzer mit einem Interesse $i \in Interest$ auf einer Seite $p \in Page$ wahrscheinlich folgt. Die Funktion versucht vorherzusagen, ob ein Benutzer einem Hyperlink folgen wird, nicht aber, ob das Hyperlink Teil einer optimalen Tour in Bezug auf das Benutzerinteresse ist. Die Aktionen von Benutzern sind nicht notwendigerweise optimal. Weiterhin ist zu beachten, daß der Benutzer keine explizite Eingabe zu der

[1] Wissen über den Benutzer können z. B. seine Hobbys oder sein Beruf sein. Aber auch das Wissen über die Seiten, die er bereits auf dem WWW besucht hat, ist hierzu zu zählen.

Funktion ist. Die Funktion $UserChoice?$ ist im allgemeinen nicht deterministisch, sondern lediglich eine probabilistische Zuordnung. Dies ergibt sich daraus, daß in unterschiedlichen Situationen Benutzer mit der gleichen Interessenbeschreibung auf derselben Seite unterschiedlichen Hyperlinks folgen können.

Ein wesentlicher Grund für die Entscheidung, die Experimente zunächst auf die Funktion $UserChoice?$ zu konzentrieren, ist, daß WebWatchers Tourprotokolle automatisch Trainingsbeispiele für diese Funktion liefern. In einem Tourprotokoll wird für jede Seite, die auf der Tour besucht wurde, gespeichert, welchen Hyperlinks der Benutzer gefolgt ist. Außerdem wird die Interessenbeschreibung, die der Benutzer zu Beginn der Tour eingegeben hat, festgehalten. Abhängig von der Länge der Tour liefert also jedes Tourprotokoll eine Vielzahl von Beispielen für die Funktion $UserChoice?$.

2.5 WebWatchers Lernmethode

WebWatchers derzeitige Lernmethode für $UserChoice?$ ist ein auf dem Vektorraummodell [Salton, 1991] basierender ad-hoc Lernalgorithmus. Er wird im weiteren Verlauf dieser Arbeit als Referenz für andere Lernalgorithmen dienen. Um den Lernalgorithmus anwenden zu können, muß die Funktion $UserChoice?$ aufgespalten werden. Für jede Seite p und jedes Hyperlink $l \in Links(p)$ auf der Seite p wird eine Funktion

$$UserChoice?_{p,l} : Interest \rightarrow \{+, -\}$$

gelernt. Sie entscheidet abhängig vom Interesse, ob ein Benutzer dem Hyperlink l auf Seite p folgen wird. Die hinter dieser Zerlegung stehenden Annahmen werden in Abschnitt 3.6 genauer analysiert.

Die heuristische Natur des Lernalgorithmus läßt eine formale Definition des Lernproblems nur bedingt zu. Für jede Seite p und jedes Hyperlink l auf der jeweiligen Seite p stehen aus den Tourprotokollen Trainingsbeispiele für $UserChoice?_{p,l}$ zur Verfügung. Ein positives Trainingsbeispiel ergibt sich daraus, daß der Benutzer während der Tour mindestens einmal dem Hyperlink l auf Seite p gefolgt ist. Negative Trainingsbeispiele können vom Algorithmus nicht verwendet werden. Beispiele werden durch die Multimenge der Worte (siehe auch Abschnitt 3.2.1) repräsentiert, die der Benutzer als Interessenbeschreibung für die betreffende Tour eingegeben hat. Das Lernziel ist es, die Funktionen $UserChoice?_{p,l}$ für alle Seiten p und Hyperlinks l einzeln zu approximieren. Die Hypothesensprache ist implizit durch den folgenden Algorithmus und das dort verwendete Ähnlichkeitsmaß gegeben.

WebWatcher lernt, indem die Beschreibungen von Hyperlinks durch die Interessenbeschreibungen der Touren angereichert werden, auf denen ein Benutzer diesen Hyperlinks gefolgt ist (positive Beispiele). Wann immer ein Benutzer einem Hyperlink auf einer Tour folgt, werden die Worte, die er zu Beginn eingegeben hat, zu der Beschreibung des Hyperlinks hinzugefügt. Die initiale

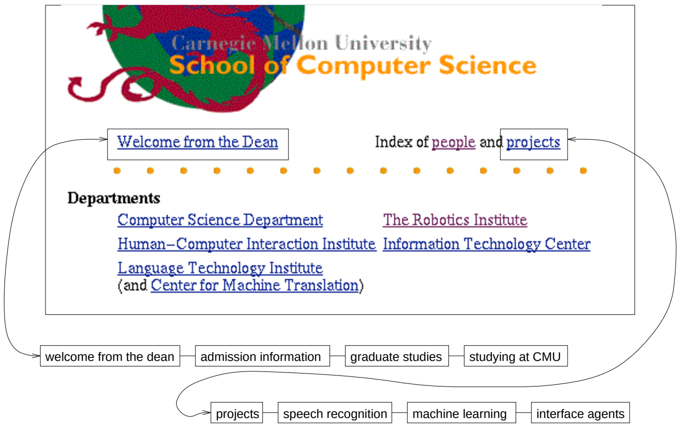

Abbildung 8: WebWatchers interne Beschreibung von Hyperlinks besteht aus dem Anchor-Text und den Interessenbeschreibungen von Benutzern, die diesem Hyperlink gefolgt sind. Wann immer ein Benutzer einem Hyperlink folgt, wird das von ihm eingegebene Interesse zu der Liste des Hyperlinks hinzugefügt.

Beschreibung ist der unterstrichene Anchor-Text[2] des Hyperlinks. Abbildung 8 gibt ein Beispiel dafür, wie Hyperlinks durch Listen von Interessenbeschreibungen repräsentiert sind. Anhang A zeigt die Beschreibung eines Hyperlinks, wie sie von WebWatcher gelernt wurde.

Um während einer Tour Hyperlinks vorzuschlagen, vergleicht WebWatcher die Interessenbeschreibung, die der Benutzer zu Beginn der Tour eingegeben hat, mit den Beschreibungen der Hyperlinks auf der aktuellen Seite. Hyperlinks, deren Beschreibung hinreichend ähnlich zu der Interessenbeschreibung ist, werden von WebWatcher dem Benutzer vorgeschlagen. Abbildung 9 gibt ein Beispiel.

Das Maß, welches WebWatcher benutzt, um die Ähnlichkeit einer Interessenbeschreibung und einer Hyperlinkbeschreibung zu bestimmen, basiert auf dem Vektorraummodell [Salton, 1991] aus dem Information Retrieval. Interessen und Hyperlinks werden durch hochdimensionale Vektoren repräsentiert. Jedes Wort der natürlichen Sprache entspricht einer Dimension. Kommt ein Wort in einer Beschreibung vor, enthält die Vektordarstellung der Beschreibung in der entsprechenden Dimension einen Wert größer als Null. Der genaue Wert wird nach der TFIDF-Heuristik berechnet, die in Abschnitt 3.4.1 genauer beschrieben wird. Abbildung 10 illustriert diese Art der Repräsentation von Text anhand eines Beispiels. Basierend auf dieser Vektordarstellung wird die Ähnlichkeit von zwei Beschreibungen anhand des Winkels zwischen den Vektoren gemessen. Je kleiner der Winkel, desto ähnlicher die Beschreibungen.

[2] Der "Anchor-Text" ist der Bereich, der sich zwischen zwei HTML Tags `<A>` und `</A>` befindet.

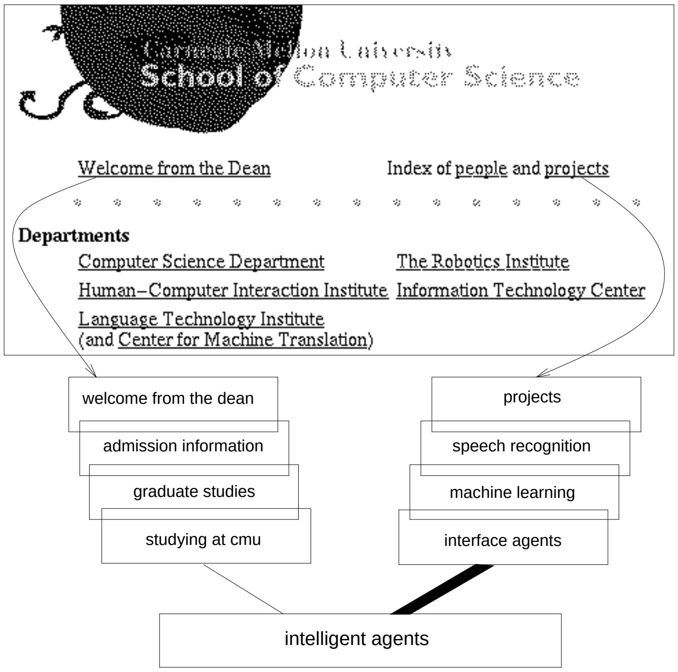

Abbildung 9: Hyperlinks werden vorgeschlagen, indem das Interesse mit der Beschreibung des Hyperlinks verglichen wird. Hier paßt das Benutzerinteresse "intelligent agents" besser zum Hyperlink "projects" als zum Hyperlink "Welcome from the Dean". Grund hierfür sind die Schlüsselworte von vorangegangenen Touren.

Der Algorithmus, den WebWatcher zum Vorschlagen von Hyperlinks verwendet, betrachtet alle Hyperlinks auf der aktuellen Seite nacheinander. Für jedes Hyperlink wird die Liste der assoziierten Interessen (einschließlich des unterstrichenen Anchor-Textes) anhand ihrer Ähnlichkeit zur Suchanfrage sortiert. Der Wert von $UserChoice?$ wird für jedes Hyperlink berechnet, indem die Ähnlichkeiten der k (normalerweise 5) obersten Beschreibungen gemittelt werden. Ein Hyperlink wird von WebWatcher vorgeschlagen, wenn der zugehörige Wert von $UserChoice?$ oberhalb eines Schwellwertes ist. Es werden maximal drei Hyperlinks pro Seite vorgeschlagen.

2.6 Ergebnisse und Statistiken

Die hier beschriebenen Resultate basieren auf 5822 Touren von der SCS-Front-Door-Seite, die WebWatcher zwischen dem 2. August 1995 und dem 4. März 1996 gegeben hat. Durchschnittlich sind Benutzer 2.8 Hyperlinks pro Tour gefolgt. 55% der Touren umfassen mindestens ein Hyperlink und 18% umfassen mindestens fünf Hyperlinks. Die längste Tour ist 135 Hyperlinks lang.

Jede Seite einer Tour, auf welcher der Benutzer mindestens einem Hyperlink von der Seite gefolgt ist, entspricht einem Beispiel der Funktion $UserChoice?$.

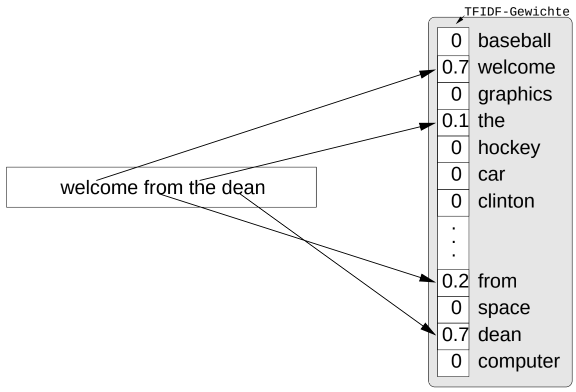

Abbildung 10: Textstücke werden als hochdimensionale Vektoren repräsentiert.

	Hyperlink hervorgehoben	Accuracy
Random	20.8%	15.3%
Learning	20.8%	43.9%

Tabelle 1: WebWatchers Performanz nach 5822 Touren.

Insgesamt umfaßt der Datensatz 6123 solcher Beispiele. Wie Tabelle 1 zeigt, hat WebWatcher während seiner Einsatzzeit mit dem oben beschriebenen Algorithmus für 20.8% der Beipiele mindestens ein Hyperlink hervorgehoben[3]. Auf Seiten mit Hervorhebungen ist der Benutzer bei 43.9% der Beispiele mindestens einem der von WebWatcher vorgeschlagenen Hyperlinks gefolgt. Diese Prozentzahl wird in Tabelle 1 als Accuracy bezeichnet.

Eine Untergrenze für die Accuracy liefert eine Zufallsstrategie *Random*. Betrachtet man die Schritte, auf denen WebWatcher einen Vorschlag gemacht hat, und hebt nachträglich durch zufällige Auswahl die gleiche Anzahl von Hyperlinks hervor, erhält man eine Accuracy von 15.3%. Es ist jedoch zu beachten, daß die Accuracy dadurch beeinflußt ist, daß WebWatcher die Entscheidungen des Benutzers durch seine Hervorhebungen beeinflußt.

[3] Auf 25% der Touren gaben die Benutzer keine Interessenbeschreibung ein. Auf solchen Touren konnte WebWatcher niemals Hinweise geben.

3 Textkategorisierung

Der in Abschnitt 2.5 vorgestellte Lernalgorithmus ist eine ad-hoc Lösung zur Implementation der Funktion $UserChoice?$. Gibt es fundiertere Methoden, die bessere Ergebnisse erzielen? Im folgenden wird auf die verschiedenen Lernmethoden zur Textkategorisierung eingegangen. Das Problem der Textkategorisierung wird zuerst formal definiert (Abschnitt 3.1). Bevor verschiedene Lernalgorithmen zur Textklassifikation in Abschnitt 3.4 vorgestellt werden, wird auf das essentielle Problem der Repräsentation von Text eingegangen (Abschnitte 3.2 und 3.3). Nach einer Übersicht über die für die Textkategorisierung geeigneten Performanzmaße (Abschnitt 3.5) wird dann gezeigt und evaluiert, wie Textkategorisierungsmethoden zur Implementation der Funktion $UserChoice?$ verwendet werden können (Abschnitte 3.6 bis 3.8).

3.1 Definition von Textkategorisierung

Gegenstand der Textkategorisierung ist es, Dokumente in eine vorgegebene Menge von Klassen einzuteilen. Somit ist Textkategorisierung eine spezielle Art von Klassifikationproblem. Jede Klasse stellt eine semantische Kategorie dar. Verschiedene Methoden wurden angewandt, um den Prozeß der Zuweisung von bislang nicht kategorisierten Dokumenten zu einer Klasse zu automatisieren. Hayes und Weinstein [Hayes und Weinstein, 1990] benutzen z. B. einen regelbasierten Ansatz mit manuell erstellten Regeln, die von bestimmten Phrasen im Dokument auf dessen Klasse schließen.

Oftmals ist es allerdings unbekannt, nach welchem Konzept die Dokumente eingeteilt werden. Oder das allgemeine Konzept ist bekannt, aber es ist schwierig, es zu operationalisieren und in einer Programmiersprache darzustellen. Deshalb wurden vielfältige Methoden vorgestellt, die es erlauben, das Konzept, nach dem Dokumente klassifiziert werden sollen, anhand von Trainingsbeispielen zu lernen. Diese Aufgabe fällt in den Bereich des überwachten Lernens, wie es in der Mustererkennung und dem maschinellen Lernen betrachtet wird.

Die Definition von Textkategorisierung, die in dieser Arbeit benutzt wird, ist die folgende. Die Anzahl der Klassen ist bekannt und konstant. Jedes Dokument wird genau einer Klasse zugewiesen. Formal betrachtet gibt es eine Menge von Klassen $\mathcal{C}$ und eine Menge von Trainingsbeispielen D. Weiterhin existiert eine Funktion $T(d) \in \mathcal{C}$, die jedem Dokument d eine Klasse zuweist. Für die Dokumente in der Trainingsmenge ist $T(d)$ gegeben. Mit Hilfe der Information aus den Trainingsbeispielen versucht der Lernalgorithmus, eine Hypothese H zu induzieren, die T approximiert. $H(d)$ kann benutzt werden, um ungesehene Beispiele zu klassifizieren. Es wird versucht, die Hypothese zu finden, welche die höchste Genauigkeit erzielt und $T(d)$ am besten approximiert.

In Abschnitt 3.6 wird gezeigt, wie die Definition von T dahingehend erweitert werden kann, daß Dokumente in mehreren Klassen enthalten sein können.

Dies bedeutet, daß $T(d) \subseteq \mathcal{C}$ zu einer Relation wird. Es wird gezeigt, daß unter gewissen Annahmen dieser Fall von der bisherigen Definition subsumiert wird.

3.2 Repräsentation

Die Repräsentation von Beispielen und Hypothesen hat einen entscheidenden Einfluß auf die Generalisationsfähigkeit von Lernalgorithmen. Die Hypothesensprache - die vom Lernalgorithmus lernbaren Hypothesen - werden zum einen durch den Algorithmus bestimmt, zum anderen aber auch ganz wesentlich von der Repräsentation der Beispiele beeinflußt. Um Text für maschinelles Lernen handhabbar zu machen, muß er also von einer Kette von Zeichen in eine Repräsentation transformiert werden, die sich sowohl für den Lernalgorithmus als auch für die Klassifikationsaufgabe eignet.

Ein grundlegendes Problem bei dem Umgang mit natürlicher Sprache ist, daß man das Gesagte immer in einem breiten Kontext betrachten muß (siehe [Blair, 1992]). Das gleiche Wort in zwei unterschiedlichen Sätzen kann unterschiedliche Bedeutung besitzen. Der gleiche Satz kann unterschiedliche Bedeutung haben, abhängig davon, wer ihn spricht, für wen er bestimmt ist und in welcher Situation er geäußert wird. Lewis Carroll [Caroll, 1972] formuliert dies auf humorvolle Art und Weise:

> "When I use a word," Humpty Dumpty said in a rather scornful tone, "it means just what I choose it to mean–neither more nor less."

Schwierigkeiten wie diese werden in unterschiedlichem Maß von den im folgenden vorgestellten Ansätzen zur Textrepräsentation berücksichtigt oder ignoriert. Die Ansätze lassen sich anhand der ersten vier der folgenden fünf Kategorien einordnen. Die fünfte Ebene sei nur der Vollständigkeit halber erwähnt.

1. Morphologische Ebene: Der Aufbau von Worten.
2. Lexikalische Ebene: Aussagen über ganze Worte.
3. Syntaktische Ebene: Die Struktur von Sätzen.
4. Semantische Ebene: Die Bedeutung von Textstücken.
5. Pragmatische Ebene: Die Bedeutung von Text in Bezug auf sprachlichen und außersprachlichen Kontext (z. B. Dialoggestaltung).

Strukturierungen anhand dieser Kategorien haben sich in der Linguistik bewährt. Allerdings kann keine der Ebenen für sich alleine betrachtet werden, da sich Mehrdeutigkeiten ergeben, die jeweils nur unter Zuhilfenahme anderer Ebenen aufgelöst werden können. Betrachtet man z. B. das Wort "book" allein auf der lexikalischen Ebene, kann nicht entschieden werden, ob es sich um ein Verb oder Nomen handelt. Auf der syntaktischen Ebene läßt sich hingegen eine

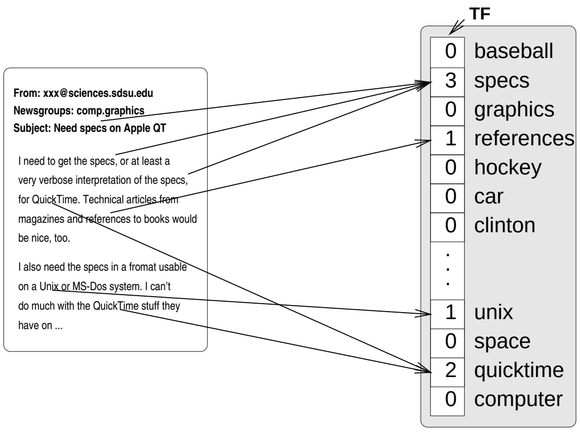

Abbildung 11: Texte werden als Vektoren repräsentiert. Jede Dimension entspricht einem Wort.

eindeutige Entscheidung fällen ("Please book that flight!").

Im folgenden werden Repräsentationsformalismen für die Englische Sprache auf morphologischer, lexikalischer, syntaktischer und semantischer Ebene vorgestellt. Je höher die Ebene, desto ausdrucksstärker ist das Modell. Mit der Ausdruckskraft steigt allerdings auch die Komplexität des Modells und es wird schwieriger, statistische Inferenzen zu machen. Als erstes und am ausführlichsten werden wortbasierte Repräsentationen behandelt. Dies ist die am häufigsten verwendete Repräsentationsform, die auch den in dieser Arbeit beschriebenen Experimenten zugrunde liegt.

3.2.1 Lexikalische Ebene

Die Aufteilung eines Textes in Worte liegt nahe. Worte bilden in vielen Fällen Bedeutungseinheiten, deren Semantik auch ohne Kontextinformation relativ eingeschränkt ist. Weiterhin ist die Definition davon, was ein Wort ausmacht, relativ eindeutig und einfache Verfahren sind in der Lage, einen Text effizient in seine Worte zu zerlegen[4]. In der Tat haben sich wortbasierte Repräsentationsformalismen in der Praxis bewährt (siehe z. B. [Lewis, 1992a]) und zählen zu den meistverwendeten.

Die Betrachtung von Worten als Indexierungsterme transformiert die Repräsentation von Text als Zeichenketten in Sequenzen von Worten. Zusätzlich wird meist die vereinfachende Annahme gemacht, daß die Reihenfolge der Worte vernachlässigt werden kann. Dokumente werden also nicht mehr als Sequenzen von Worten dargestellt, sondern als Multimengen ("bags") von Worten. Diese Repräsentationsform wird deshalb häufig als *Bag-of-Words-Ansatz* bezeichnet.

[4] Dies gilt insbesondere für die englische Sprache. Im Türkischen hingegen können ganze Sätze durch Kompositabildung gebildet werden, was den wortbasierten Ansatz verkompliziert.

Die Bag-of-Words-Repräsentation ist konsistent mit der im maschinellen Lernen benutzten Attribut-Wert-Darstellung von Beispielen. Jedes unterschiedliche Wort ist ein Attribut. Der Wert eines Attributs für ein Dokument ist die Anzahl der Vorkommen des entsprechenden Wortes. Diese Anzahl wird als *Term Frequency* $TF(w,d)$ des Wortes w in Dokument d bezeichnet. Abbildung 11 illustriert an einem Beispiel, wie Dokumente in der Bag-of-Words-Darstellung durch einen Attributvektor repräsentiert werden können.

Die Repräsentation von Dokumenten als Multimengen ist eine gebräuchliche Technik im Bereich des Information Retrieval. Es ist zu beachten, daß durch die oben beschriebenen Transformationen von Text in diese Repräsentation Information über das Dokument verloren geht. Trotz erheblichen Einsatzes ist es bis jetzt nicht gelungen, durch ausgefeiltere Repräsentationsformalismen konsistent wesentlich bessere Ergebnisse zu erzielen. Für das Problem des Information Retrievals formuliert Lewis [Lewis, 1992b] das *Equal Effektiveness Paradox*:

> ... all reasonable text representations have been found to result in very similar effectiveness on the retrieval task.

Lewis [Lewis, 1992a] führt dies darauf zurück, daß komplexere Repräsentationsformalismen zwar in der Lage sind, die Semantik von Text besser zu erfassen. Allerdings verschlechtert die höhere Komplexität die statistischen Eigenschaften des Modells. Die Bag-of-Words-Repräsentation scheint einen guten Mittelweg zwischen Modellkomplexität und Ausdrucksfähigkeit zu gehen. Deshalb, und wegen ihrer einfachen Handhabbarkeit, wird sie in den hier beschriebenen Experimenten verwendet.

3.2.2 Morphologische Ebene

Der bekannteste morphologische Repräsentationsformalismus ist der der Trigramme (oder allgemeiner n-Gramme). Indexierungsterme sind nicht die im Text vorkommenden Worte, wie im vorangegangenen Abschnitt, sondern alle vorkommenden Zeichenketten der Länge drei (oder n). Das Wort *“book”* wird z. B. auf die Multimenge *“_bo”, “boo”, “ook”, “ok_”* abgebildet. Die Trigram-Repräsentation hat die Vorteile, daß sie robust gegenüber Schreibfehlern ist und Ähnlichkeiten zwischen Wörtern modelliert.

3.2.3 Syntaktische Ebene

Fortschritte bei der Entwicklung von Parsern für natürliche Sprache haben es ermöglicht, große Mengen von Text effizient auf ihre syntaktische Struktur zu untersuchen. Die Idee von syntaktischen Repräsentationen ist es, Indexierungsterme nicht nur aus einzelnen, sondern aus mehreren Worten bestehen zu lassen, die eine bestimmte syntaktische Funktion im Satz haben. Nominalphrasen wurden hierbei am intensivsten untersucht [Fagan, 1987] [Lewis, 1992b] [Lewis, 1992a]. Dieser Ansatz wird als *Syntactic Phrase Indexing* bezeichnet.

Ein weiterer Ansatz zur Bildung von Mehr-Wort-Termen beruht auf statistischen Methoden. Die Satzstruktur wird nicht betrachtet. Statt dessen werden Terme gebildet, indem Statistiken darüber angefertigt werden, welche Worte oft zusammen auftreten [Fagan, 1987]. In [Fuhr et al., 1991] wird ein einfaches syntaktisches Verfahren benutzt, um Kandidaten für Mehr-Wort-Terme zu generieren, aus denen anschließend geeignete Phrasen durch eine statistische Methode herausgefiltert werden.

3.2.4 Semantische Ebene

Information Retrieval und Textklassifikationssysteme können nur dann optimal arbeiten, wenn sie die Semantik von Dokumenten erfassen können. Bei dem heutigen Stand der Forschung ist es allerdings noch nicht möglich, die Semantik von freiem Text automatisch vollständig zu erfassen und zu repräsentieren.

Eingeschränkte semantische Aussagen über Dokumente lassen hierarchische *Beschreibungssprachen mit fixiertem Vokabular* zu. Beispiele hierfür sind die MeSH im Bereich der Medizin, die Dewey Decimal Klassifikation oder Yahoo [Yahoo!, 1996] für das World Wide Web. Dokumente werden manuell in eine hierarchische Struktur eingeordnet. Steigt man in der Hierarchie aufwärts, gelangt man zu semantisch übergeordneten Konzepten. Eine solche manuelle Einordnung ist für beliebige Dokumente in der Regel aber natürlich nicht gegeben. Eine Methode zur Automatisierung des Einordnungsprozesses wird in [Yang und Chute, 1993] vorgeschlagen.

In Abschnitt 3.3.2 werden zwei Methoden vorgestellt, die ausgehend von einer Bag-of-Words-Repräsentation automatisch semantische Kategorien bilden. *Term Clustering* (siehe z. B. [Lewis, 1992a]) versucht, durch statistische Untersuchungen semantisch ähnliche Terme zu identifizieren und zusammenzufassen. Ähnlich argumentieren Deerwester et al. [Deerwester et al., 1990], daß *Latent Semantic Indexing* (siehe auch Abschnitt 3.3.2) semantische Kategorien durch Transformation in eine kompaktere Repräsentation findet.

Große Operationalität bieten Dokumentationssprachen, die auf Prädikatenlogik oder semantischen Netzen [Brachman und Schmolze, 1985] basieren. Allerdings erfordern sie eine manuelle Übersetzung des Textes in die Beschreibungssprache.

3.3 Attributselektion

Das Problem der Attributselektion[5] ist eng mit den im vorangegangenen Abschnitt betrachteten Problemen verbunden. Im maschinellen Lernen ist man zu der Erkenntnis gelangt, daß es für viele Probleme von Vorteil ist, nicht alle zur Verfügung stehenden Attribute beim Lernen zu berücksichtigen. Die Zusammenstellung eines "guten" Attributsatzes, der die Performanz eines Lernsystems optimiert, wird als *Attributselektion* bezeichnet. Ein guter Attributsatz zeichnet sich dadurch aus, daß er zum einen genug Information zur Lösung des Lernproblems bereitstellt, zum anderen aber keine irrelevanten Attribute enthält.

Attributselektion wird durch eine Funktion

$$AS : Problem \times LearnAlg \times AttributeSet \rightarrow NewAttributeSet$$

realisiert. Abhängig vom Lernproblem $Problem$ und dem verwendeten Lernalgorithmus $LearnAlg$ bildet sie von der Menge der verfügbaren Attribute auf eine neue Attributmenge ab. Legt man die im vorangegangenen Abschnitt beschriebene Repräsentation von Dokumenten als Multimengen von Worten zugrunde[6], entspricht die Menge der Attribute, die potentiell zur Beschreibung von Dokumenten dienen können, der Menge der Worte in der (englischen) Sprache. Man unterscheidet zwei Arten von Attributselektionsfunktionen AS:

- AS bildet die Menge der verfügbaren Attribute auf eine Teilmenge hiervon ab, d. h. $AttributeSet \supseteq NewAttributeSet$.
- AS führt auch neue Attribute ein.

Methoden für beide Arten von Attributselektion werden in den folgenden zwei Abschnitten vorgestellt.

3.3.1 Selektion einer Teilmenge von Attributen

Eine nur auf Text anwendbare Methode zur Identifikation von irrelevanten Attributen ist die *Stoppwort-Elimination.* Hierbei wird von der Annahme ausgegangen, daß Worte wie z. B. "the" oder "and" unabhängig vom Klassifikationsproblem nicht relevant sind. Sie werden deshalb aus der Attributmenge entfernt.

Während bei der Stoppwort-Elimination i. d. R. besonders häufig vorkommende Worte aus der Attributmenge gelöscht werden, entfernt man beim *Document Frequency Thresholding* [Yang und Pedersen, 1996] gerade die besonders selten vorkommenden Worte. Auch diese Methode ist speziell auf die Eigenschaften von Text zugeschnitten. Nur die Worte werden als Attribute selektiert, die in mindestens n Dokumenten der Kollektion vorkommen. Durch

[5] Die im folgenden vorgestellten Methoden beschränken sich auf attributive Beschreibungssprachen, da sie bei der Textkategorisierung hauptsächlich eingesetzt werden. Eine Ausnahme ist [Cohen, 1995a], wo das prädikatenlogische Äquivalent zur Attributselektion - Relationenselektion - beschrieben wird.

[6] Viele der im folgenden aufgeführten Methoden lassen sich aber auch auf andere Arten von Indexierungstermen anwenden, z. B. Trigramme oder Phrasen.

geeignete Wahl von n kann die Anzahl der selektierten Attribute gesteuert werden. Einige Annahmen, die hinter dieser Auswahlmethode stehen, sind, daß nur selten vorkommende Worte entweder nicht informativ sind oder durch ihr seltenes Auftreten die Performanz des Lernsystems nur unwesentlich beeinflussen [Yang und Pedersen, 1996]. Apté und Damerau [Apté und Damerau, 1994] rechtfertigen diese Selektionsmethode dadurch, daß für selten vorkommende Worte keine verläßlichen Statistiken erstellt werden können und sie somit keinen Beitrag beim Lernen liefern können.

Das Auswahlkriterium *Mutual Information* aus der Informationstheorie [Shannon und Weaver, 1949] ist eine Methode, die im maschinellen Lernen auch in anderen Domänen vielfach angewendet wird[7]. Mutual Information $I(T,W)$ [Cover und Thomas, 1991] mißt die Entropiereduktion, die sich daraus ergibt, daß zwei Zufallsvariablen T und W nicht getrennt, sondern abhängig voneinander betrachtet werden.

$$\begin{aligned} I(T,W) &= E(T) - E(T|W) & (1) \\ &= -\sum_{C\in\mathcal{C}} \Pr(T(d)=C)\cdot\log\Pr(T(d)=C) \\ &\quad + \sum_{C\in\mathcal{C}} \Pr(T(d)=C, W=0)\cdot\log\Pr(T(d)=C|W=0) \\ &\quad + \sum_{C\in\mathcal{C}} \Pr(T(d)=C, W=1)\cdot\log\Pr(T(d)=C|W=1) & (2) \end{aligned}$$

In unserer Domäne beschreibt die Zufallsvariable T die Klasse, in der sich ein Dokument befindet. Die Zufallsvariable W beschreibt, ob ein bestimmtes Wort w in dem Dokument vorkommt. Die Entropie $E(X)$ ist ein Maß für die Unsicherheit einer Zufallsvariable X, indem sie die erwartete Anzahl der Bits angibt, die ein optimaler Code zur Beschreibung eines Zuges von der Zufallsvariable benötigt [Cover und Thomas, 1991]. $I(T,W)$ ist also ein Maß für die Informationsmenge, die das Vorkommen eines Wortes W unabhängig von anderen Worten in Bezug auf die Klasse eines Dokumentes liefert. Die Worte mit den höchsten $I(T,W)$ werden selektiert. Anwendungen von Mutual Information auf Attributselektion für Text finden sich z. B. in [Yang und Pedersen, 1996] [Lewis und Ringuette, 1994] [van Rijsbergen, 1977] [Lewis, 1992a]. Im Gegensatz zu den beiden oben vorgestellten Methoden berücksichtigt Mutual Information die Klassifikationsaufgabe bei der Attributauswahl.

χ^2 Statistiken (siehe [Kreyszig, 1975]) sind eine weitere aufgabenbezogene Methode zur Attributauswahl. Für jedes Wort w wird untersucht, inwiefern seine Vorkommenswahrscheinlichkeit in einem Dokument von der Klassifikation des Dokuments abhängt. Anwendungen finden sich in [Yang und Pedersen, 1996] [Schütze et al., 1995].

[7] z. B. basiert das Verfahren ID3 [Quinlan, 1986] zum Lernen von Entscheidungsbäumen auf der iterativen Anwendung dieses Maßes.

Schließlich sei noch das Auswahlkriterium *Word Strength* erwähnt [Wilbur und Sirotkin, 1992]. Die Klassifikationsaufgabe bleibt bei dieser Methode unberücksichtigt. Ein interessanter Aspekt dieses Maßes ist, daß im Gegensatz zu den oben beschriebenen Methoden zusätzliche Information in die Attributauswahl einfließen kann. Von einem Menschen werden Paare von gegenseitig relevanten Dokumenten gekennzeichnet. Diese Relevanzrelation wird benutzt, um die Wahrscheinlichkeit

$$\Pr(Wort\ w\ kommt\ in\ x\ vor \mid Wort\ w\ kommt\ in\ y\ vor) \tag{3}$$

zu definieren, wobei x und y zwei als gegenseitig relevant gekennzeichnete Dokumente sind. Die Wahrscheinlichkeit wird als "Word Strength" $s(w)$ des Wortes w definiert. Worte, deren $s(w)$ signifikant über dem Erwartungswert bei angenommener stochastischer Unabhängigkeit von der Relevanzrelation liegen, werden selektiert. Ursprünglich als Methode zur automatischen Identifikation von Stoppworten im Information Retrieval gedacht, zeigen Yang und Wilbur [Yang und Wilbur, 1996], daß es auch im Bereich der Textkategorisierung erfolgreich eingesetzt werden kann. Es wurde eine vereinfachte Version des Verfahrens benutzt, die nicht auf von Menschen erstellten Relevanzurteilen beruht, sondern Relevanz mittels Kosinusähnlichkeit (siehe 3.4.1) bestimmt. Weiterhin wurde auf den Signifikanztest zur Wortselektion verzichtet. Worte wurden lediglich anhand der Höhe von $s(w)$ selektiert.

Verfahren zur Attributselektion, die auf Suche und Cross Validation beruhen (siehe z. B. [Caruana und Freitag, 1994]), stellen vielversprechende Methoden zur Verfügung, wurden aber im Bereich der Textkategorisierung noch nicht eingesetzt.

3.3.2 Attributselektion und Veränderung der Repräsentation

Bei den in diesem Abschnitt beschriebenen Verfahren wird eine Reduktion der Dimensionalität dadurch erreicht, daß neue Attribute die Beispiele "effizienter" beschreiben. Die neuen Attribute sollen so beschaffen sein, daß möglichst wenig relevante Information über die Beispiele verloren geht, aber dennoch die Anzahl der Attribute minimiert wird. Die Verfahren Stamm/Grundformreduktion, Benutzung von Thesauri, Latent Semantic Indexing und Term Clustering werden im folgenden vorgestellt.

Die *Stammformreduktion* beruht auf der morphologischen Analyse von Wörtern. Es wird die Annahme gemacht, daß die Unterscheidung von verschiedenen Derivations- und Flexionsformen eines Wortes bei der Textkategorisierung von nur untergeordneter Bedeutung ist. Deshalb werden Worte auf ihren Stamm reduziert. Dies führt dazu, daß Worte wie "computing" "computability" und "computer" auf das gleiche Attribut "comput" abgebildet werden. Ähnlich zur Stammformreduktion ist die *Grundformreduktion*. Worte werden nicht auf ihren Stamm, sondern auf ihre Grundform abgebildet. Bei Verben ist dies der Infinitiv, bei Substantiven der Nominativ Singular. In dieser Arbeit wird der in

[Porter, 1980] beschriebene Algorithmus zur Stammformreduktion verwendet.

Während Stamm- und Grundformreduktion von der Flexionsform eines Wortes abstrahieren, sollen *Thesauri* Worte nach semantischen Kategorien zusammenfassen. In einem Thesaurus sind Relationen zwischen Worten gespeichert. So werden z. B. Synonyme in Äquivalenzklassen zusammengefaßt. Oftmals sind in Thesauri auch Relationen wie spezieller-als/genereller-als oder Assoziationsrelationen enthalten.

Latent Semantic Indexing (LSI) [Deerwester et al., 1990] ist die Anwendung von Singular Value Decomposition (SVD) auf die Repräsentation von Text. SVD führt eine spezielle lineare Transformation der Trainingsbeispiele durch. Die Beispiele werden in einer Matrix A zusammengefaßt, deren Zeilen den Attributen und deren Spalten den Dokumenten entsprechen. Die Elemente von A sind die Vorkommenshäufigkeiten des jeweiligen Attributs in einem Dokument. SVD wird dazu benutzt, eine lineare Transformation T zu bestimmen, die A auf eine niederdimensionale Matrix A' abbildet. Dabei soll möglichst wenig Information über A verloren gehen, d. h. es soll eine weitere lineare Funktion D^T existieren, die A' mit nur geringen Fehlern wieder zurück in A überführt. Die Anzahl der Dimensionen der Matrix A', in die T den ursprünglichen Attributraum abbildet, ist ein Parameter. Durch die Transformation werden Redundanzen des ursprünglichen Attributraumes reduziert. Man hofft, daß hierdurch von der Wortwahl in Dokumenten abstrahiert wird und die Dimensionen des neuen Vektorraumes von mehr semantischer Natur sind. Anwendungen von Latent Semantic Indexing finden sich in [Yang und Pedersen, 1996] [Foltz, 1990] [Berry et al., 1995] [Foltz und Dumais, 1992]. Ähnliche Ansätze finden sich im maschinellen Lernen z. B. im Bereich der auto-encoding neuronalen Netze (siehe [Mitchell, 1997]). Bartel et al. [Bartell et al., 1992] zeigen die Beziehung zwischen LSI und Analyseverfahren aus dem Bereich des Multidimensional Scaling. Ein Nachteil des LSI ist, daß das betrachtete Klassifikationsproblem nicht in die Berechnung von T mit eingeht. Diesbezügliche Erweiterungen werden in [Schütze et al., 1995] [Dumais, 1994] [Yang, 1995] vorgeschlagen.

Ein ähnlicher Ansatz wie beim LSI wird beim *Term Clustering* verfolgt. Semantisch ähnliche Terme sollen in Clustern zusammengefaßt werden, um die Dimensionalität zu reduzieren. Stamm- und Grundformreduktion sind sehr einfache Methoden zum Zusammenfassen von unterschiedlichen Wörtern zu Clustern. Die Cluster werden zu den neuen Attributen des Dokuments. Bei komplexeren Verfahren werden Cluster mit Methoden der Clusteranalyse (siehe z. B. [Lebowitz, 1987] [Cheeseman et al., 1988]) gebildet. Diese Verfahren benötigen Beschreibungen von Termen durch Meta-Attribute[8]. Die gebräuchlichste Art und Weise, diese Meta-Attribute für Terme zu generieren, ist, deren Vorkommenshäufigkeit in jedem Dokument der Kollektion als Meta-Attribut aufzufassen (siehe z. B. [Spark-Jones, 1973]). In einer Kollektion von 1000 Doku-

[8] Die Bezeichnung Meta-Attribute wurde gewählt, um Begriffsverwirrungen zu vermeiden. Terme selbst sind ja Attribute in Bezug auf das Klassifikationsproblem.

menten hat also jeder Term 1000 Meta-Attribute. Ähnliche Ansätze werden in [Croft und Lewis, 1990] [Lewis, 1992a] und [Crouch, 1988] verfolgt. Clusteranalyseverfahren gruppieren Terme zusammen, deren Meta-Attribute anhand eines Ähnlichkeitsmaßes eng zusammengehören. Man hofft, daß durch geeignete Wahl der Meta-Attribute und des Ähnlichkeitsmaßes semantisch verwandte Terme in den gleichen Clustern zusammengruppiert werden.

3.4 Lernalgorithmen

Neben der Repräsentation der Beispiele hat natürlich auch der verwendete Lernalgorithmus und der von ihm induzierte Bias einen wesentlichen Einfluß auf die Performanz eines lernenden Systems. Im folgenden wird eine Übersicht über die zur Textklassifikation verwendeten Lernalgorithmen gegeben. Besondere Beachtung finden hierbei der Rocchio-Algorithmus, der naive Bayes'sche Klassifikator, die k-Nearest-Neighbor-Regel und der Winnow-Algorithmus. Sie werden genauer dargestellt und motiviert, da dies die Algorithmen sind, die bei den in dieser Arbeit beschriebenen Experimenten zum Einsatz kommen.

3.4.1 Der Rocchio-Algorithmus

Der Rocchio-Algorithmus [Rocchio, 1971] wurde als Methode zum Relevance Feedback im Rahmen des Information-Retrieval-Systems SMART [Salton, 1971] entwickelt. Der Algorithmus basiert auf dem Vektorraum-Retrieval-Modell [Salton, 1991]. In diesem Modell werden Dokumente ebenso wie Anfragen nach dem im vorangegangenen Abschnitt 3.2.1 beschriebenen Verfahren als Wortvektoren[9] repräsentiert. Weiterhin wird ein Abstandsmaß definiert, welches die (semantische) Ähnlichkeit dieser Vektoren messen soll. Wird eine Anfrage an das Retrieval-System gestellt, werden die Dokumente aus der Datenbank anhand ihrer so definierten Ähnlichkeit zur Anfrage sortiert. Es wird davon ausgegangen, daß Dokumente um so wahrscheinlicher relevant in Bezug auf die Anfrage sind, je ähnlicher sie zu ihr sind.

Wie genau werden Dokumente und Anfragen als Vektoren repräsentiert und welches Ähnlichkeitsmaß wird benutzt? Ausgehend von der Repräsentation durch den Vektor der Term Frequencies TF, wie in Abschnitt 3.2.1 eingeführt, werden Worte noch anhand eines weiteren Faktors, der sogenannten *Inverse Document Frequency* $IDF(w)$, gewichtet. Die Inverse Document Frequency berechnet sich als

$$IDF(w) = \log \frac{n}{DF(w)} \tag{4}$$

n ist die Gesamtanzahl der betrachteten Dokumente und $DF(w)$ ist die Anzahl der Dokumente hiervon, die das Wort w mindestens einmal enthalten.

Die Kombination von TF und IDF wird *TFIDF-Gewichtung* genannt. Diese Gewichtung sagt aus, daß ein Wort um so aussagekräftiger für den Inhalt eines

[9] Aber auch Phrasen etc. sind einsetzbar.

Dokumentes oder einer Anfrage ist, je häufiger dieses Wort in dem Dokument vorkommt (TF-Teil). Andererseits wird angenommen, daß Worte weniger aussagekräftig sind, wenn sie in vielen Dokumenten vorkommen (IDF-Teil). Die Repräsentation $\vec{d}$ eines Dokuments oder einer Anfrage im Vektorraum-Retrieval-Modell ergibt sich also daraus, daß der Vektor der Term Frequencies mit dem IDF-Vektor elementweise multipliziert wird.

$$\vec{d} = \overrightarrow{TF} * \overrightarrow{IDF} \tag{5}$$

Eine Übersicht über andere Gewichtungsformeln findet sich in [Salton und Buckley, 1987].

Als Maß für die Ähnlichkeit von Dokumenten wird der Kosinus der zugehörigen Vektoren benutzt. Der Kosinus mißt den Winkel zwischen Dokumentvektoren und kann wie folgt berechnet werden.

$$cos(\angle(\vec{d_1}, \vec{d_2})) = \frac{\vec{d_1} \cdot \vec{d_2}}{||\vec{d_1}|| \cdot ||\vec{d_2}||} = \frac{\vec{d_1} \cdot \vec{d_2}}{\sqrt{\vec{d_1} \cdot \vec{d_1}} \cdot \sqrt{\vec{d_2} \cdot \vec{d_2}}} \tag{6}$$

$||\vec{d}||$ bezeichnet die Länge eines Vektors d. Dieses Ähnlichkeitsmaß ist von der Länge des Dokuments und des zugehörigen Dokumentvektors unabhängig. Andere Ähnlichkeitsmaße werden z. B. in [Wang et al., 1992] vorgeschlagen.

Relevance Feedback bezeichnet eine Form von mehrstufigem Retrievalprozeß. Nachdem das System eine Rangordnung von Dokumenten in Bezug auf die Anfrage $\vec{q}$ dem Benutzer präsentiert hat, kann dieser eine Auswahl von Dokumenten als relevant bzw. nicht relevant kennzeichnen. Diese Kennzeichnungen werden dann dazu benutzt, eine verfeinerte Anfrage automatisch zu generieren. Der Rocchio-Algorithmus schreibt vor, daß die neue Anfrage $q\prime$ anhand der folgenden Formel erzeugt werden soll. Die Formel besagt, daß die Vektoren der relevanten Dokumente zu der ursprünglichen Anfrage addiert und die der nicht relevanten Dokumente subtrahiert werden sollen. Es läßt sich zeigen, daß ein so berechnetes $q\prime$ relevante und irrelevante Dokumente unter bestimmten Bedingungen optimal trennt [Rocchio, 1971].

$$\vec{q\prime} = \vec{q} + \alpha \frac{1}{|D_+|} \sum_{\vec{d} \in D_+} \vec{d} - \beta \frac{1}{|D_-|} \sum_{\vec{d} \in D_-} \vec{d} \tag{7}$$

D_+ ist die Menge der vom Benutzer als relevant gekennzeichneten Dokumente, D_- enthält die nicht relevanten Dokumente. $|D_+|$ und $|D_-|$ bezeichnen ihre Kardinalitäten. α und β sind Parameter. Elemente von $\vec{q\prime}$, deren Wert negativ ist, werden auf Null gesetzt [Rocchio, 1971].

Der Rocchio-Algorithmus läßt sich auf natürliche Art und Weise zur Textkategorisierung adaptieren. Voraussetzung ist, daß es sich um ein binäres Klassifikationsproblem handelt (o. B. d. A. mit den Klassen + und $-$). D_+ sind nun nicht die vom Benutzer als relevant gekennzeichneten Dokumente, sondern die Trainingsdokumente, die sich in der Klasse + befinden. D_- sind die in der anderen Klasse. Eine initiale Anfrage $\vec{q}$ existiert bei der Textkategorisierung nicht.

$$\vec{m} = \alpha \frac{1}{|D_+|} \sum_{\vec{d} \in D_+} \vec{d} - \beta \frac{1}{|D_-|} \sum_{\vec{d} \in D_-} \vec{d} \tag{8}$$

Auch hier werden Elemente von $\vec{m}$, deren Wert negativ ist, auf Null gesetzt. Der so berechnete Vektor $\vec{m}$ kann zur Klassifikation von neuen Dokumenten benutzt werden. Wann immer die Ähnlichkeit eines neu zu klassifizierenden Dokumentes d mit $\vec{m}$ größer ist als ein Schwellwert Θ, wird dieses Dokument als zugehörig zu + klassifiziert. Ansonsten wird es als − klassifiziert. Die Höhe von $\cos(\vec{m}, \vec{d})$ kann als Konfidenzwert verwendet werden.

$$H_{ROCCHIO}(d) = \begin{cases} + & wenn \cos(\vec{m}, \vec{d}) > \Theta \\ \\ - & sonst \end{cases} \tag{9}$$

Experimente, bei denen diese oder eine ähnliche Form des Rocchio-Algorithmus zur Textkategorisierung verwendet wird, finden sich z. B. in [Schütze et al., 1995] [Balabanovic und Shoham, 1995] [Lang, 1995] [de Kroon et al., 1996] [Cohen, 1996].

3.4.2 Naiver Bayes'scher Klassifikator

Der Klassifikator, der in diesem Abschnitt vorgestellt wird, basiert auf einer Modellierung von Text mit wahrscheinlichkeitstheoretischen Methoden. Es wird davon ausgegangen, daß Texte von verschiedenen Wahrscheinlichkeitsverteilungen erzeugt werden. Diese Verteilungen sind sehr komplex und lassen sich direkt nicht handhaben. Deshalb werden vereinfachende Annahmen gemacht, die es ermöglichen, die Verteilungen auf einem Computer darzustellen und aus Trainingsdaten zu schätzen. Die vereinfachenden Annahmen sollten so gewählt werden, daß die Komplexität gerade genug reduziert wird, um das Problem handhabbar zu machen, aber dennoch die für die Klassifikationsaufgabe relevanten Eigenschaften der Verteilung erhalten bleiben.

Bei dem hier beschriebenen sog. naiven Bayes'schen Klassifikator wird die Annahme gemacht, daß Worte unabhängig sind gegeben die Klasse, aus der das Beispiel stammt[10]. Auf Text angewandt läßt sich das wie folgt interpretieren. Dokumente sind Sequenzen von Worten. Die Sequenzen werden von einer Verteilung erzeugt, die von der Klasse C abhängt, in welcher sich das Dokument befindet. Das jeweils nächste Wort der Sequenz wird durch Ziehen von der Verteilung ermittelt. Die vereinfachende Annahme, die hierbei gemacht wird, ist, daß das jeweils nächste Wort nicht von den anderen im Dokument befindlichen

[10]Genaugenommen ist diese Annahme stärker als nötig und die schwächere Annahme der "linked-dependence" reicht aus [Cooper, 1991]. Die folgende Herleitung würde aber wesentlich unübersichtlicher und weniger intuitiv. Beide Annahmen führen zu der gleichen Entscheidungsregel.

Dokument (d):	*Several*	*methods*	*for*	*text*	*categorization*	*have*	*...*	
Kategorie1 (C1):	0.012 *	0.007 *	0.023 *	0.035 *	0.021 *	0.040 *	... =	$\hat{P}(d\|C1)$
Kategorie2 (C2):	0.011 *	0.001 *	0.026 *	0.006 *	0.001 *	0.042 *	... =	$\hat{P}(d\|C2)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Abbildung 12: Es wird die Wahrscheinlichkeit berechnet, daß der Text *"Several methods for text categorization have..."* in Kategorie C_1 (bzw. C_2) vorkommt. Durch die Unabhängigkeitsannahme kann die Wahrscheinlichkeit, ein Dokument d in einer Kategorie C_i zu beobachten, durch ein Produkt bestimmt werden. Die Faktoren des Produkts sind die Auftrittswahrscheinlichkeiten der einzelnen Wörter.

Worten abhängt. Diese Annahme wird "Konditionale Unabhängigkeitsannahme" genannt. Sie trifft für Text nachweisbar nicht zu[11]. Dennoch ist sie als "erste Approximation" sinnvoll, um das Problem der Textkategorisierung überhaupt handhabbar zu machen.

Wie benutzt der hier vorgestellte Klassifikator diese Modellierung von Text durch Wahrscheinlichkeitsverteilungen? Es wird versucht die Wahrscheinlichkeit $\Pr(C|d)$ zu ermitteln, daß Dokument d sich in Klasse C befindet. Die Bayes'sche Regel [James, 1985] besagt, daß ein Klassifikator optimal ist, wenn er ein Dokument d der Klasse zuweist, für die $\Pr(C|d)$ am höchsten ist.

$$H_{BAYES}(d) = \operatorname*{argmax}_{C \in \mathcal{C}} \Pr(C|d) \tag{10}$$

argmax $f(x)$ liefert den Wert x, für den $f(x)$ maximal wird. Bayes' Theorem [James, 1985] kann dazu benutzt werden, die Wahrscheinlichkeit $\Pr(C|d)$ in zwei Teile aufzuspalten.

$$\Pr(C|d) = \frac{\Pr(d|C) \cdot \Pr(C)}{\sum_{C \in \mathcal{C}} \Pr(d|C) \cdot \Pr(C)} \tag{11}$$

$\Pr(C)$ ist die a priori Wahrscheinlichkeit, das ein Dokument in Klasse C ist. $\Pr(d|C)$ ist die Wahrscheinlichkeit, daß in Klasse C Dokument d beobachtet wird.

$\hat{\Pr}(C)$, der Schätzwert für $Pr(C)$, kann von dem Anteil der Trainingsbeispiele berechnet werden, die sich in der entsprechenden Klasse befinden.

$$\hat{\Pr}(C) = \frac{|C|}{\sum_{C' \in \mathcal{C}} |C'|} \tag{12}$$

[11] Das gleiche gilt auch für die schwächere Annahme der "linked-dependence".

$|C|$ ist die Anzahl der Dokumente in der Klasse.

Das Schätzen der Wahrscheinlichkeit $\Pr(d|C)$ ist weniger einfach. $\Pr(d|C)$ ist die Wahrscheinlichkeit, ein Dokument d in Klasse C zu beobachten. Abbildung 12 gibt ein Beispiel. Hier wird versucht die Wahrscheinlichkeit zu schätzen, daß in einem Dokument der Klasse C_1 (bzw. C_2) der Satz "Several methods for text categorization have.." vorkommt. Für eine Kategorie C_1 mit Dokumenten über Information Retrieval ist diese Wahrscheinlichkeit höher als für eine Kategorie C_2 mit Kochrezepten.

Das Schätzen der Wahrscheinlichkeit $\Pr(d|C)$ ist allerdings nur mit starken Vereinfachungen möglich. Da es eine riesige Menge von unterschiedlichen Dokumenten gibt, ist es nicht möglich, durch einfaches Beobachten von Vorkommenshäufigkeiten diese Wahrscheinlichkeit zu bestimmen. Es ist sehr unwahrscheinlich, daß das genau gleiche Dokument überhaupt ein zweites Mal in der Trainingsmenge vorkommt. Einen Schätzwert zu bestimmen, wird erst durch die vereinfachende Annahme der konditionalen Unabhängigkeit möglich. Macht man diese Annahme, so kann $\Pr(d|C)$ aus den Vorkommenswahrscheinlichkeiten der einzelnen Worte berechnet werden (siehe Abbildung 12).

$$\Pr(d|C) = \prod_{i=1}^{|d|} \Pr(w_i|C) \tag{13}$$

Der Index i läuft über die Sequenz der Worte in Dokument d, wobei w_i das i-te Wort der Sequenz bezeichnet. Im Gegensatz zu anderen probabilistischen Klassifikatoren (siehe z. B. [Lewis, 1992b] [Maron, 1961]) benutzt der hier vorgestellte nicht nur das Vorkommen von Worten als binäre Attribute, sondern zieht auch die Vorkommenshäufigkeit von Worten in Betracht. Außerdem bietet die hier gewählte Art der Modellierung eine elegante Möglichkeit, das Nicht-Vorkommen von Worten zu ignorieren, was bei anderen Modellierungen erst durch explizite Zusatzannahmen (z. B. [Maron, 1961] [Cohen, 1995a]) erreicht wird. Durch die Unabhängigkeitsannahme reduziert sich das Schätzen von $\Pr(d|C)$ auf das Schätzen der einzelnen $\Pr(w_i|C)$. Ein Bayes'scher Schätzer wird hierzu benutzt.

$$\hat{\Pr}(w_i|C) = \frac{1 + TF(w_i, C)}{|F| + \sum_{w' \in |F|} TF(w', C)} \tag{14}$$

F ist die Menge der Attribute. Dieser Schätzer, der oft auch Laplace-Schätzer genannt wird, wird in [Vapnik, 1982] (Seiten 54-55) beschrieben. Er benutzt die Annahme, daß die Beobachtung aller Worte a priori gleich wahrscheinlich ist. Dieser Schätzer hat sich in der Praxis als günstig erwiesen, da er entgegen dem Maximum Likelihood Schätzer (siehe z. B. [Larson, 1982]) nicht fälschlicherweise Wahrscheinlichkeiten als Null schätzt.

Die resultierende Klassifikationsregel ergibt sich, wenn die Gleichungen (10),

(11) und (13) kombiniert werden.

$$H_{BAYES}(d) \quad = \quad \operatorname*{argmax}_{C \in \mathcal{C}} \frac{\Pr(C) \cdot \prod_{i=1}^{|d|} \Pr(w_i|C)}{\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \prod_{i=1}^{|d|} \Pr(w_i|C')} \tag{15}$$

Auch dieser Algorithmus basiert auf der Vektorrepräsentation, wie sie in Abschnitt 3.2 eingeführt wurde. Wie Gleichung 15 zeigt, ist die Kenntnis der Term Frequencies für die Anwendung des Bayes'schen Klassifikators ausreichend. $\Pr(C|d)$ kann als Konfidenzwert für die Klassifikation benutzt werden. Experimente mit Bayes'schen Klassifikatoren werden u. a. in [Lewis, 1992b] [Lewis und Ringuette, 1994] [Joachims, 1996] [Lang, 1995] [Pazzani et al., 1996] durchgeführt.

3.4.3 k-Nearest-Neighbor

Bei dem k-Nearest-Neighbor-Verfahren (k-NN) handelt es sich um einen weiteren Klassifikationsalgorithmus, der in der Statistik entwickelt wurde. Im Unterschied zu dem naiven Bayes'schen Klassifikator werden keine Verteilungsannahmen gemacht. Vorwissen kann allerdings an anderer Stelle, nämlich bei der Definition des sog. Abstandsmaßes, eingebracht werden. Dieses Abstandsmaß wird dazu benutzt, ähnliche Situationen aus der Vergangenheit zu finden. Es sollte so beschaffen sein, daß Dokumente, die anhand dieses Maßes ähnlich sind, möglichst zur selben Klasse gehören. Die Grundidee des k-Nearest-Neighbor-Verfahrens ist es, ein neues Beispiel so zu klassifizieren wie die ähnlichsten Beispiele, für die man die Klassifikation kennt.

Im folgenden werde ich die k-Nearest-Neighbor-Entscheidungsregel für den Euklidschen Abstand herleiten und anschließend die Herleitung für das normalerweise bei der Textklassifikation verwendete Ähnlichkeitsmaß, den Kosinus, erweitern. Das k-Nearest-Neighbor-Verfahren läßt sich, wie im vorangegangenen Abschnitt der Bayes'sche Klassifikator, aus Bayes' Regel herleiten [Michie et al., 1994]. Bayes' Regel besagt, daß der optimale Klassifikator sich dann ergibt, wenn Dokumente d in die Klasse $C \in \mathcal{C}$ klassifiziert werden, für die $\Pr(C|d)$ maximal ist.

$$H_{kNN}(d) = \operatorname*{argmax}_{C \in \mathcal{C}} \Pr(C|d) \tag{16}$$

Wieder kann Bayes' Theorem [James, 1985] dazu benutzt werden, die Wahrscheinlichkeit $\Pr(C|d)$ in zwei Teile aufzuspalten. Der Beschreibungsvektor $\vec{d}$ von d kann hier allerdings nicht als diskret angenommen werden und muß daher durch eine Wahrscheinlichkeitsdichtefunktion $p(\vec{d}|C)$ modelliert werden.

$$\Pr(C|d) = \frac{p(\vec{d}|C) \cdot \Pr(C)}{\sum_{C \in \mathcal{C}} p(\vec{d}|C) \cdot \Pr(C)} \tag{17}$$

$\hat{\Pr}(C)$, der Schätzwert für $Pr(C)$, wird, wie auch beim naiven Bayes'schen Klassifikator, aus dem Anteil der Trainingsbeispiele berechnet, die sich in der entsprechenden Klasse befinden.

$$\hat{\Pr}(C) = \frac{|C|}{\sum_{C' \in \mathcal{C}} |C'|} \tag{18}$$

$|C|$ ist die Anzahl der Dokumente in der Klasse.

Der k-Nearest-Neighbor-Algorithmus unterscheidet sich vom naiven Bayes'schen Klassifikator in der Bestimmung von $p(\vec{d}|C)$. Geht man von der Euklidschen Distanz als Abstandsmaß aus, kann $p(\vec{d}|C)$ wie folgt geschätzt werden.

Man ermittelt die k nächsten Nachbarn $\vec{d_1}, ..., \vec{d_k}$ zu dem zu klassifizierenden Beispiel $\vec{d}$. Sei V der Raum in der Kugel um $\vec{d}$, die genau die k nächsten Nachbarn einschließt. Dann berechnet sich die Wahrscheinlichkeit $\Pr(\vec{d} \in V|C)$, daß ein beliebiges Dokument d aus Kategorie C in dieser Kugel auftaucht, als das Integral

$$\Pr(\vec{d} \in V|C) = \int_V p(\vec{d'}|C)\delta\vec{d'} \tag{19}$$

Wie kann dieses Integral berechnet werden? Nimmt man an, daß $p(\vec{d'}|C)$ auf V hinreichend homogen ist (d. h. $p(\vec{d'}|C)$ ist ungefähr gleich für alle Dokumente in V), kann das Integral aufgelöst werden. Da $p(\vec{d}|C)$ in V liegt, wird es zu

$$\Pr(\vec{d} \in V|C) = p(\vec{d}|C) * \int_V 1\delta\vec{d'} \tag{20}$$

Das Integral reduziert sich auf $v := \int_V 1\delta\vec{d'}$, das Volumen von V. Abkürzend kann man also schreiben:

$$\Pr(\vec{d} \in V|C) = p(\vec{d}|C) * v \tag{21}$$

Die Wahrscheinlichkeit $\Pr(\vec{d} \in V|C)$ kann aus den nächsten Nachbarn $\vec{d_1}, ..., \vec{d_k}$ geschätzt werden. Man ermittelt die Anzahl k_C der k nächsten Nachbarn, die in Kategorie C sind, und teilt durch die Gesamtzahl $|C|$ der Dokumente in C.

$$\hat{\Pr}(\vec{d} \in V|C) = \frac{k_C}{|C|} \tag{22}$$

Durch Einsetzen von (22) in (21) und durch Äquivalenzumformungen erhält man einen Schätzer für die gesuchte Dichtefunktion.

$$\hat{p}(\vec{d}|C) = \frac{k_C}{v * |C|} \tag{23}$$

Setzt man (23) und (18) in (17) ein, erhält man

$$\hat{\Pr}(C|d) = \frac{\frac{k_C}{v*|C|} \cdot \frac{|C|}{\sum_{C'' \in \mathcal{C}} |C''|}}{\sum_{C' \in \mathcal{C}} \frac{k_{C'}}{v*|C'|} \cdot \frac{|C'|}{\sum_{C'' \in \mathcal{C}} |C''|}} \tag{24}$$

was sich durch Kürzen auf

$$\hat{\Pr}(C|d) = \frac{k_C}{\sum_{C' \in \mathcal{C}} k_C} = \frac{k_C}{k} \tag{25}$$

reduzieren läßt.

Die Entscheidungsregel für den k-Nearest-Neighbor-Algorithmus hat also die einfache Form:

$$H_{kNN}(d) = \operatorname*{argmax}_{C \in \mathcal{C}} k_C \tag{26}$$

$\hat{\Pr}(C|d)$ kann als Konfidenz für die Entscheidung verwendet werden.

Für Textkategorisierungsprobleme haben sich andere Abstandsmaße als die Euklidsche Distanz bewährt. Probleme treten bei Verwendung des Euklidschen Abstandes auf, wenn Dokumente in der Vektorraum-Repräsentation mit TFIDF-Gewichtung dargestellt werden. Dokumente unterschiedlicher Länge haben selbst dann großen Euklidschen Abstand, wenn sie die gleiche Verteilung von Worten enthalten. Um dem zu entgegnen, benutzt man im Information Retrieval den Winkel bzw. den monoton mit dem Winkel verlaufenden Kosinus zwischen Dokumentvektoren als Abstandsmaß. Der Kosinus ist unabhängig von der Dokumentlänge und hat sich als Ähnlichkeitsmaß für Retrievalanwendungen bewährt.

Obige Herleitung der k-NN-Entscheidungsregel wurde für den Euklidschen Abstand als Ähnlichkeitsmaß durchgeführt. Im folgenden werde ich zeigen, daß sich auch für den Kosinus die gleiche Entscheidungsregel ergibt.

Behauptung: Für den Kosinus als Ähnlichkeitsmaß ergibt sich die gleiche Entscheidungsregel (26) wie für den Euklidschen Abstand als Distanzmaß.

Beweis: Bis zur Gleichung (19) verläuft der Beweis mit dem Kosinus als Abstandsmaß analog zu dem für den Euklidschen Abstand. Allerdings gilt für den Kosinus die folgende Abschätzung nicht mehr.

$$\Pr(\vec{d} \in V|C) = p(\vec{d} \in V|C) * \int_V 1 \delta \vec{d'} \tag{27}$$

Im allgemeinen ist der Raum V, der durch den Kosinus bestimmt wird, nicht endlich. Die Annahme, daß $p(\vec{d'} \in V|C)$ homogen ist in V, kann also nicht mehr zutreffen. Für jeden Wert größer 0 für $p(\vec{d'} \in V|C)$ würde $\Pr(\vec{d} \in V|C)$ gleich ∞.

Um dies zu beheben, muß man einen anderen Wahrscheinlichkeitsraum betrachten. Sei der neue Wahrscheinlichkeitsraum nicht mehr die Menge aller beliebigen Vektoren, sondern nur noch die Menge der Vektoren mit der Länge 1. Jeder Vektor $\vec{d}$ (ausgenommen der Nullvektor) kann durch folgende Normierung auf einen Vektor $\vec{d_1}$ der Länge 1 abgebildet werden.

$$\vec{d_1} = \frac{\vec{d}}{||\vec{d}||} = \frac{\vec{d}}{\sqrt{\vec{d} \cdot \vec{d}}} \tag{28}$$

Durch diese Normierung wird genau die durch die Wahl des Kosinus gewünschte Unabhängigkeit von der Dokumentlänge erreicht. Die Berechnung des Kosinus vereinfacht sich zu

$$\begin{aligned} \cos(\angle(\vec{d}, \vec{d'})) &= \frac{\vec{d} \cdot \vec{d'}}{||\vec{d}|| \cdot ||\vec{d'}||} && (29) \\ &= \vec{d} \cdot \vec{d'} && (30) \end{aligned}$$

Für normierte Dokumentvektoren reduziert sich auch die Berechnung der Euklidschen Distanz $d(\vec{d}, \vec{d'})$ auf eine einfachere Form.

$$\begin{aligned} d(\vec{d}, \vec{d'}) &= \sqrt{(\vec{d} - \vec{d'})^2} && (31) \\ &= \sqrt{\vec{d}^2 - 2 \cdot \vec{d} \cdot \vec{d'} + \vec{d'}^2} && (32) \\ &= \sqrt{1 - 2 \cdot \vec{d} \cdot \vec{d'} + 1} && (33) \\ &= 2 \cdot \sqrt{1 - \vec{d} \cdot \vec{d'}} && (34) \end{aligned}$$

Wie man leicht sieht, besteht für normierte Vektoren zwischen Kosinus und Euklidscher Distanz ein monotoner Zusammenhang.

$$d(\vec{d}, \vec{d'}) = 2 \cdot \sqrt{1 - \cos(\vec{d}, \vec{d'})} \tag{35}$$

Für normierte Vektoren sind also die k Vektoren mit dem größten Kosinus gleich den k nächsten Nachbarn anhand des Euklidschen Abstandes. Der Kosinus ist somit für normierte Vektoren auf den Euklidschen Abstand zurückgeführt. Da für den Euklidschen Abstand die k nächsten Nachbarn immer in einem endlichen Raum liegen, ist das Integral $\int_V 1 \delta \vec{d'}$ aus Zeile (27) jetzt auch endlich. Dies gilt, weil der neue Wahrscheinlichkeitsraum, über den jetzt integriert wird, ein Teilraum von $\Re^{|F|}$ ist. Die Herleitung der Entscheidungsregel verläuft nun wie oben beschrieben, wenn man ab Gleichung (19) für V den Schnittraum mit dem neuen Wahrscheinlichkeitsraum betrachtet. $\Box$

Experimente mit k-Nearest-Neighbor-Verfahren zur Textklassifikation werden in [Masand et al., 1992] beschrieben. Ein auf dem Vektorraummodell beruhendes Information-Retrieval-System dient zur Berechnung der nächsten Nachbarn. Der Einfachheit und der guten Anwendbarkeit dieses Verfahren stehen die vergleichsweise hohen Kosten in Bezug auf Rechenzeit gegenüber.

Weitere Experimente mit k-Nearest-Neighbor-Verfahren zur Textklassifikation finden sich in [Freitag et al., 1996] und [Yang, 1996].

3.4.4 Winnow

Winnow [Littlestone, 1987] lernt eine lineare Schwellwertfunktion, die zur Klassifikation von Texten benutzt werden kann. Ähnlich wie bei dem Rocchio-Algorithmus aus Abschnitt 3.4.1 ist Winnow nur auf binäre Klassifikationsprobleme (o. B. d. A. mit Klassen + und −) anwendbar. Der Winnow-Algorithmus wurde im Bereich der algorithmischen Lerntheorie entwickelt. Es kann gezeigt werden, daß dieser Algorithmus in der Lage ist, bei einer angemessenen Parameterwahl jedes linear separierbare Problem effizient zu lernen [Littlestone, 1987]. Obwohl Textklassifikationsprobleme im allgemeinen nicht linear separierbar sind, bestärkt empirische Evidenz [Blum, 1995] die Vermutung, daß sich Winnow auch hier erfolgreich einsetzen läßt.

Wie bereits erwähnt, ist Winnows Hypothesensprache die Menge der linearen Schwellwertfunktionen. Sei $OCC(w,d)$ gleich 1, wenn das Wort w in Dokument d mindestens einmal vorkommt. Um ein Dokument d zu klassifizieren, benutzt Winnow für einen gegebenen Satz von Gewichten $weight(w)$ die folgende Entscheidungsregel (36). Die Summe $\sum_{w \in F} weight(w) \cdot OCC(w,d)$ wird berechnet und mit dem Schwellwert θ verglichen. Ist die Summe größer als θ, wird das Dokument als zu + zugehörig klassifiziert. Ansonsten als −.

$$H_{WINNOW}(d) = \begin{cases} + & \sum_{w \in F} weight(w) \cdot OCC(w,d) \geq \theta \\ \\ - & otherwise \end{cases} \tag{36}$$

F ist die Menge der Attribute. Die Höhe von $\sum_{w \in F} weight(w) \cdot OCC(w,d)$ kann als Konfidenz für die Zugehörigkeit zur Klasse + verwendet werden. Im Gegensatz zu den bis hierhin vorgestellten Algorithmen benutzt Winnow also nicht die Häufigkeit, mit der ein Wort in einem Dokument vorkommt.

Entscheidend für die Klassifikation sind die Gewichte $weight(w)$ für jedes Wort w. Im Gegensatz zu den anderen hier vorgestellten Klassifikationsmethoden beruht der Winnow-Algorithmus auf Suche durch einen Hypothesenraum und explizite Fehlerminimierung. Die Gewichte $weight(w)$ werden während der Trainingsphase anhand des folgenden Schemas eingestellt. Die Lernrate γ ist vorgegeben.

Durchlaufe alle Dokumente der Trainingsmenge m-mal und wende die folgenden Regeln für jedes Dokument an:

- Winnow klassifiziert das Dokument d als +, aber die korrekte Klassifikation ist −: Die Gewichte $weight(w)$ für Wörter w mit $OCC(w,d) = 1$ werden durch die Lernrate γ geteilt.
- Winnow klassifiziert das Dokument d als −, aber die korrekte Klassifikation ist +: Die Gewichte $weight(w)$ für Wörter w mit $OCC(w,d) = 1$ werden mit der Lernrate γ multipliziert.
- Winnows Klassifikation ist korrekt: Die Gewichte bleiben unverändert.

Winnow minimiert auf diese Weise die Anzahl der fehlerhaft klassifizierten Beispiele, indem es eine Hyperebene findet, die positive und negative Beispiele trennt. Es handelt sich also um ein Diskriminanzanalyseverfahren [Michie et al., 1994].

Ein anderer Algorithmus, der auch auf dem Hypothesenraum der linearen Schwellwertfunktionen arbeitet, ist der Perceptron Algorithmus [Rosenblatt, 1962]. Im Gegensatz zur Lernregel von Winnow, die Gewichte multiplikativ verändert, benutzt der Perceptron Algorithmus eine additive Lernregel. Die Hauptvorteile von Winnow gegenüber dem Perceptron sind seine guten Online-Learning-Eigenschaften. Online Learning bedeutet, daß Winnow auch während der Anwendungsphase seine Gewichte auf einfache Weise verändern kann. Wann immer der Algorithmus während der Anwendungsphase eine falsche Klassifikation macht, können die Gewichte anhand der obigen Regeln verändert werden. Für linear separierbare Probleme kann gezeigt werden, daß die Anzahl der Klassifikationsfehler, die Winnow im Worst-Case macht, nur um einen konstanten Faktor vom Optimum entfernt ist [Littlestone, 1987]. Die Fähigkeit des Lernalgorithmus zum effizienten Online Learning wäre für WebWatcher zu begrüßen. Nach einem schlechten Hinweis während des Betriebs könnte die Hypothese des Systems so jeweils verbessert werden. Um jedoch die Vergleichbarkeit zu den anderen hier vorgestellten Algorithmen sicherzustellen, wurden die Gewichte in den hier beschriebenen Experimenten nach der Trainingsphase fixiert.

3.4.5 Weitere Ansätze

Auch beim Verfahren der *logistischen Regression* (siehe z. B. [Michie et al., 1994]) wird versucht, die Wahrscheinlichkeit $\Pr(C|d)$ zu schätzen. Logistische Regression findet eine Hyperebene im Attributraum, so daß positive und negative Beispiele bestmöglich getrennt werden. Die Hyperebene wird so gewählt, daß das konditionale Likelihood über den Trainingsbeispielen maximiert wird. Experimente zur Textklassifikation mit logistischer Regression werden in [Schütze et al., 1995] beschrieben. Eng verwandt mit logistischer Regression sind *neuronale Netze* und *Backprop* (siehe z. B. [Mitchell, 1997]). Diese Art von neuronalen Netzen berechnet mehrstufige logistische Regressionen. Schütze et al. [Schütze et al., 1995] und Wiener et al. [Wiener et al., 1995] benutzen neuronale Netze in Verbindung mit Latent Semantic Indexing. Ähnliche Ansätze werden auch von Yang verfolgt [Yang, 1996]. Sie benutzt *lineare Regression* zur Textkategorisierung. Lineare Regression ist ein Spezialfall von *Regression mit Polynomen* [Fuhr, 1989], wie sie bei dem System AIR/PHYS [Fuhr und Knorz, 1984] verwendet wird.

Ähnlich zum naiven Bayes'schen Klassifikator sind *Bayesian Inference Networks*, wie sie in [Tzeras und Hartmann, 1993] zur Anwendung kommen. Wie beim naiven Bayes'schen Klassifikator wird ein probabilistisches Modell erstellt, anhand dessen klassifiziert wird. Allerdings erlauben Bayesian Inference Networks eine genauere Modellierung der Abhängigkeitsbeziehungen zwischen Termen.

Neben diesen auf statistischen Lernverfahren beruhenden Ansätzen werden auch *Regellerner* zur Textkategorisierung eingesetzt. Die im folgenden vorgestellten Ansätze unterscheiden sich im wesentlichen durch die Repräsentation von Beispielen und die Mächtigkeit des Hypothesenraumes. Es werden zwei Repräsentationsformalismen unterschieden: Propositionale und relationale Repräsentation.

Die *propositionale Darstellung* von Dokumenten ist der Vektorraumrepräsentation sehr ähnlich. Jedes in den Trainingsdokumenten vorkommende Wort ist ein Attribut. Der Wert eines Attributs ist 1, wenn das betreffende Wort in dem Dokument vorkommt. Ansonsten ist er 0.

Lewis und Ringuette [Lewis und Ringuette, 1994] vergleichen einen Entscheidungsbaumlerner mit einem naiven Bayes'schen Klassifikator. Beide Verfahren benutzen dieselbe wortbasierte Attribut-Wert-Darstellung von Dokumenten und erreichen ähnliche Genauigkeit. Die Entscheidungsbaummethode liefert ein für einen Menschen besser interpretierbares Entscheidungsmodell, benötigt aber einen höheren Rechenaufwand als der naive Bayes'sche Klassifikator.

In [Apté und Damerau, 1994] wird das Regellernverfahren SWAP-1 [Weiss und Indurkhya, 1993] auf einer propositionalen Repräsentation eingesetzt. SWAP-1 lernt eine Menge von Regeln mit Konjunktionen von Literalen im Regelrumpf und dem Zielattribut im Regelkopf. Die Repräsentation ist in dem Sinne erweitert, daß auch Paare von aufeinanderfolgenden Worten als Attribute berücksichtigt werden. Weitere Experimente mit ähnlichen Regellernern finden sich in [Cohen, 1996] [Moulinier et al., 1996] [Moulinier und Ganascia, 1996].

Relationale Formalismen sind in ihrer Ausdruckskraft einfachen propositionalen Repräsentationen überlegen. Neben den Attributen eines Dokuments können in einer relationalen Darstellung z. B. auch Relationen zwischen den Attributen ausgedrückt werden. Hierdurch kann die Reihenfolge von Worten repräsentiert werden, welche bei propositionaler Darstellung in der Regel verloren geht. Cohen [Cohen, 1995b] [Cohen, 1995a] benutzt die folgenden Relationen zwischen den Attributen (Worten) eines Dokuments.

- $near1/2$: Zwei Worte sind im Text benachbart.
- $near2/2$: Zwei Worte sind höchstens durch ein Wort getrennt.
- $near3/2$: Zwei Worte sind höchstens durch zwei Worte getrennt.
- $after/2$: Ein Wort erscheint im Text hinter einem anderen.
- $succ/2$: Ein Wort erscheint im Text direkt hinter einem anderen.

	T(d)="+"	T(d)="-"
H(d)="+"	A	B
H(d)="-"	C	D

Abbildung 13: Contingency Table für ein Klassifikationsproblem mit zwei Klassen. T(d) entspricht der korrekten Klassifikation, H(d) ist die Klassifikation des Lernalgorithmus.

Cohen wendet die ILP-Verfahren FOIL6 [Quinlan und Cameron-Jones, 1993] und FLIPPER [Cohen, 1995a] auf dieser Repräsentation an. Im Vergleich zu verschiedenen propositionalen Lernverfahren kommt Cohen zu dem Schluß, daß die mächtigere relationale Repräsentation bei manchen Problemen bessere Ergebnisse ermöglicht. Die Verbesserungen sind allerdings nur gering und Cohen bemerkt, daß andere Parameter größeren Einfluß auf die Lernqualität haben [Cohen, 1995a].

3.5 Performanzmaße

Im folgenden wird eine Auswahl der im Information Retrieval und im maschinellen Lernen verwendeten Performanzmaße vorgestellt. Insbesondere werden *Accuracy* und *Precision/Recall* genauer erläutert, da sie bei den hier beschriebenen Experimenten Verwendung finden. Im folgenden werden Maße anhand des "contingency tables" aus Abbildung 13 erläutert.

Das Maß *Accuracy* ist im maschinellen Lernen die gebräuchlichste Art, die Performanz eines lernenden Systems zu messen. Accuracy ist die Wahrscheinlichkeit, daß ein zufällig aus der Verteilung der Beispiele gezogenes Beispiel richtig klassifiziert wird. Oftmals wird auch die Gegenwahrscheinlichkeit betrachtet, die als *Fehlerrate* bezeichnet wird. Ein Schätzer für die Accuracy ergibt sich daraus, daß man die Anzahl der korrekten Klassifizierungen durch die Gesamtanzahl der Klassifizierungen teilt. Bei einem Klassifikationsproblem mit zwei Klassen ergibt sich in Bezug auf Abbildung 13 folgende Formel:

$$Accuracy = \frac{A + D}{A + B + C + D} = 1 - Fehlerrate \tag{37}$$

Obwohl Accuracy ein sehr anschauliches und universelles Maß für die Performanz eines Klassifikationssystems ist, gibt es Probleme, bei denen andere Maße die Leistung praxisorientierter messen. Bei manchen Problemen sind Fehlklassifizierungen in einer Klasse mit höheren Kosten verbunden als in anderen Klassen. Bei einem Diagnosesystem, das z. B. Patienten in die Klassen "Krebs" und "keinen Krebs" klassifizieren soll, möchte man die Anzahl der kranken Patienten, die man fälschlicherweise als gesund nach Hause schickt, minimieren. Dies kann man z. B. durch ein Kostenmodell erreichen (siehe z. B. [Michie et al., 1994]).

Weitere Möglichkeiten zur Performanzmessung bieten *Precision/Recall*-basierte Maße. Sie lassen sich nur auf Klassifikationsprobleme mit zwei Klassen anwenden. Eine der Klassen wird speziell ausgezeichnet und tritt meist weniger häufig auf als die andere. Beim Information Retrieval ist dies die Klasse der relevanten Dokumente, denen eine große Anzahl von nichtrelevanten Dokumenten gegenübersteht. Eine gebräuchliche Definition von Recall und Precision ist die folgende [Raghavan et al., 1989]. Recall ist die Wahrscheinlichkeit, daß ein relevantes Dokument von dem System als solches erkannt wird. Precision ist die Wahrscheinlichkeit, daß ein vom System als relevant gekennzeichnetes Dokument wirklich relevant ist. Anhand des Contingency Tables lassen sich die folgenden Schätzer beschreiben:

$$Precision = \frac{A}{A+B} \qquad Recall = \frac{A}{A+C} \tag{38}$$

Da sich zwischen Recall und Precision ein Trade-Off ergibt, werden Precision/Recall-Graphen betrachtet. Der Recall wird künstlich verändert und die Precision wird z. B. an festgelegten Recall-Punkten gemessen. Zur Veränderung des Recalls werden die Beispiele anhand ihrer geschätzten Relevanz geordnet. Durch Abstieg in der Rangordnung gelangt man, ggf. nach Interpolation, zu den gewünschten Recall-Punkten.

Bei nur schwach geordneten Rangordnungen muß beachtet werden, daß mehrere Dokumente den gleichen Rang haben können. Verschiedene Ansätze wurden zur Behandlung dieses Falles vorgeschlagen, wie z. B. die PRECALL-Methode oder die erwartete Precision [Raghavan et al., 1989] [Bollmann et al., 1992]. Konsistent mit der obigen Definition von Precision ist die PRR Methode [Raghavan et al., 1989]. Bei der PRR Methode wird jeweils für den letzten Rang untersucht, wieviele irrelevante Dokumente man erwartungsgemäß in Kauf nehmen muß, um den gewünschten Recall-Wert *recall* zu erreichen. In der folgenden Formel ist n die Gesamtzahl der relevanten Dokumente. r ist die Anzahl der relevanten und i die Anzahl der irrelevanten Dokumente im letzten Rang. j ist die Anzahl der irrelevanten Dokumente in Rängen oberhalb des letzten Ranges und s ist die Anzahl der relevanten Dokumente des letzten Ranges, die zum Erreichen des gewünschten Recall-Wertes noch benötigt werden. Dann berechnet sich die Precision als

$$Precision(recall) = \frac{recall \cdot n}{recall \cdot n + j + s \cdot i/(r+1)} \tag{39}$$

Diese Methode wird bei den hier beschriebenen Experimenten verwendet.

Sollen Precision/Recall über mehrere binäre Klassifikationsaufgaben gemittelt werden, bieten sich zwei Methoden an. Bei der *Makrobewertung* werden Precision/Recall-Graphen für jede Aufgabe[12] getrennt berechnet und anschließend gemittelt. Hierbei ergeben sich insbesondere dann Probleme, wenn bei einer der Aufgaben keines der Testbeispiele relevant ist. Die Berechnung des Recall nach der oben angegebenen Formel ist dann durch eine Null im Nenner nicht

[12] oder, bei Verwendung der gleichen Beispiele zum Lernen unterschiedlicher Zielfunktionen, für jedes Beispiel

definiert. Behandlungen dieses Spezialfalls werden in [Fuhr und Knorz, 1984] und [Tague, 1981] vorgeschlagen. Unproblematischer ist die *Mikrobewertung*, die bei der Textkategorisierung bevorzugt verwendet wird. Die Klassifikationen aller Aufgaben werden in einer Rangordnung zusammengefaßt, für welche dann Precision und Recall berechnet wird. Voraussetzung ist die Kompatibilität der Relevanzmaße zwischen den Aufgaben.

Um eine einzelne Zahl zur Bewertung eines Klassifikationssystems zu erhalten, stehen aufbauend auf den gemittelten Precision/Recall-Graphen drei Methoden zur Verfügung. Bei der ersten Methode wird der Mittelwert der Precision über verschiedenen Recall-Punkten (z. B. $0.25, 0.5, 0.75$) als Bewertung herangezogen. Andere Bewertungen sind der Breakeven-Point[13] von Recall und Precision oder das E-Maß (siehe [van Rijsbergen, 1979], Seiten 168-176).

Kurz erwähnt seien alternative Ansätze zur Performanzmessung. Ein Ansatz aus dem maschinellen Lernen ist es, den Informationsgewinn [Shannon und Weaver, 1949], den der Klassifikationsalgorithmus besteuert, zu messen und zur Bewertung heranzuziehen. Andere Informationsmaße werden im Information Retrieval betrachtet, um den Informationsgehalt einer Ordnung von Dokumenten im Vergleich zu einer optimalen Ordnung zu ermitteln [Tague-Sutcliffe, 1992]. Eine weitere Methode zum Vergleich von Rangordnungen ist das Nützlichkeitsmaß von Frei und Schäuble [Frei und Schäuble, 1991].

3.6 Das Vorschlagen von Hyperlinks als Textkategorisierung

Wie läßt sich Textkategorisierung auf WebWatchers Aufgabe anwenden? Wir betrachten die Funktion $UserChoice?$, wie sie in Abschnitt 2.4 eingeführt wurde.

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

Die Funktion gibt an, welchen Hyperlinks ein Benutzer mit dem Interesse $i \in Interest$ auf einer Seite $p \in Page$ wahrscheinlich folgt. Wenn ein Benutzer in Begleitung von WebWatcher auf einer WWW-Seite ankommt, können die von $UserChoice?$ vorhergesagten Hyperlinks auf der Seite hervorgehoben werden, so daß die Aufmerksamkeit des Benutzers auf besonders relevante Teile der Seite fokussiert wird.

Um die Funktion $UserChoice?$ handhabbarer zu machen, wird sie zunächst in separate Teilfunktionen $UserChoice?_p$ für jede Seite $p \in Page$ aufgespalten.

$$UserChoice?_p : Interest \rightarrow 2^{Links(p)}$$

Generalisierungen über mehrere Seiten werden hiermit ausgeschlossen.

Da $UserChoice?_p$ in die Potenzmenge der Hyperlinks auf der Seite abbildet, kann jede Teilmenge von $Links(p)$ als Funktionswert auftreten. Wollte

[13] Schnittpunkt des Precision/Recall-Graphen mit der Identitätsfunktion

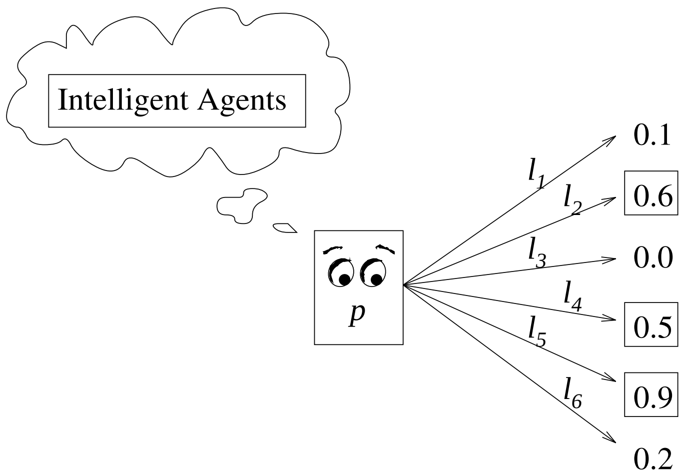

Abbildung 14: Auf jeder Seite p wird für jedes Hyperlink $l_1, ..., l_n$ anhand von $UserChoice?_{p,l}$ die Konfidenz bestimmt, daß ein Benutzer mit einem gegebenen Interesse dem Hyperlink folgen wird. In dem obigen Beispiel stehen sechs Hyperlinks zur Auswahl von denen $k = 3$ vorgeschlagen werden dürfen. In Bezug auf das Interesse "Intelligent Agents" erhalten die drei Hyperlinks l_2, l_4 und l_5 die höchste Konfidenz und werden vorgeschlagen.

man $UserChoice?_p$ direkt als Klassifikationsproblem formulieren, müßte man jede Teilmenge von $Links(p)$ als eigene Klasse betrachten. Es ergäben sich also $2^{|Links(p)|}$ Klassen! Diese Anzahl von Klassen für jede Seite wäre so groß, daß es im Rahmen von WebWatcher nicht möglich wäre, genügend Trainingsbeispiele zu sammeln. Dem zu entgegnen hilft die Annahme, daß die Wahl eines Hyperlinks nicht von der Wahl anderer Hyperlinks abhängt. Mit dieser Annahme kann die Funktion $UserChoice?_p$ weiter vereinfacht werden. Für jedes Hyperlink $l \in Links(p)$ wird eine Funktion $UserChoice?_{p,l}$ eingeführt, die angibt, ob ein Benutzer mit einem bestimmten Interesse dem Hyperlink l folgen wird.

$$UserChoice?_{p,l} : Interest \rightarrow \{+, -\} \tag{40}$$

Das Klassifikationsproblem nach $2^{Links(p)}$ wurde somit in $|Links(p)|$ binäre Klassifikationsprobleme aufgespalten, wobei die 2 Klassen jeweils dafür stehen, daß der Benutzer dem Hyperlink folgt oder nicht.

Aus den Funktionen $UserChoice?_{p,l}$ für eine Seite p kann zum Zeitpunkt einer Klassifikation der Funktionswert von $UserChoice?_p$ wieder zusammengesetzt werden. Alle Klassifikationsmethoden, die in den Abschnitten 3.4.1 bis 3.4.4 vorgestellt wurden, liefern nicht nur die von ihnen vorgeschlagene Klassifikation als Ausgabe, sondern zusätzlich eine Konfidenz für ihre Entscheidung. Im Bezug auf $UserChoice?_{p,l}$ ist dies die Konfidenz (im Idealfall eine Schätzung der Wahrscheinlichkeit), daß der Benutzer dem Hyperlink l folgen wird. Mit diesen Konfidenzen kann eine Rangordnung erstellt werden, indem die Hyperlinks nach absteigender Konfidenz sortiert werden. Die obersten k Hyperlinks werden von WebWatcher als die Hyperlinks vorhergesagt, denen der Benutzer wahrscheinlich folgen wird. Abbildung 14 illustriert dies an einem Beispiel. In dem Fall, daß die Konfidenz entweder der Wahrscheinlichkeit entspricht oder doch zumindest dieselbe Rangordnung erzeugt, läßt sich in der Decision Theory eine Rechtfertigung für dieses Vorgehen finden. In einem linearen Kostenmodell liefert eine so erzeugte Rangordnung minimale Kosten, wenn eine vorgegebene Anzahl k von Hyperlinks vorgeschlagen werden soll. Im Information Retrieval ist dies unter dem Namen *Probability Ranking Principle* [Robertson, 1977] bekannt.

Die hier vorgestellte Modellierung des Problems entspricht konzeptionell in weiten Teilen dem in [Fuhr und Buckley, 1991] beschriebenen *Binary Independence Indexing (BII) Model*, das auf den Arbeiten von Maron und Kuhns [Maron und Kuhns, 1960] basiert. Betrachtet man Hyperlinks als Dokumente des BII-Modells und die Benutzerinteressen als Anfragen, findet ein dokumentorientiertes Lernen in WebWatcher statt. Für eine feste Menge von Dokumenten (Hyperlinks) wird einzeln gelernt, für welche Anfragen (Interessen) das jeweilige Dokument (Hyperlink) relevant ist. Bei Fuhr und Buckley wird hierzu ein probabilistisches Modell benutzt, welches ähnlich zu dem in Abschnitt 3.4.2 vorgestellten naiven Bayes'schen Klassifikator ist.

3.7 Aufbau des Experiments

In den folgenden Experimenten wird die Performanz der vorgestellten Textklassifikationsalgorithmen untereinander verglichen und der Performanz der von WebWatcher verwendeten Methode gegenübergestellt. Es soll untersucht werden, ob und wie von vorangegangenen Benutzerinteraktionen in WebWatcher gelernt werden kann. Die Resultate werden anschließend analysiert und interpretiert. Um das Experiment handhabbarer und besser auswertbar zu machen, wird im folgenden eine Seite exemplarisch betrachtet. Es handelt sich bei dieser Seite um eine Version der SCS-Front-Door-Seite, dargestellt in Abbildung 15. Es ist eine ältere Version des Dokuments aus Abbildung 1.

Die SCS-Front-Door-Seite wurde gewählt, da für sie die meisten Trainingsdaten vorhanden sind. Trainingsbeispiele werden aus Touren generiert, die WebWatcher vom 2. August 1995 bis zum 31. Oktober 1995 gegeben hat. Für jede Tour, auf welcher der Benutzer einem oder mehreren Hyperlinks auf der Seite gefolgt ist, werden die entsprechenden Hyperlinks zusammen mit der eingegebenen Interessenbeschreibung als Beispiel abgespeichert. Nur die Beispiele werden betrachtet, deren Interessenbeschreibung nicht leer ist. Beispiele entsprechen Tupeln der Funktion

$$UserChoice?_{SCS-Front-Door} : Interest \rightarrow 2^{Links(SCS-Front-Door)} \tag{41}$$

Die Funktion bildet das vom Benutzer eingegebene Interesse auf die Menge der Hyperlinks ab, denen der Benutzer auf der SCS-Front-Door-Seite gefolgt ist. Insgesamt umfaßt der Datensatz 408 Beispiele dieser Art. Es ist zu beachten, daß WebWatcher den Benutzer in seiner Wahl von Hyperlinks möglicherweise

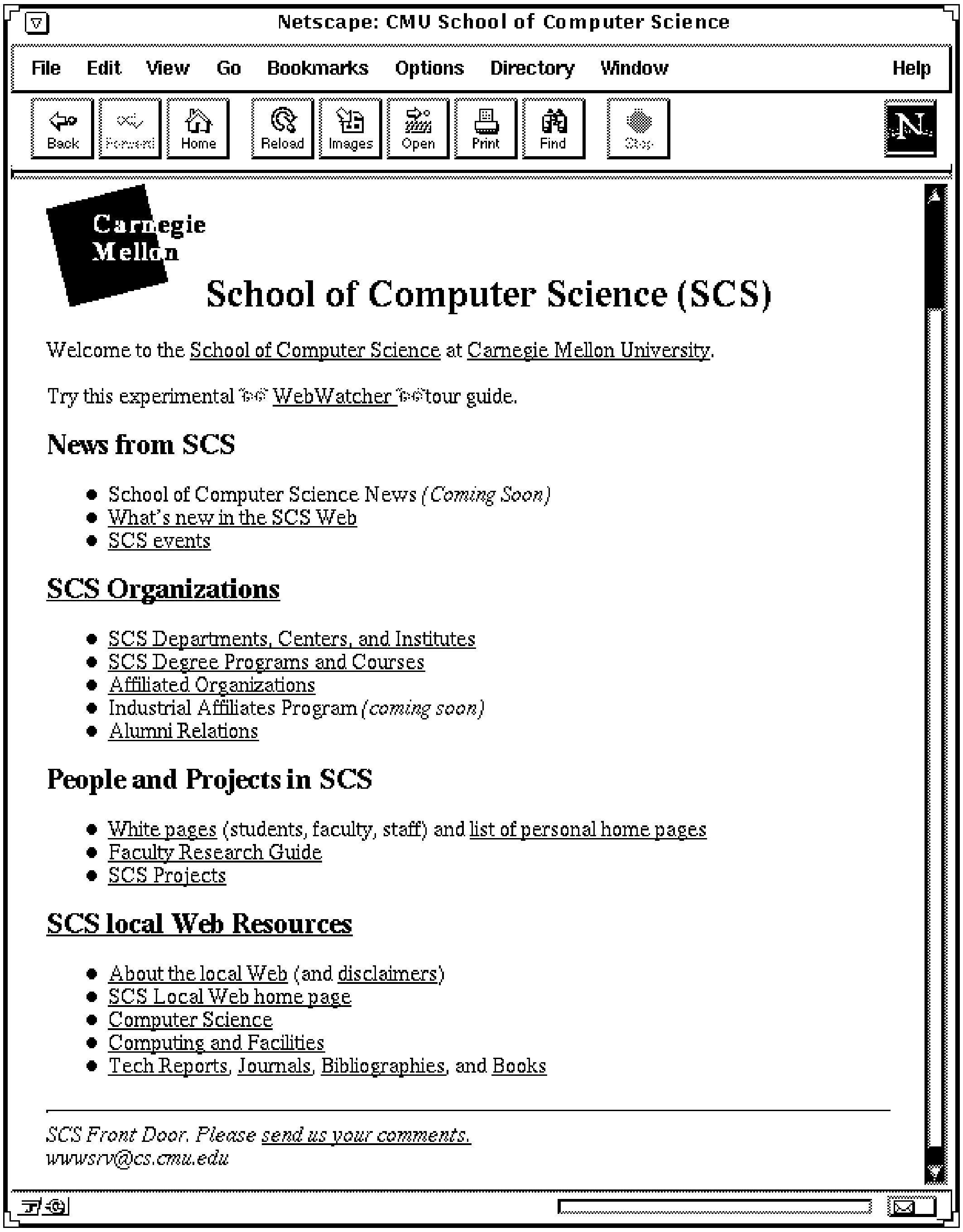

Abbildung 15: Eine ältere Version der SCS-Front-Door-Seite.

beeinflußt. Bei den in diesem Experiment benutzten Daten kann es also sein, daß WebWatchers Lernmethode besser abschneidet, da diese Methode bereits bei der Präsentation der Seite verwendet wurde, um Hyperlinks hervorzuheben.

Formal läst sich das Lernproblem wie folgt definieren. Die Funktion $UserChoice?_{SCS-Front-Door}$ soll gelernt werden. Hierzu wird die Funktion wie im vorangegangenen Abschnitt beschrieben in Teilfunktionen $UserChoice?_{SCS-Front-Door,l}$ zerlegt, die separat gelernt werden. Aus jedem Beispiel für $UserChoice?_{SCS-Front-Door}$ wird genau ein positives bzw. negatives Beispiele für jede Funktion $UserChoice?_{SCS-Front-Door,l}$ generiert. Die Interessenbeschreibungen in den Beispielen werden in der Bag-of-Words-Darstellung repräsentiert. Es wird von der Stammformreduktion nach Porter (siehe Abschnitt 3.3.2) Gebrauch gemacht. Aus den $UserChoice?_{SCS-Front-Door,l}$ wird zum Zeitpunkt einer Klassifikation ein Funktionswert für $UserChoice?_{SCS-Front-Door}$ anhand der im vorangegangenen Abschnitt beschriebenen Methode zusammengesetzt.

Insgesamt befinden sich 18 unterschiedliche Hyperlinks auf der SCS-Front-Door-Seite, denen der Benutzer folgen kann. Das Hyperlink *"send us your comments"* wurde nicht betrachtet, da es sich um einen *"mailto"* URL handelt. Ebenso wurde das Hyperlink zu *"WebWatcher"* ausgelassen, da das System das Auswählen des Hyperlinks unterbindet, um rekursive Aufrufe abzufangen. Die Hyperlinks *"School of Computer Science", "SCS Organizations", "SCS Departments, Centers, and Institutes", "SCS Degree Programs and Courses" und "Affiliated Organization"*, sowie die Hyperlinks *"Local Web Resources" und "SCS Local Web home page"* zeigen jeweils auf die gleiche Seite. Sie werden nicht unterschieden, sondern als äquivalent betrachtet.

Im hier beschriebenen Experiment werden die folgenden Lernverfahren verglichen. Es folgt eine Auflistung der Lernverfahren mit den verwendeten Parametern. Die Parameterwerte wurden heuristisch optimiert.

- *Bayes*: Der naive Bayes'sche Klassifikator aus Abschnitt 3.4.2.
- *Nearest-Neighbor*: Der k-Nearest-Neighbor-Algorithmus wie in Abschnitt 3.4.3 beschrieben. Der Parameter k wird im folgenden auf 3 gesetzt.
- *Rocchio*: Der Rocchio-Relevance-Feedback-Algorithmus modifiziert zur Textkategorisierung (siehe Abschnitt 3.4.1). Es wurden $\alpha := 1$ und $\beta := 0.25$ benutzt.
- *Winnow*: Der Winnow-Algorithmus aus Abschnitt 3.4.4. Als Lernrate wurde $\gamma := 1.3$ gewählt und jedes Trainingsbeispiel wurde dem Algorithmus 40mal präsentiert.
- *WebWatcher*: WebWatchers ursprüngliche Lernmethode (siehe Abschnitt 2.5).

- *Popularität*: Bei dieser einfachen Lernstrategie werden Hyperlinks anhand ihrer Popularität beurteilt. Je häufiger ein Hyperlink von Benutzern gewählt wird, desto höher wird es bewertet. Im maschinellen Lernen wird eine solche Strategie normalerweise als "Default-Strategie" bezeichnet.

Zusätzlich werden noch zwei nicht lernende Verfahren zum Vergleich mit aufgeführt.

- *Match*: Das Benutzerinteresse und der unterstrichene Anchor-Text werden anhand ihrer TFIDF-Kosinusähnlichkeit verglichen. Je größer die Ähnlichkeit, desto höher ist die Bewertung des Hyperlinks.

- *Random*: Die Bewertung von Hyperlinks erfolgt zufällig. Diese Methode dient als Untergrenze für die zu erreichenden Performanz.

Alle Ergebnisse sind über 20 zufällige Aufteilungen in Test- und Trainingsdaten gemittelt. Auf jeweils 33.3% der zur Verfügung stehenden Beispiele wird getestet. Die restlichen Beispiele werden zum Training verwendet. Ähnlich wie in 2.5 beschrieben, wird der Anchor-Text des Hyperlinks als initiales Trainingsbeispiel benutzt. Die oben aufgeführten Klassifikatoren werden jeweils auf dieselben Test-/Trainingsmengen angewandt. Dadurch werden paarbasierte Signifikanztests einsetzbar. Hier wird ein Wilcoxon/Mann-Whitney-Rangtest (siehe [Kreyszig, 1975]) mit 95-prozentiger Konfidenz als Signifikanzmaß benutzt.

Um einen anschaulichen Eindruck von der Streuung und der Signifikanz der Ergebnisse zu vermitteln, wird zusätzlich ein Konfidenzintervall um den Mittelwert angegeben. Das Intervall bezeichnet jeweils den einfachen Standardfehler.

3.8 Ergebnisse

Die Resultate der Lernmethoden werden in diesem Abschnitt anhand von zwei Performanzmaßen vorgestellt und analysiert. Die verwendeten Maße sind mikrobewertete Precision/Recall-Graphen und eine modifizierte Variante von Accuracy, genannt $Accuracy(k)$. $Accuracy(k)$ ist an das in Abschnitt 2.6 eingeführte Maß angelehnt. $Accuracy(k)$ ist die Wahrscheinlichkeit, daß ein Algorithmus mindestens eines der Hyperlinks auf einer Seite vorhersagen kann, denen der Benutzer folgt. Hierzu ist es dem Algorithmus erlaubt, k Hyperlinks vorzuschlagen.

Tabelle 2 zeigt die erreichten $Accuracy(3)$. Jeder Algorithmus durfte drei der 18 Hyperlinks vorschlagen. Wird Stammformreduktion verwendet, erreicht der naive Bayes'sche Klassifikator *Bayes* mit 55.9% die höchste Genauigkeit. Die anderen Algorithmen folgen in der Tabelle absteigend sortiert. Die Abstände sind jeweils signifikant. Alle Lernalgorithmen liefern weit bessere Genauigkeiten als die nicht lernenden Ansätze *Match* und *Random*. Allerdings schneidet die sehr einfache Lernstrategie *Popularität* gut ab. Nur *Bayes* und *Nearest-Neighbor* erzielen eine bessere Accuracy. Die gute Genauigkeit von *Popularität* erklärt sich

	mit Stammformreduktion		ohne Stammformreduktion	
	Accuracy	Standardfehler	Accuracy	Standardfehler
Bayes	55.9%	0.8%	54.6%	0.8%
Nearest-Neighbor	51.8%	0.7%	51.1%	0.9%
Popularitaet	47.9%	0.8%	47.9%	0.8%
Rocchio	45.4%	0.8%	43.1%	1.0%
Winnow	40.5%	0.7%	38.8%	1.0%
WebWatcher	37.7%	0.8%	37.2%	1.1%
Match	24.1%	0.6%	22.8%	0.6%
Random	19.3%	0.9%	19.3%	0.9%

Tabelle 2: Gemittelte Accuracy(3) der Textkategorisierungsalgorithmen. Ein Algorithmus darf drei Hyperlinks vorschlagen. Die Prozentzahlen sind eine Schätzung der Wahrscheinlichkeit, daß der Benutzer einem der vorgeschlagenen Hyperlinks folgt.

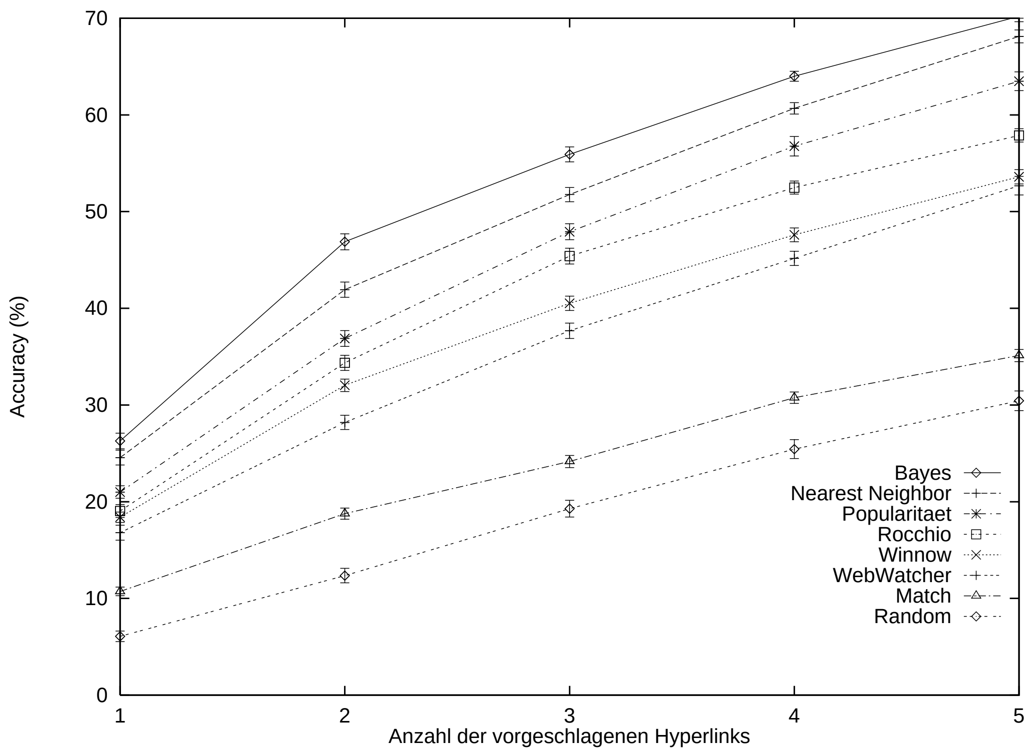

Abbildung 16: Accuracy in Abhängigkeit von der Anzahl der vorgeschlagenen Hyperlinks (mit Stammformreduktion).

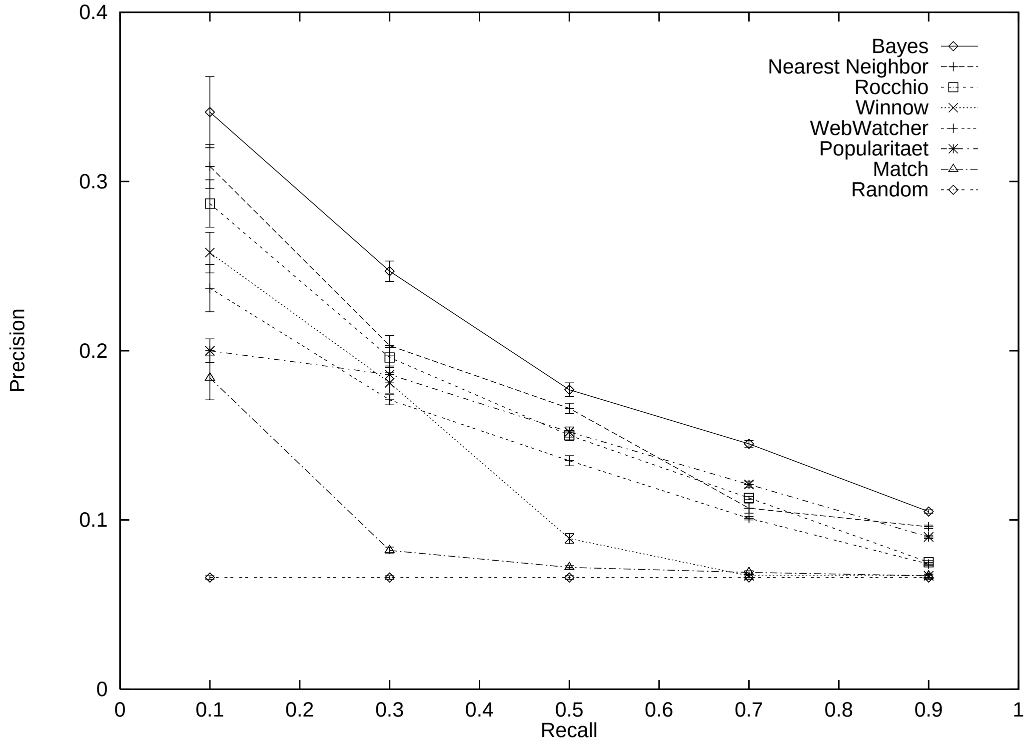

Abbildung 17: Precision/Recall-Graph (Microaveraging) (mit Stammformreduktion)

daraus, daß einige Hyperlinks wesentlich häufiger von Benutzern gewählt werden als andere. Dementsprechend schneiden die Algorithmen *Rocchio*, *Winnow* und *WebWatcher* schlechter ab, da sie die Popularität von Hyperlinks nicht explizit in ihre Vorhersage mit einbeziehen.

Vergleicht man die rechte und die linke Spalte von Tabelle 2 zeigt sich, daß unabhängig von der Methode (für *Bayes*, *Rocchio*, *Winnow* und *Match* signifikant) eine höhere Performanz erzielt wird, wenn bei der Repräsentation der Beispiele Stammformreduktion verwendet wird[14]. Deshalb werden im folgenden nur noch die Ergebnisse bei eingeschalteter Stammformreduktion betrachtet.

Abbildung 16 zeigt die $Accuracy(k)$ in Abhängigkeit von k. Die erste Spalte von Tabelle 2 entspricht den Werten des Graphen an der Stelle 3. Auch für andere Werte von k ergibt sich die gleiche Rangfolge der Methoden. Mit steigender Anzahl von vorgeschlagenen Hyperlinks steigt naturgemäß die Wahrscheinlichkeit, daß ein Benutzer einem der Links folgt. Wenn Hyperlinks zufällig vorgeschlagen werden, ergibt sich ein linearer Zusammenhang zwischen der Anzahl der vorgeschlagenen Hyperlinks und der $Accuracy(k)$. Dies wird auch ex-

[14] Natürlich ausgenommen die Strategien *Random* und *Popularität*, da diese den Text unberücksichtigt lassen.

perimentell an dem Verlauf des Graphen der Strategie *Random* deutlich. Die Graphen aller Lernmethoden weisen hingegen einen konvexen Verlauf auf. Für kleine k liegt ihre Steigung über der des Graphen von *Random*. Hieran ist zu erkennen, daß die Lernmethoden nicht nur das mit der höchsten Konfidenz bewertete Hyperlink sinnvoll auswählen können, sondern auch die folgenden Hyperlinks mit besserer Genauigkeit als zufällig vorschlagen.

Andere Aspekte der Daten und der verwendeten Algorithmen lassen sich an dem Precision/Recall-Graphen in Abbildung 17 ablesen. Der Graph wurde nach der PRR-Methode aus Abschnitt 3.5 berechnet. Auch hier ergibt sich eine ähnliche Anordnung der Methoden wie bei der Betrachtung der $Accuracy(k)$. An allen Recall-Punkten erreicht der naive Bayes'sche Klassifikator die signifikant höchste Precision. Die anderen Lernmethoden sind nur schwer einzuordnen, da sich ihre Graphen schneiden. Deutlich von den Lernmethoden abgesetzt ist die nicht lernende Methode *Match*. An ihrem Graphen ist abzulesen, daß diese Methode nur bei einer geringen Anzahl von Beispielen sinnvolle Vorhersagen machen kann. Schon für einen Recall von 0.3 hat sich ihr Graph dem von *Random* stark angenähert.

3.9 Schlußfolgerungen

Die experimentellen Ergebnisse lassen sich wie folgt zusammenfassen.

- Es ist möglich, aus der Beobachtung vorangegangener Benutzerinteraktionen zu lernen und mit dem gelernten Wissen Aktionen neuer Benutzer vorherzusagen. Die Methoden, die aus Benutzerinteraktionen lernen, erreichten deutlich höhere Performanz als ein uninformiertes Verfahren (*Random*) und ein nicht lernendes Verfahren (*Match*).

- Auch zwischen den Lernverfahren gibt es signifikante Performanzunterschiede. Zumindest in Bezug auf $Accuracy(k)$ liegen *Bayes* und *Nearest-Neighbor* über den Methoden *Rocchio*, *Winnow* und *WebWatcher*. Auffällig ist, daß auch die einfache Lernmethode *Popularität* gut abschneidet, obwohl bei ihr das vom Benutzer eingegebene Interesse unberücksichtigt bleibt. Es gibt also einige wenige Hyperlinks, denen Benutzer besonders häufig folgen. Die gute Performanz von *Bayes* und *Nearest-Neighbor* läßt sich dadurch erklären, daß diese Methoden sowohl die Popularität von Hyperlinks als auch das Interesse des Benutzers in ihre Vorhersagen einbeziehen. Die anderen Methoden nutzen jeweils nur eine der beiden Informationsquellen explizit.

- In der vorgestellten Domäne führt Stammformreduktion unabhängig von der verwendeten Kategorisierungsmethode zu meist signifikant besserer Performanz.

- Eine Limitierung des Textklassifikationsansatzes zur Tourenplanung ist, daß die Methoden auf Trainingsbeispiele angewiesen sind, die sich aus Benutzerinteraktionen ergeben. Begibt sich z. B. ein Benutzer mit WebWatcher auf eine Seite, für die keine Trainingsbeispiele zur Verfügung stehen, können die Lernmethoden keine bessere Genauigkeit als *Match* erreichen. Weiterhin existiert kein Qualitätskriterium für Touren als Ganzes, sondern es wird nur auf jeder Seite versucht, das Benutzerverhalten nachzuahmen. Wie diese Limitierungen zu beheben sind, ist der Gegenstand des folgenden Abschnitts.

4 Reinforcement Learning auf dem WWW

Wie sollte eine gute Tour durch das World Wide Web aussehen? Eine naheliegende Definition wäre die folgende:

> Finde eine Sequenz von Hyperlinks und WWW-Seiten, so daß die Menge der relevanten Information auf den Seiten der Tour maximiert wird, aber die Tour nicht unnötig lang ist.

Bei dem auf Textkategorisierung beruhenden Ansatz aus dem vorangegangenen Abschnitt wurde versucht, Benutzerverhalten nachzuahmen. Zur Fortsetzung der Tour wurden die Hyperlinks vorgeschlagen, denen der Benutzer am wahrscheinlichsten folgen wird. Im Gegensatz dazu wird bei dem nun vorgestellten Ansatz Wissen über die Hypertextstruktur hinter einem Hyperlink in Betracht gezogen. Es werden die Hyperlinks zur Fortsetzung der Tour identifiziert, die zu vielen relevanten Seiten führen. Das Ziel ist es, die Menge der relevanten Information auf einer Tour zu maximieren.

Formal läßt sich dies mit der folgenden Version von *OptimalLinks*? darstellen. Sie ermittelt unabhängig vom Benutzer auf jeder Seite $p \in Page$ die optimalen Hyperlinks in Bezug auf ein Interesse $i \in Interest$.

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

Die Funktion ist wie folgt zu interpretieren. Die optimalen Hyperlinks auf einer Seite sind die Hyperlinks, die zu Seiten führen, welche auf einem Pfad mit maximaler relevanter Information liegen, wenn die Tour von dort wiederum optimal fortgesetzt wird. Im folgenden wird gezeigt, wie *OptimalLinks*? mit Methoden des Reinforcement Learning approximiert werden kann.

Der Reinforcement-Learning-Ansatz hat zwei wesentliche Vorteile gegenüber den Methoden aus dem vorangegangenen Abschnitt.

1. Es wird ein Qualitätskriterium für ganze Touren verwendet, das explizit ist und mit algorithmischen Methoden maximiert werden kann. Bei dem Textkategorisierungsansatz wurde lediglich versucht, Benutzerverhalten nachzuahmen.

2. Ein Problem von lernenden Softwareagenten ist, daß sie zu Beginn ihres Einsatzes erst lernen müssen und den Benutzer noch nicht unterstützen können [Witten et al., 1996]. Bei dem Textkategorisierungsansatz werden Trainingsbeispiele benötigt, die sich aus Interaktionen des Benutzers mit dem System ergeben. Für Seiten, die zuvor noch kein Benutzer mit WebWatcher besucht hat, steht demnach kein gelerntes Wissen zur Verfügung. Der nun vorgestellte Ansatz zur Tourenplanung benötigt keine expliziten Trainingsbeispiele aus Benutzerinteraktionen, sondern bezieht seine Information implizit aus Dokumenten und der Hypertextstruktur.

Im folgenden wird eine Lernmethode basierend auf Reinforcement Learning vorgestellt, die für ein gegebenes Relevanzmaß und unter vereinfachenden

Annahmen anhand des oben vorgestellten Kriteriums optimale Touren durch das WWW lernt. Reinforcement Learning ist als Methode zum Planen von WebWatchers Touren besonders vielversprechend, da es sich bereits beim Planen von Pfaden in der realen Welt, z. B. bei Robotern [Lin, 1991] [Thrun, 1995] [Mahadevan und Connell, 1992], bewährt hat. Nach einem kurzen Überblick über Reinforcement Learning und dynamische Programmierung im Absatz 4.1, wird deren Anwendung auf das Problem der Tourenplanung im WWW vorgestellt (Absatz 4.2). Die Methode wird experimentell evaluiert (Absätze 4.3 und 4.4) und mit dem Textkategorisierungsansatz verglichen (Absatz 4.5).

4.1 Reinforcement Learning

Reinforcement Learning ist eine Methode, die es Agenten erlaubt, Handlungssequenzen und Strategien zu lernen. In bestimmten Situationen (Markov'schen Entscheidungsprozessen, siehe 4.1.2) können so optimale Kontrollstrategien automatisch gefunden werden. Mit Reinforcement Learning können Entscheidungsprozesse der folgenden Form bearbeitet werden. Ein Agent bewegt sich durch einen Zustandsraum. Um von einem Zustand in den nächsten zu gelangen, kann der Agent bestimmte Operationen ausführen. In jedem Zustand erhält der Agent eine gewisse Belohnung $R(s)$ (die auch null oder negativ sein kann). Das Ziel des Agenten ist es, eine Sequenz von Operationen zu finden, die seine Belohnung maximiert.

Ein anschauliches Beispiel für einen solchen Agenten ist z. B. ein Roboter, der Operationssequenzen zur Ausführung einer Aufgabe durchführen soll. Die Aufgabe könnte sein, möglichst effizient alle Cola-Dosen, die in einem Raum verstreut liegen, in einen Abfallbehälter zu befördern. Für jede abgelieferte Cola-Dose erhält der Roboter eine Belohnung. Der Zustand des Roboters ist seine jeweilige Position, Orientierung und ob er eine Cola-Dose bei sich hat. Die Operatorsequenzen bestehen aus Aktionen wie "Cola-Dose aufheben", "90 Grad nach links drehen" oder "einen Meter vorwärts gehen". Das Ziel ist es, eine Operatorsequenz zu finden, so daß möglichst schnell alle Cola-Dosen aufgesammelt werden.

Eine weitere Anwendung von Reinforcement Learning ist z. B. einen Computer das Backgammonspielen - ein bis dahin ungelöstes Problem - automatisch lernen zu lassen [Tesauro, 1992]. Die Zustände entsprechen bei diesem Problem den möglichen Positionen von Steinen auf dem Spielbrett. Die Operationen sind die erlaubten Spielzüge. Der Agent erhält dann eine Belohnung, wenn er ein Spiel gewonnen hat. Durch Reinforcement Learning ist es gelungen, einen Computer (durch wiederholtes Spielen gegen sich selbst) lernen zu lassen, Backgammon auf Weltklasseniveau zu spielen.

Die Idee des Reinforcement Learning ist sehr einfach. Angenommen der Agent hat eine Bewertungsfunktion $Q(s, a)$ für jede Operation a in jedem Zustand s gelernt, die angibt, wie hoch die zu erwartende discountierte Belohnung ist, wenn er diesen Operator anwendet und danach der optimalen Operator-

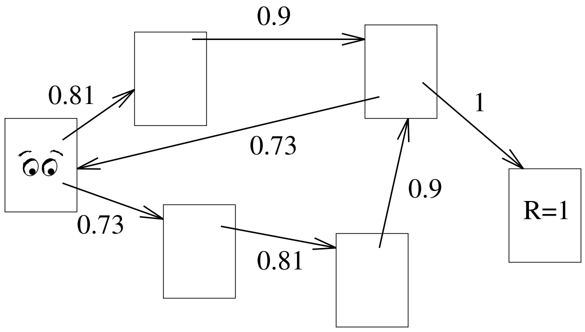

Abbildung 18: Zustandsraum eines einfachen Beispielproblems.

sequenz folgt. Die Discountierung von Belohnung erlaubt es, Belohnungen geringer zu bewerten, je weiter sie entfernt sind. Bei Kenntnis von $Q(s,a)$ wüßte der Agent in jedem Zustand, welchen Operator er jeweils als nächstes anwenden soll, um möglichst viel discountierte Belohnung auf seinem Weg durch den Zustandsraum zu erhalten. Formal ist $Q(s,a)$

$$Q(s,a) = R(s') + \sum_{i=1}^{\infty} \gamma^i \cdot R(s_i) \tag{42}$$

s ist der Zustand, in dem sich der Agent aktuell befindet. s' ist der Zustand, in den der Agent nach Ausführung von Aktion a gelangt. Die s_i sind die Zustände, die der Agent auf einer optimalen Operatorsequenz ausgehend von s' besucht. $0 < \gamma \leq 1$ ist der sogenannte Discount-Faktor. Durch γ kann gesteuert werden, wie stark weit entfernt liegende (nur durch eine lange Kette von Operationen erreichbare) Belohnungen bewertet werden. Je kleiner γ gewählt wird, desto mehr bevorzugt der Agent Belohnungen, die er schnell erreichen kann.

Bei den folgenden Beispielen wird die Funktion $Q(s,a)$ als bekannt vorausgesetzt. In Abschnitt 4.1.2 wird eine Methode der dynamischen Programmierung vorgestellt, welche die Approximation von $Q(s,a)$ ermöglicht.

4.1.1 Illustrative Beispiele

Abbildung 18 illustriert die Idee und das Ziel des Reinforcement Learning an einem Beispiel. Rechtecke repräsentieren mögliche Zustände. Die Pfeile stellen Operationen dar, die es dem Agenten erlauben, von einem Zustand in den nächsten zu gelangen. Die Pfeile sind jeweils mit dem zu ihnen gehörenden Wert $Q(s,a)$ beschriftet[15]. Im am weitesten rechts stehenden Zustand erhält der Agent eine Belohnung von 1. In allen anderen Zuständen erhält er keine Belohnung, d. h. die Belohnung ist 0. Um von jedem beliebigen Zustand auf dem schnellsten Weg zur Belohnung zu gelangen, muß der Agent jeweils die Operation wählen, für die $Q(s,a)$ maximal ist. Auf diese Weise maximiert der

[15] Hier mit Discount-Faktor $\gamma = 0.9$.

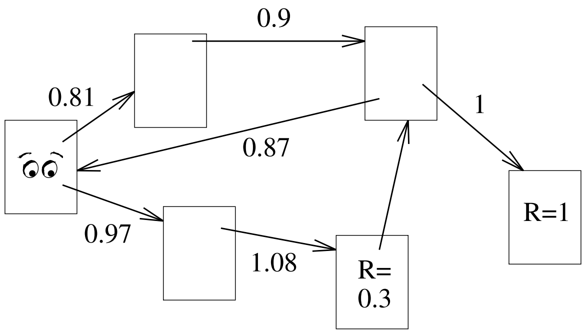

Abbildung 19: Zustandsraum eines Beispielproblems mit mehreren Zuständen, in denen der Agent eine Belohnung erhält.

Agent die discountierte Belohnung. Ausgehend vom Zustand mit den Augen, würde der Agent also in Abbildung 18 den "oberen" Weg zum Zustand mit der Belohnung wählen.

Das Beispiel in Abbildung 19 unterscheidet sich von dem vorangegangenen dadurch, daß es zwei Zustände gibt, in denen der Agent eine Belohnung erhält. Dadurch ändert sich die optimale Handlungsstrategie des Agenten. Ausgehend von dem am weitesten links stehenden Zustand wählt der Agent nun den "unteren" Weg. Dadurch erhält er in zwei Zuständen eine Belohnung.

4.1.2 Dynamische Programmierung: Die Berechnung von $Q(s, a)$

Unter relativ allgemeinen Bedingungen kann $Q(s, a)$ durch eine iterative Berechnungsvorschrift mittels *dynamischer Programmierung* [Bellman, 1957] approximiert werden. Dynamische Programmierung kann zur Lösung von Markov'schen Entscheidungsproblemen (MDP) verwendet werden. Das oben beschriebene Problem des Reinforcement Learning ist eine spezielle Art von Markov'schem Entscheidungsproblem. Hauptannahme von MDP ist, daß die in einem Zustand zu wählenden Aktionen nur von dem Zustand selbst abhängen, nicht aber von anderen bereits besuchten Zuständen. Ebenso hängt die in einem Zustand erhältliche Belohnung nur von dem Zustand selbst ab. Eine formale Definition dieser Problemklasse findet sich in [Barto et al., 1995].

Sehr allgemeine Bedingungen sind hinreichend für die Anwendbarkeit von dynamischer Programmierung zur Bestimmung der $Q(s, a)$. Es ist lediglich gefordert, daß $\gamma < 1$ und die Belohnung $R(s)$ in jedem Zustand s endlich ist[16]. Im folgenden wird der synchrone *Value-Iteration-Algorithmus* nach Bellman [Bellman, 1957] vorgestellt.

[16] Ist $\gamma = 1$ (d. h. ohne Discountierung), sind zusätzliche Annahmen erforderlich. Da Discountierung in der hier beschriebenen Anwendung des Tourenlernens allerdings erwünscht ist, wird auf diesen Spezialfall nicht weiter eingegangen.

Der Algorithmus startet mit einer beliebigen Schätzung von $Q(s,a)$, genannt $Q_0(s,a)$ (normalerweise $Q_0(s,a) := 0$). In mehreren Durchgängen werden dann jeweils alle Zustände s mit allen in ihnen zulässigen Operationen a in einer beliebigen Reihenfolge betrachtet. In jedem Durchgang wird der Funktionswert $Q_{k+1}(s,a)$ anhand der folgenden "Backup-Formel" [Bellman, 1957] aus dem vorangegangenen Durchgang k berechnet:

$$Q_{k+1}(s,a) := R(s') + \gamma \max_{a' \in Aktionen\ in\ s'} [Q_k(s',a')] \tag{43}$$

s' ist hierbei der Zustand, in den der Agent gelangt, wenn er im Zustand s die Operation a ausführt. $R(s')$ ist die Belohnung, die er in s' erhält und die $Q_k(s',a')$ sind die aktuellen geschätzten Werte der Q-Funktion für alle erlaubten Aktionen a' im Zustand s'. In [Bertsekas, 1987] wird gezeigt, daß jedes $Q_\infty(s,a)$ konvergiert, d. h.

$$\forall s,a : \lim_{k\to\infty} Q_k(s,a) \longrightarrow Q(s,a) \tag{44}$$

Als Konvergenzkriterium für $Q_k(s,a)$ kann die sog. Bellman-Gleichung herangezogen werden:

$$\forall s,a : Q(s,a) = R(s') + \gamma \max_{a' \in Aktionen\ in\ s'} [Q(s',a')] \tag{45}$$

Diese Gleichung ist genau für die in Formel (42) definierten $Q(s,a)$ erfüllt. Die Ähnlichkeit von (45) und (43) erlaubt das einfache Überprüfen des Konvergenzkriteriums. Die Bellman-Gleichung ist approximativ erfüllt, wenn bei einem Durchgang des Value-Iteration-Algorithmus die Summe der Unterschiede zwischen den $Q_{k+1}(s,a)$ und $Q_k(s,a)$ kleiner als ein Schwellwert wird (siehe z. B. [Barto et al., 1995]). Der Algorithmus stoppt dann und gibt $Q_{k+1}(s,a)$ als Approximation für $Q(s,a)$ aus.

4.1.3 Erweiterungen des Reinforcement Learning

Das oben dargestellte Problem ist eines der einfachsten Modelle des Reinforcement Learning. Zum einen existieren diverse Ansätze, das Modell zu erweitern, um so komplexere Probleme bearbeiten zu können. Zum anderen können mit verfeinerten Methoden der dynamischen Programmierung die Q-Funktion auch unter allgemeineren Bedingungen approximiert werden.

Eine häufig benötigte Erweiterung des Modells ist es, den Ausgang von Operationen nicht mehr als deterministisch, sondern als probabilistisch zu modellieren. Dies bedeutet, daß nach Anwendung von Operation a in Zustand s sich der Agent mit Wahrscheinlichkeit $\Pr(s'|s,a)$ im Zustand s' befindet. Der Zustand, in dem der Agent sich nach Ausführung der Operation befindet, wird also durch eine Wahrscheinlichkeitsverteilung bestimmt. Solche Situationen treten z. B. bei der Roboterkontrolle auf. Soll ein Roboter z. B. fünf Meter geradeaus fahren, wird seine Endposition nur ungefähr der berechneten Endposition entsprechen. Durch Bodenunebenheiten etc. wird die Abweichung mehr oder weniger groß sein. Weiterhin kann auch die Belohnungsfunktion $R(s)$ probabilistisch sein, d.

h. der Agent erhält die Belohnung nur mit einer gewissen Wahrscheinlichkeit. Probleme dieser Art können bei Kenntnis der Wahrscheinlichkeitsverteilungen mit dem oben beschriebenen Verfahren der sequentiellen Value Iteration gelöst werden.

Aufwendigere Verfahren werden bei nur partiell beobachtbaren Markov'schen Entscheidungsproblemen [Cassandra et al., 1994] notwendig. Bei diesen Problemen kann der Agent den Zustand, in dem er sich befindet, nicht direkt feststellen. Er kann nur die Wahrscheinlichkeit schätzen, daß er sich in einem bestimmten Zustand befindet[17].

Verbesserungen des sequentiellen Value-Iteration-Algorithmus erlauben zusätzlich zur Bearbeitung von komplexeren Problemen auch flexiblere Einsatzmöglichkeiten. Beim Gauß-Seidel-Algorithmus (siehe z. B. [Barto et al., 1995]) werden die $Q_{k+1}(s, a)$ zwar noch wie beim sequentiellen Algorithmus in Durchgängen neu berechnet, allerdings braucht die Version $Q_k(s, a)$ nicht gespeichert zu werden, was zu einer Speicherplatzersparnis führt. Beim asynchronen Value-Iteration-Algorithmus kann ganz auf die Organisation nach Durchgängen verzichtet werden.

Eine wesentliche Erweiterung stellen auch die sog. real-time Algorithmen wie real-time Q-Learning [Watkins, 1989] [Watkins und Dayan, 1992] oder Temporal Difference Learning [Sutton, 1984] [Sutton, 1988] dar. Diese Lernmethoden erlauben das inkrementelle Lernen von Handlungsstrategien durch Exploration. Ein Agent wird ohne oder mit nur unvollständigem Wissen über die zu lösende Aufgabe gestartet. Der Agent hat die Möglichkeit, den Zustandsraum durch Experimente zu erforschen. Zuerst sind die Aktionen des Agenten weitgehend zufällig. Inkrementell lernt der Agent dann, welche Aktionen sich als nützlich erweisen und bei welchen Aktionen keine Belohnung zu erwarten ist. Real-Time Reinforcement Learning integriert also Lernen mit der automatischen Exploration des Zustandsraumes. Ein solcher Algorithmus kam z. B. bei der oben beschriebenen Anwendung des Backgammonspielens zum Einsatz [Tesauro, 1992]. Der Agent experimentierte mit verschiedenen Spielzügen und war in der Lage, seine Spielstrategie inkrementell bis nahe dem Optimum zu verbessern.

4.1.4 Anwendungen von Reinforcement Learning

Bereits erwähnt wurde das erfolgreiche Lernen von Backgammonspielen durch Temporal Difference Learning [Tesauro, 1992]. Diese Arbeit basiert unter anderem auf den frühen Arbeiten von Samuel zum Brettspiel Dame [Samuel, 1959]. Die Ansätze unterscheiden sich unter anderem dadurch, daß Dame deterministisch ist, wohingegen bei Backgammon durch Würfeln zusätzliche Unsicherheiten behandelt werden müssen. Mit mäßigem Erfolg wurden ähnliche Methoden auch für Schach [Thrun, 1994] eingesetzt.

[17] Das bekannte Parkhaus Problem :-) Befinde ich mich zur Zeit auf Parkdeck 4, 5 oder 6 - alles sieht gleich aus.

Gerade in Verbindung mit Spieltheorie wird die Verwandtschaft von dynamischer Programmierung und Suche deutlich. Korf [Korf, 1990] stellt ein A^* Suchverfahren [Hart et al., 1968] vor, das die Funktion h der geschätzen Kosten bis zum Zielzustand mit Hilfe dynamischer Programmierung automatisch lernt und verbessert.

Vielfach wurde Reinforcement Learning auch zur Roboternavigation eingesetzt. Thrun [Thrun, 1995] beschreibt eine Aufgabe, bei der ein Roboter in einer Büroumgebung zu einem Zielobjekt fahren soll. Dabei müssen das Zielobjekt erkannt und Hindernisse umgangen werden. Ähnliche Aufgaben beschreiben auch Lin [Lin, 1991] und Mahadevan und Connell [Mahadevan und Connell, 1992].

Neuere Arbeiten beschäftigen sich mit dem Einsatz von Reinforcement Learning zur Optimierung von industriellen Produktionsprozessen [Mitchell, 1996].

4.2 Reinforcement Learning zur Tourenplanung im WWW

Es wird nun gezeigt, wie Reinforcement Learning auf das Problem der Tourenplanung auf dem World Wide Web angewandt werden kann. Das WWW läßt sich sehr anschaulich als gerichteter Graph betrachten. Dokumente entsprechen Knoten des Graphen und Hyperlinks sind gerichtete Verbindungen zwischen den Knoten. Bei dieser Betrachtungsweise liegt die Analogie zu den Beispielen aus Abschnitt 4.1.1 sehr nahe. Ein Agent auf dem WWW wandert durch eine Kollektion von Dokumenten (Zuständen). Von jedem Dokument stehen eine Menge von Hyperlinks (Aktionen) zur Auswahl, mit denen der Pfad fortgesetzt werden kann. Die Belohnung, die der Agent bei dem Besuch eines Dokuments erhält, richtet sich nach der Relevanz des Dokuments in Bezug auf das Benutzerinteresse. Es wird der Pfad durch das WWW gesucht, der die Menge der (discountierten) Belohnung maximiert.

4.2.1 Was wird gelernt?

Im Unterschied zu den Beispielen in 4.1.1, wo jeweils nur eine Q-Funktion gelernt wurde, wird nun ein eigenes $Q_w(s,a)$ für jedes Wort w gelernt (siehe Abbildung 20). Die jeweils zugehörige Belohnungsfunktion $R_w(s)$ ist der normalisierte TFIDF-Wert des Wortes w auf der Seite s. Die Belohnungsfunktion ist im folgenden Abschnitt genauer beschrieben. Bei dieser Problemdefinition lernt also jede Q-Funktion Wege durch das WWW zu finden, so daß die Summe der discountierten TFIDF-Werte der dabei besuchten Seiten maximiert wird.

Um zur Laufzeit Hyperlinks vorzuschlagen, kann WebWatcher die gelernten Q-Funktionen wie folgt benutzen. Wenn das vom Benutzer eingegebene Interesse aus nur einem Wort w besteht, werden die Hyperlinks anhand der entsprechenden Funktion $Q_w(s,a)$ geordnet. Besteht das Benutzerinteresse aus mehre-

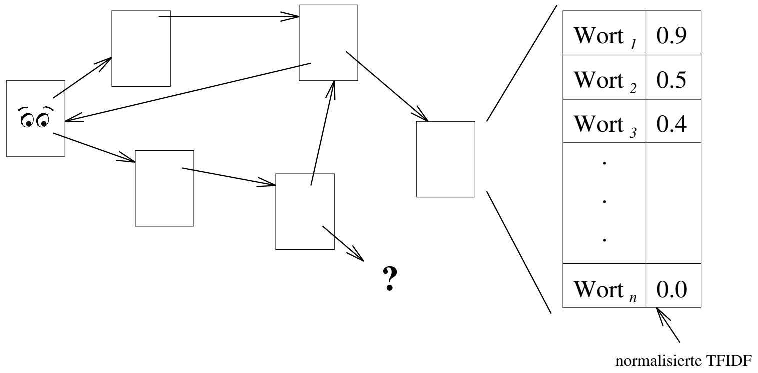

Abbildung 20: Für jedes Wort wird eine eigene Q-Funktion gelernt. Die Belohungsfunktion ist jeweils der normalisierte TFIDF-Wert des Wortes auf der betreffenden Seite.

ren Worten $w_1 ... w_n$, ist die Bewertungsfunktion die Summe[18] der Q-Funktionen $\sum_{i=1}^{n} Q_{w_i}(s,a)$. Die Hyperlinks mit der höchsten Bewertung sind die vielversprechendsten Fortsetzungen der Tour.

4.2.2 Beschreibung der Belohnungsfunktion

Wie kann die Relevanz eines Dokuments in Bezug auf das Benutzerinteresse gemessen werden? Ein weitverbreiteter Ansatz hierzu ist das bereits oben mehrfach erwähnte Vektorraum-Retrieval-Modell mit der TFIDF-Gewichtung [Salton, 1991]. Dokumente werden durch Wortvektoren repräsentiert. Die Länge eines jeden Dokumentvektors sei auf 1 normiert. Die Elemente des Vektors geben dann ein heuristisches Maß dafür an, wieviel Information das Dokument über das betreffende Wort enthält. Genau diese Gewichte werden im folgenden als Belohnungsfunktion dienen. $R_w(s)$ berechnet sich also als

$$R_w(s) = \frac{TF(w,s) \cdot IDF(w)}{\sqrt{\sum_{w'} (TF(w',s) \cdot IDF(w'))^2}} \tag{46}$$

Die Belohnung hängt also ausschließlich vom Inhalt des jeweiligen Dokuments ab und nicht z. B. davon, welche Dokumente der Benutzer bereits gelesen hat. Diese Unabhängigkeit wird bei der Modellierung durch Reinforcement Learning vorausgesetzt. Allerdings können Abhängigkeiten dadurch modelliert werden, daß man nicht von jedem Zustand in jeden beliebigen anderen Zustand gelangen kann. Gerade bei Hypertext ist dies besonders ausgeprägt. Durch Hyperlinks soll der Benutzer direkt zu semantisch verwandten Dokumenten gelangen, wodurch Abhängigkeiten zwischen Dokumenten modelliert werden.

[18] In Information-Retrieval-Terminologie entspricht dies einer Anordnung mit dem Skalarprodukt als Ähnlichkeitsmaß, den $Q_w(s,a)$ als Dokumentgewichten und der Term Frequency als Fragegewichtung.

4.2.3 Funktionsapproximation für unbekannte Hyperlinks

Da nicht zu erwarten ist, daß Benutzer immer genau die gleichen Seiten besuchen, wird WebWatcher häufig Dokumente und Hyperlinks behandeln müssen, die das System zuvor noch nicht gesehen hat. In solchen Fällen müssen Q-Werte approximiert werden. Das System muß also unbekannte Hyperlinks und Dokumente anhand von bekannten bewerten können, d. h. es muß generalisieren können.

Im folgenden wird ein k-Nearest-Neighbor-Funktionsapproximator [Mitchell, 1997] zur Approximation der Q-Werte $Q_w(s,a)$ von unbekannten Hyperlinks a benutzt. In Vergleich zu anderen Funktionsapproximatoren, wie den häufig verwendeten neuronalen Netzen (z. B. [Tesauro, 1992]), hat er theoretische Vorteile [Gordon, 1995] und ist erfahrungsgemäß relativ stabil in Bezug auf kleine Mengen von Trainingsbeispielen und viele (spärlich besetzte und relevante) Attribute.

Der k-Nearest-Neighbor-Funktionsapproximator ist sehr ähnlich zu dem in Abschnitt 3.4.3 vorgestellten k-Nearest-Neighbor-Klassifikator. Anstelle der Klassenzugehörigkeiten werden bei der Funktionsapproximation die Funktionswerte der k ähnlichsten Trainingsbeispiele betrachtet. Der approximierte Funktionswert $\tilde{Q}(s,a)$ ist der anhand der Ähnlichkeit $sim()$ (wie im folgenden definiert) der k ähnlichsten Trainingsbeispiele gewichtete Mittelwert.

$$\tilde{Q}(s,a) = \frac{1}{k} \cdot \sum_{i=1}^{k} sim((s,a),(s_i,a_i)) \cdot Q(s_i,a_i) \tag{47}$$

Hyperlinks a werden durch die TFIDF-Vektorraum-Repräsentation $\vec{a}$ ihres unterstrichenen Anchor-Textes beschrieben. Analog werden Dokumente s durch die TFIDF-Vektorraum-Repräsentation $\vec{s}$ ihres Titels[19] dargestellt. Um die Ähnlichkeit zwischen einem Hyperlink a_1 auf Seite s_1 und einem Hyperlink a_2 auf Seite s_2 zu bestimmen, wird zunächst die Ähnlichkeit von a_1 mit a_2 sowie s_1 mit s_2 berechnet. Das verwendete Ähnlichkeitsmaß ist, wie bereits in Abschnitt 3.4.3 beschrieben, der Kosinus. Die Gesamtähnlichkeit ergibt sich aus der Ähnlichkeit der Dokumentbeschreibungen von s_1 und s_2 plus zweimal[20] der Ähnlichkeit der Hyperlinkbeschreibungen von a_1 und a_2.

$$sim((s_1,a_1),(s_2,a_2)) = \frac{1}{3} \cdot [\cos(\angle(\vec{s_1},\vec{s_2})) + 2 \cdot \cos(\angle(\vec{a_1},\vec{a_2}))] \tag{48}$$

4.3 Aufbau des Experiments

In dem folgenden Experiment soll untersucht werden, ob es durch Reinforcement Learning möglich ist, gute Touren zu konstruieren. Die von Benutzern gewählten

[19] Der Titel ist der Text, welcher durch die HTML-Tags `<TITLE>` und `</TITLE>` eingeschlossen wird.

[20] Diese Gewichtung basiert auf der Heuristik, daß der Anchor-Text die größere Aussagekraft in Bezug auf die Semantik des Hyperlinks hat.

Pfade werden wieder zur Evaluation herangezogen und als "optimal"[21] betrachtet. Analog zu dem Experiment mit den Textkategorisierungmethoden wird nun gemessen, wie gut mit Reinforcement Learning vorhergesagt werden kann, welchen Hyperlinks der Benutzer folgen wird. Im Gegensatz zu dem Experiment aus Abschnitt 3.7 wird hier nicht nur auf einer speziellen Seite evaluiert, sondern auf allen Seiten, die im Laufe von Touren besucht wurden. Auf diese Weise kann die Performanz der Lernmethode auch auf weniger häufig besuchten (oder gänzlich unbekannten) Seiten untersucht werden.

Bei den im folgenden präsentierten Experimenten wurde aus Effizienzgründen eine Teilmenge der Touren verwendet, die WebWatcher vom 2. August 1995 bis zum 4. März 1996 von der SCS-Front-Door-Seite (siehe Abb. 15) gegeben hat. Nur die Touren werden betrachtet, bei denen der Benutzer ein Interesse eingegeben hat und die mindestens vier Hyperlinks lang sind. Insgesamt genügen 1777 Touren diesen Kriterien. Fünf verschiedene Test/Trainings-Splits wurden benutzt, wobei jeweils 66.7% der Touren zur Auswahl der Trainingsbeispiele und 33.3% zum Testen verwendet wurden. Um die Experimente effizienter durchführbar zu machen, werden Q-Funktionen $Q_w(s,a)$ nur für die Worte w gelernt, die in den Interessenbeschreibungen der Touren in den Testbeispielen vorkommen.

Durchschnittlich ergeben sich jeweils 1380 Testbeipiele aus den Testtouren. Um die Menge der Trainingsbeispiele klar zu definieren, werden die Q-Funktionen nur auf den Hyperlinks trainiert, denen mindestens ein Benutzer während der Trainingstouren gefolgt ist. Wurden also z. B. 4 Hyperlinks auf einer Seite mit 30 Hyperlinks in den Trainingsdaten mindestens einmal besucht, so wird nur auf diesen 4 Hyperlinks trainiert. Die restlichen 26 Hyperlinks werden approximiert. Dies bedeutet, daß keine zusätzliche Exploration von neuen WWW Seiten stattfindet, obwohl dies prinzipiell machbar und wahrscheinlich vorteilhaft wäre.

Neben dem Ergebnis des Reinforcement-Learning-Ansatzes *RL* werden wieder die zwei einfachen Strategien *Random* und *Match* aufgeführt.

- *RL*: Reinforcement-Learning-Methode wie oben beschrieben mit $\gamma := 0.6$. Die Methode ist dahingehend erweitert, daß zum Zeitpunkt der Bewertung von Hyperlinks zusätzlich eine "unmittelbare Belohnung" zu den Q-Funktionen addiert wird. Die Höhe dieser Belohnung entspricht der Kosinusähnlichkeit von Anchor-Text und Benutzerinteresse, wie sie von *Match* berechnet wird.

- *Match*: Hyperlinks werden anhand der Ähnlichkeit ihres Anchor-Textes mit dem Benutzerinteresse ausgewählt. Es wird die TFIDF-Vektorraumdarstellung und der Kosinus als Ähnlichkeitsmaß benutzt.

- *Random*: Es werden Hyperlinks gleichverteilt durch eine Zufallsstrategie ausgewählt.

[21]... was in der Realität wahrscheinlich nicht zutrifft.

	Accuracy(3) (auf allen Seiten)	Accuracy(3) (nur trainierte Seiten)	Accuracy(3) (nur unbekannte Seiten)
RL	44.9% ±0.5%	41.8% ±0.4%	48.1% ±0.5%
Match	40.2% ±0.6%	33.0% ±0.4%	47.5% ±0.8%
Random	31.6% ±0.4%	22.5% ±0.4%	41.1% ±0.9%

Tabelle 3: Resultate für den Reinforcement-Learning-Ansatz zur Tourenplanung. Die Resultate sind über fünf verschiedene Aufteilungen in Test- und Trainingsmenge gemittelt. 49% der Testbeispiele befinden sich auf unbekannten Seiten. Dies sind Seiten, die nicht in der Trainingsmenge sind.

4.4 Ergebnisse

Tabelle 3 zeigt die erreichte *Accuracy*(3) für den Reinforcement-Learning-Ansatz *RL* und die Methoden *Match* und *Random*. Alle Werte wurden anhand einer Mikrobewertungsstrategie ermittelt. Dies bedeutet, daß gleichgewichtet über alle Beispiele auf allen Seiten gemittelt wird.

Die erste Spalte von Tabelle 3 enthält die *Accuracy*(3) gemittelt über alle in den Testdaten auftretenden Seiten. Mit 44.9% erreicht *RL* die signifikant höchste *Accuracy*(3) vor *Match* mit 40.2% und *Random* mit 31.6%. Deutlicher wird der Vorsprung der Reinforcement-Learning-Methode, betrachtet man nur Seiten, auf denen der Lerner auch trainiert wurde. Wie in dem vorangegangenen Abschnitt beschrieben, wird in diesem Experiment ein Hyperlink genau dann in die Trainingsmenge aufgenommen, wenn mindestens ein Benutzer ihm während einer Trainingstour gefolgt ist. Seiten mit mindestens einem solchen Hyperlink werden als "trainiert", ansonsten als "unbekannt" bezeichnet. Durchschnittlich sind 49% der Beispiele auf unbekannten Seiten. Für Beispiele auf trainierten Seiten erreicht *RL* eine Genauigkeit von 41.8% und *Match* 33.0%. Auf unbekannten Seiten ist der Unterschied zwischen *RL* und *Match* nicht signifikant. Es ist zu beachten, daß auch auf trainierten Seiten in der Regel viele Hyperlinks unbekannt sind. Für diese Hyperlinks muß $Q(s,a)$ wie auf unbekannten Seiten approximiert werden.

Stellt man die Ergebnisse für trainierte und unbekannte Seiten gegenüber, so fällt auf, daß paradoxerweise alle Methoden auf unbekannten Seiten eine höhere *Accuracy*(3) erzielen als auf trainierten. Dies erklärt sich dadurch, daß die durchschnittliche Anzahl von Hyperlinks auf trainierten Seiten mit 85 (Median 25) höher ist, als auf unbekannten Seiten mit 32 (Median 14). Die durchschnittliche Anzahl von Hyperlinks, denen ein Benutzer folgt, ist mit ca. 1.5 aber gleich. Dies bestätigen auch die unterschiedlichen Werte der Zufallsstrategie *Random* für trainierte und unbekannte Seiten.

Abbildung 21 zeigt die *Accuracy*(k) in Abhängigkeit von k. Die Werte aus der zweiten Spalte der obigen Tabelle 3 finden sich in dem Graphen an der

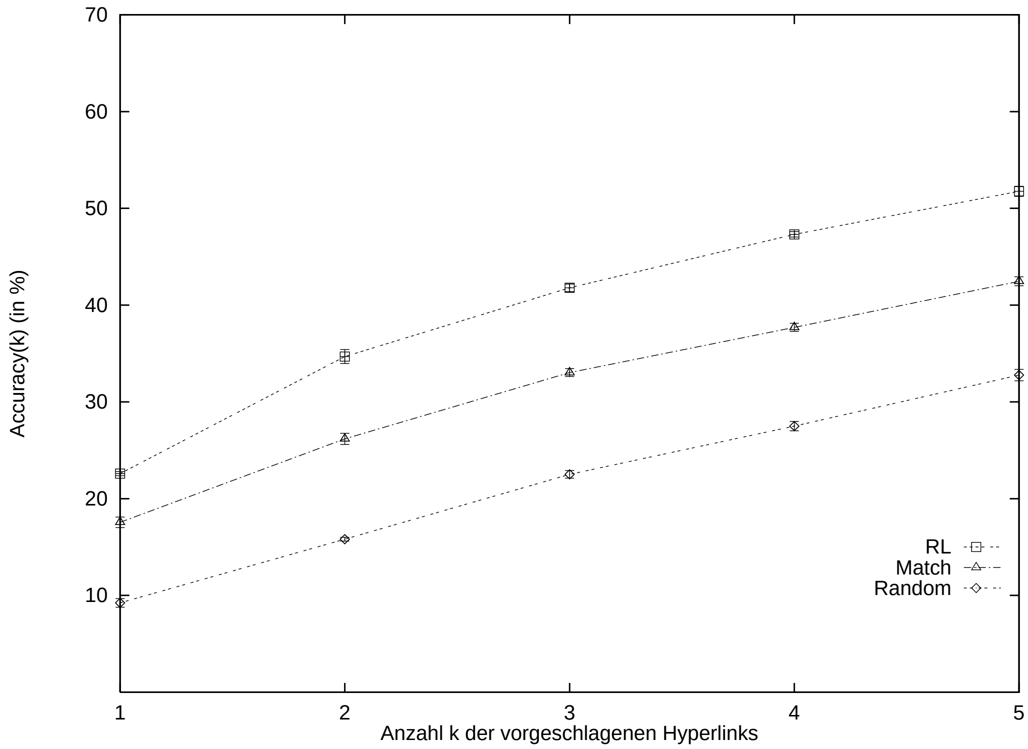

Abbildung 21: *Accuracy*(k) in Abhängigkeit von der Anzahl der vorgeschlagenen Hyperlinks k. Es werden nur trainierte Seiten betrachtet.

Stelle $k = 3$ wieder. Der Graph zeigt, daß auch für andere Werte von k die Performanz des Reinforcement-Learning-Ansatzes über der des einfachen Ansatzes *Match* liegt.

Eine Darstellung des Ergebnisses in Form von Precision/Recall-Graphen ist bei diesem Experiment nicht sinnvoll und wird deshalb weggelassen. Das Problem ist, daß bei diesem Experiment Precision nicht nur über Kategorien (Hyperlinks), sondern auch über verschiedene Seiten gemittelt werden müßte. Die erreichte Precision ist aber ganz wesentlich von der Anzahl der Hyperlinks auf der Seite abhängig, die sehr unterschiedlich sein kann. Es ist z. B. wesentlich einfacher, Vorhersagen auf einer Seite mit 5 Hyperlinks zu machen, als auf einer Seite mit 500. Der Precision/Recall-Graph würde also hauptsächlich die Seitenlängen widerspiegeln.

4.5 Vergleich mit Textkategorisierungsansätzen

Tabelle 4 enthält zusätzlich zu der Performanz des Reinforcement-Learning-Ansatzes *RL* die Ergebnisse mit der Methode *Bayes* (siehe Abschnitt 3.4.2). Gemittelt über allen Seiten erzielt der naive Bayes'sche Klassifikator mit 47.8%

	Accuracy(3) (auf allen Seiten)	Accuracy(3) (nur trainierte Seiten)	Accuracy(3) (nur unbekannte Seiten)
Bayes	47.8% ±0.4%	49.0% ±0.5%	46.5% ±0.6%
RL	44.9% ±0.5%	41.8% ±0.4%	48.1% ±0.5%
Match	40.2% ±0.6%	33.0% ±0.4%	47.5% ±0.8%
Random	31.6% ±0.4%	22.5% ±0.4%	41.1% ±0.9%

Tabelle 4: Der Reinforcement-Learning-Ansatz im Vergleich zum besten Textkategorisierungsansatz *Bayes*.

eine signifikant bessere Genauigkeit als *RL* mit 44.9%. Wie die zweite und dritte Spalte in Tabelle 4 zeigen, entsteht dieser Performanzgewinn auf den Seiten, auf denen Trainingsbeispiele aus vergangenen Benutzerinteraktionen zur Verfügung stehen. Die hier gezeigten Ergebnisse sprechen also dafür, den Bayes'schen Klassifikator zu verwenden, wenn genügend Trainingsbeispiele für eine Seite vorhanden sind.

Es ist allerdings zu beachten, daß *Bayes* durch die verwendete Evaluationsmethode einen Vorteil besitzt. *Bayes* wird darauf trainiert vorherzusagen, welchem (möglicherweise suboptimalen) Hyperlink der Benutzer folgen wird. Demgegenüber versucht *RL* zu bestimmen, welchem Hyperlink der Benutzer folgen sollte. Beide werden aber anhand der tatsächlichen Aktionen des Benutzers evaluiert.

Ohne Trainingsbeispiele ist mit dem Bayes'schen Klassifikator keine höhere Genauigkeit erreichbar als mit der Methode *Match*. Dies ist aufgrund der Konzeption des Algorithmus zu erwarten und wird durch die Ergebnisse in der dritten Spalte von Tabelle 4 bestätigt. Der Reinforcement-Learning-Ansatz ist hingegen nicht auf benutzergenerierte Beispiele angewiesen. Nur durch den experimentellen Aufbau fallen hier Seiten mit benutzergenerierten Trainingsbeispielen und Seiten, auf denen die Q-Funktionen trainiert wurden, zusammen. Es ist also denkbar, vor dem Einsatz eine große Anzahl von WWW-Seiten automatisch zu erforschen und die Q-Funktionen hierauf zu trainieren. Speziell zu Beginn des Einsatzes ist die Reinforcement-Learning-Methode also von Vorteil.

4.6 Schlußfolgerungen

Aus den obigen Experimenten können die folgenden Schlußfolgerungen gezogen werden:

- Durch Reinforcement Learning können Benutzeraktionen besser vorhergesagt werden als durch einfachen Vergleich mit dem Anchor-Text (*Match*). Die *RL* Methode erlaubt es, Touren ganzheitlicher zu planen und Information über Seiten, die mehrere Hyperlink Schritte entfernt sind, in die Entscheidung mit einfließen zu lassen.

- Die *RL* Methode ist nicht auf benutzergenerierte Beispiele angewiesen, sondern lernt aus der Hypertextstruktur. Die Methode kann also speziell in der Einführungsphase eines Systems nützlich sein und die Zeit überbrücken, in der noch keine benutzergenerierten Beispiele zur Verfügung stehen. Sind genügend benutzergenerierte Trainingsbeispiele vorhanden, ist die Methode *Bayes* vorzuziehen.

- Der erzielte Performanzgewinn ist auf den Seiten, auf denen die Q-Funktionen trainiert wurden, wesentlich größer als auf unbekannten Seiten, wo er kaum praktische Relevanz haben dürfte. Allerdings definieren sich bekannte Seiten nicht notwendigerweise dadurch, daß ein Benutzer diese Seite einmal besucht hat. Es ist denkbar, einen Agenten Gebiete des WWW autonom explorieren zu lassen, um so den Anteil der bekannten Seiten zu erhöhen und auch mehr Trainingsbeispiele für die Generalisierung auf unbekannten Seiten zu sammeln.

5 Experiment mit Menschen

Um ein besseres Gefühl für die Natur der Lernaufgabe zu bekommen, wird abschließend ein Experiment mit Menschen vorgestellt. Hierzu kommen wir noch einmal zu dem Datensatz und der Problemstellung aus Abschnitt 3 zurück, wo die SCS-Front-Door-Seite isoliert betrachtet wurde. Acht Testpersonen wird die gleiche Aufgabe gestellt, wie sie oben von den Textkategorisierungsalgorithmen zu lösen war. Anhand des vom Benutzer eingegebenen Interesses sollen die Testpersonen die drei Hyperlinks vorhersagen, denen der Benutzer am wahrscheinlichsten folgen wird. Jede Testperson erhielt 15 zufällig aus der Beispielmenge ausgewählte Beispiele und mußte jeweils genau drei der 18 Hyperlinks markieren.

Es ist zu beachten, daß dieses Experiment keinesfalls den Anspruch erhebt, psychologischen Standards zu genügen. So entspricht die Auswahl der acht Testpersonen nicht einer repräsentativen Stichprobe. Alle Testpersonen waren Studenten oder Professoren der Informatik. Sie waren mit der SCS-Front-Door-Seite und den angrenzenden Seiten vertraut. Auch ist die Anzahl der Testpersonen relativ gering, so daß die Ergebnisse zufälligen Schwankungen unterliegen können. Aus diesen Gründen ist der Ergebniswert nur begrenzt aussagekräftig und mit großen Unsicherheiten behaftet. Dennoch liefert seine Größenordnung eine gute Vorstellung von der Natur der Aufgabe.

Wie in Tabelle 5 gezeigt, erreichten die Testpersonen eine mittlere $Accuracy(3)$ von 47.5% (95%-Konfidenzradius 8.9%). Dieser Wert liegt in der gleichen Größenordnung wie die Ergebnisse der Textkategorisierungsalgorithmen. Es handelt sich bei der Aufgabe also nicht um ein Problem, das von Menschen durch ihr Sprach- und Weltwissen mit Leichtigkeit zu lösen ist. Dies legt den Schluß nahe, daß die Aufgabe des Vorhersagens, welchem Hyperlink der Benutzer folgen wird, in der hier beschriebenen Form relativ schwierig ist. Anhand des vom Benutzer eingegebenen Interesses kann nur mit Unsicherheit

	Accuracy(3)	Standardfehler
Bayes	55.9%	0.8%
Nearest-Neighbor	51.8%	0.7%
Popularitaet	47.9%	0.8%
Menschen	**47.5%**	**4.6%**
Rocchio	45.4%	0.8%
Winnow	40.5%	0.7%
WebWatcher	37.7%	0.8%
Match	24.1%	0.6%
Random	19.3%	0.9%

Tabelle 5: Gemittelte *Accuracy*(3) der Textkategorisierungsalgorithmen im Vergleich mit der Performanz von Menschen. Es ist zu beachten, daß es sich bei dem Standardfehler für die Performanz von Menschen um den einer Stichprobe handelt und nicht, wie bei den anderen Methoden, um den Standardfehler über einer Reihe von Stichproben.

auf die Hyperlinks geschlossen werden, die der Benutzer wählen wird.

Folgende Schlußfolgerungen lassen sich aus dem Experiment ziehen:

- Die Information aus der Interessenbeschreibung reicht in vielen Fällen nicht aus, um eindeutig die Hyperlinks vorherzusagen, denen der Benutzer folgen wird.

- Die Performanz der Lernalgorithmen ist nur begrenzt auf mangelndes Weltwissen zurückzuführen, da auch Menschen nicht wesentlich bessere Genauigkeit erzielen.

- Um eine höhere Performanz zu erreichen, erscheint die Verfeinerung der Information über Benutzerziele und die Erschließung von neuen Informationsquellen am vielversprechendsten. Eine mögliche Informationsquelle könnte ein Benutzermodell sein, das aus vorangegangenen Touren erlernt werden könnte. Die Möglichkeit zum interaktiven Feedback und die Einbeziehung der bereits auf der Tour besuchten Seiten kann zur weiteren Disambiguierung des Benutzerinteresses beitragen.

6 Zusammenfassung

Motiviert durch WebWatcher, einen Agenten, der das Browsing auf dem World Wide Web unterstützt, wurde die Frage untersucht, ob und wie das von WebWatcher benötigte Wissen zur Tourenplanung automatisch gelernt werden kann. Hierzu wurden zwei Ansätze untersucht. Der eine basiert auf Lernmethoden aus dem Bereich der Textkategorisierung. Der andere betrachtet Touren als Markov'sche Entscheidungsprozesse und benutzt Methoden des Reinforcement

Learning und der dynamischen Programmierung.

Die Fragen aus der Einleitung können durch die hier gezeigten Überlegungen und Experimente wie folgt beantwortet werden:

1. *Wie kann WebWatcher Wissen für die Tourenplanung durch Beobachtung von Benutzern lernen?*
   Es wurde gezeigt, wie die Entscheidung darüber, mit welchem Hyperlink eine Tour fortgesetzt werden soll, als Textkategorisierungsproblem modelliert werden kann. Die Trainingsbeispiele bei diesem Ansatz bestehen aus den Aktionen, die vorangegangene Benutzer von WebWatcher durchgeführt haben. Verschiedene Textkategorisierungsmethoden wurden experimentell verglichen. Der Bayes'sche Klassifikator erreichte die höchste Performanz und übertrifft die bislang von WebWatcher eingesetzte ad-hoc Lernmethode erheblich.

2. *Wie kann Wissen für die Tourenplanung aus Dokumenten und der Hypertextstruktur gelernt werden?*
   Methoden des Reinforcement Learning eröffnen die Möglichkeit, Touren auf dem WWW als Entscheidungsprozesse zu modellieren. Es wurde gezeigt, wie Hypertextdokumente auf Zustände und Hyperlinks auf Aktionen in einem Markov'schen Entscheidungsprozeß abgebildet werden können. Ein auf Information Retrieval Methoden beruhendes Relevanzmaß bildet die Basis dafür, mit Hilfe von dynamischer Programmierung Touren zu berechnen, die möglichst kompakt sind und viel relevante Information beinhalten. Da eine explizite Modellierung des gesamten Zustandsraumes aus Effizienzgründen ausscheidet, wurde gezeigt, wie ein Funktionsapproximator zur Generalisierung eingesetzt werden kann. Experimente belegen, daß mit dieser Methode Benutzeraktionen vorhergesagt werden können. Vorteilhaft gegenüber dem Textkategorisierungsansatz ist, daß zum Lernen keine benutzergenerierten Beispiele benötigt werden. Dadurch ist dieser Ansatz immer dann von Vorteil, wenn für eine Seite noch keine Beispiele existieren. Sind genügend Beispiele vorhanden, erreicht man mit dem Textkategorisierungsansatz - speziell dem Bayes'schen Klassifikator - allerdings eine höhere Performanz.

Durch das Experiment mit Menschen wurde abschließend untersucht, wo sich Verbesserungsmöglichkeiten bieten. Die Performanz von Menschen lag in der gleichen Größenordnung wie die der maschinellen Verfahren. Daraus läßt sich schließen, daß fehlendes Welt- und Hintergrundwissen nicht eine der Hauptlimitierungen der Lernalgorithmen ist. Verbesserungsmöglichkeiten bieten sich eher im Bereich der Erschließung von neuen Informationsquellen über das Benutzerinteresse. Benutzermodelle und Möglichkeiten zur Verarbeitung von während der Tour gegebenem Feedback erscheinen vielversprechend.

Inwieweit die interessenbezogene Sicht auf Hypertextdokumente, welche WebWatcher dem Benutzer mit Hilfe der Lernmethoden bietet, die Effektivität von Informationssuche durch Browsing erhöht, kann hier nicht abschließend geklärt werden. Ebenso bleibt die Frage, wie hoch die Genauigkeit von Lernalgorithmen für einen effektiven Einsatz sein muß, ungeklärt. Zu ihrer Beantwortung sind umfassende Studien mit Benutzern notwendig, die nicht Gegenstand dieser Arbeit sind. Experimente, welche die Effektivität von WWW-Agenten messen sollen, werden zur Zeit von Lara Ashmore [Ashmore] vorbereitet.

7 Perspektiven

Die in dieser Arbeit im Rahmen von WebWatcher untersuchten Methoden lassen sich auch auf andere Anwendungsgebiete übertragen. Im folgenden werden zwei weitere Einsatzbereiche aufgezeigt. Weiterhin werden Möglichkeiten beschrieben, wie sich die vorgestellten Methoden erweitern und verfeinern lassen.

- *Integration und Kombination von Textkategorisierung und Reinforcement Learning:* Natürlich stellt sich die Frage, wie beide Ansätze integriert werden können. Denkbar ist ein Multistrategieansatz, bei dem die Vorschläge von beiden Methoden zur Laufzeit integriert werden. Ein anderer Ansatz wäre, das Feedback von Benutzern direkt in die Belohnungsfunktion des Reinforcement-Learning-Ansatzes einfließen zu lassen.

- *Dialog Tour Guide ⇔ Benutzer:* In der aktuellen Version von WebWatcher ist der Dialog zwischen Benutzer und System stark eingeschränkt. Ergebnisse aus der Sprachverarbeitung und aus der Forschung über Intelligente Agenten (siehe z. B. [Sengers, 1996]) könnten hier für eine bessere Modellierung eingesetzt werden.

- *Intelligente Tour Guides in der realen Welt:* An der Carnegie Mellon University und an der Universität Bonn laufen zur Zeit Projekte zum Einsatz von Robotern als Tour Guides in Museen. Ähnliche Methoden, wie sie hier für das WWW vorgestellt wurden, könnten auch in dieser Domäne eingesetzt werden.

- *Intelligente WWW-Server:* Angenommen mit jeder Anfrage an einen WWW Server würde zusätzlich zum URL der gewünschten Seite Information über die Interessen des Benutzers mitgeschickt[22]. Solche Information könnten z. B. die generellen Interessen des Benutzers, aber auch die letzten drei Fehlermeldungen des Java-Interpreters sein, bevor der Benutzer sich zur Home-Page von SUN begibt. Wie könnte ein intelligenter WWW-Server diese Information benutzen? Die von WebWatcher zur Verfügung gestellten Funktionen, wie das Hervorheben von Hyperlinks und das Finden von ähnlichen Seiten, könnten auch direkt in einen intelligenten Server integriert werden. Dokumente könnten in vieler Hinsicht an die Interessen des jeweiligen Benutzers angepaßt werden.

[22] Probleme in Bezug auf Datenschutz seien vorerst vernachlässigt.

8 Danksagungen

Mein besonderer Dank geht an Prof. Tom Mitchell und Dayne Freitag für die sehr interessante Zeit unserer Zusammenarbeit beim Entwurf von WebWatcher. Für ihre Mitwirkung hierbei möchte ich mich auch bei David Zabowski, Robert Armstrong, Ada Lee und Shannon Mitchell bedanken.

Ganz herzlichen Dank für die vielen Kommentare zu der Ausarbeitung dieser Diplomarbeit gebühren Phoebe Sengers, meiner Schwester Martina und meinen Eltern für ihre Unterstützung. Dank auch an Ken Lang für einen Teil der Implementation des Textkategorisierungssystems und an Ian Phillipps für die Implementation des Stemmers.

Last but not least möchte ich bei Frau Prof. Morik und Herrn Prof. Fuhr für die Betreuung meiner Diplomarbeit bedanken.

A Beschreibung für das Hyperlink *projects* auf der SCS-Front-Door-Seite

Es folgt eine Auflistung der Interessen von Benutzern, die auf ihrer Tour dem Hyperlink *projects* auf der SCS-Front-Door-Seite (Abbildung 15) gefolgt sind. Für WebWatchers Lernmethode (siehe 2.5) werden die Interessenbeschreibungen zu einem Teil der internen Repräsentation dieses Hyperlinks.

70703 161 compuserve com,
72457 1460 compuserve com,
admission information intelligent agents,
ai,
ai,
andrew,
apple iigs stuff,
architecture operating systems distributed systems,
artificial intelligence and music,
australia,
browser,
browsers,
callback,
credit based,
electronic documents,
fast program,
geo,
graduate school,
human computer interfaces,
inex,
information about real time mach,
information on a life or ai,
information retrieval knowledge representation,
information retrieval,
informedia multimedia,
intelligent tutoring systems,
intelligent tutoring,
interesting computer graphics,
lamborghini,
machine learning at cmu,
machine learning,
machine learning,
mathematica,
natural language systems,
network management,
newsweeder,
project listen,
projects,
projects,
punch recipes,
recipes,
reinforcement learning,
reinforcement learning,
reinforcement learning,
reinforcement learning,
robot,
robotics,
robotics,
robotics,
robotics,
speech recognition,
study compiler computer theory,
technology info,
text learning,
tom mitchell,
warez,
xavier

B Implementation der Textkategorisierungsmethoden

Alle Textkategorisierungsmethoden sind in einem Programmpaket integriert. Die Implementation erfolgte in Perl5. Ablauf und Datenstrukturen basieren im wesentlichen auf UNIX-Pipes und temporären Files, wodurch auch große Datensätze mit mehreren 10000 Dokumenten problemlos bearbeitet werden können. Jeder Programmteil liest eine Menge von Dateien und erzeugt neue Dateien, die von den folgenden Stufen ausgewertet werden. Große Teile des Entwurfs, des Parsers und des Attributselektionsteils wurden aus Arbeiten von Ken Lang übernommen. Der Stemmer wurde von Ian Phillipps implementiert.

Im folgenden wird zuerst das Eingabeformat für Beispiele beschrieben (Abschnitt B.1). Dann wird in Abschnitt B.2 der Ablauf eines Kategorisierungsdurchlaufs vorgestellt und abschließend werden die einzelnen Programmodule mit ihren gegenseitigen Abhängigkeiten erläutert (Abschnitt B.3).

B.1 Eingabeformat

Das Eingabeformat für Dokumente[23] orientiert sich an Verzeichnisstrukturen. In einem Basisverzeichnis wird für jede Kategorie[24] ein Unterverzeichnis angelegt.

```
<Basisverzeichnis>/<Kategorie 1>
.
.
.
<Basisverzeichnis>/<Kategorie n>
```

In den Verzeichnissen sind die Dokumente entsprechend ihrer Klassenzugehörigkeit enthalten. Ein Dokument kann in mehreren Verzeichnissen enthalten sein (z. B. über einen symbolischen Link). Jedes Dokument wird in einer separaten Datei gespeichert. Dokumente haben einen Header nach MIME-Standard. Die MIME-Zeile `Message-ID` wird zur eindeutigen Identifikation von Dokumenten und zum Entfernen von Duplikaten verwendet. Die Zeile `Subjekt` enthält eine eventuell vorhandene Überschrift.

B.2 Ablauf

Abbildung 22 zeigt den groben Aufbau des Systems. Im ersten Schritt werden die Dokumente mit einem Parser (`parse.pl`) in eine effizientere Repräsentation überführt. Hierzu wird ein Isomorphismus zwischen den unterschiedlichen (auf Stammform reduzierten) Worten und einem Ausschnitt der natürlichen

[23] Bei den Experimenten in Abschnitt 3 entspricht jedes Benutzerinteresse einem Dokument.
[24] Entspricht Hyperlinks.

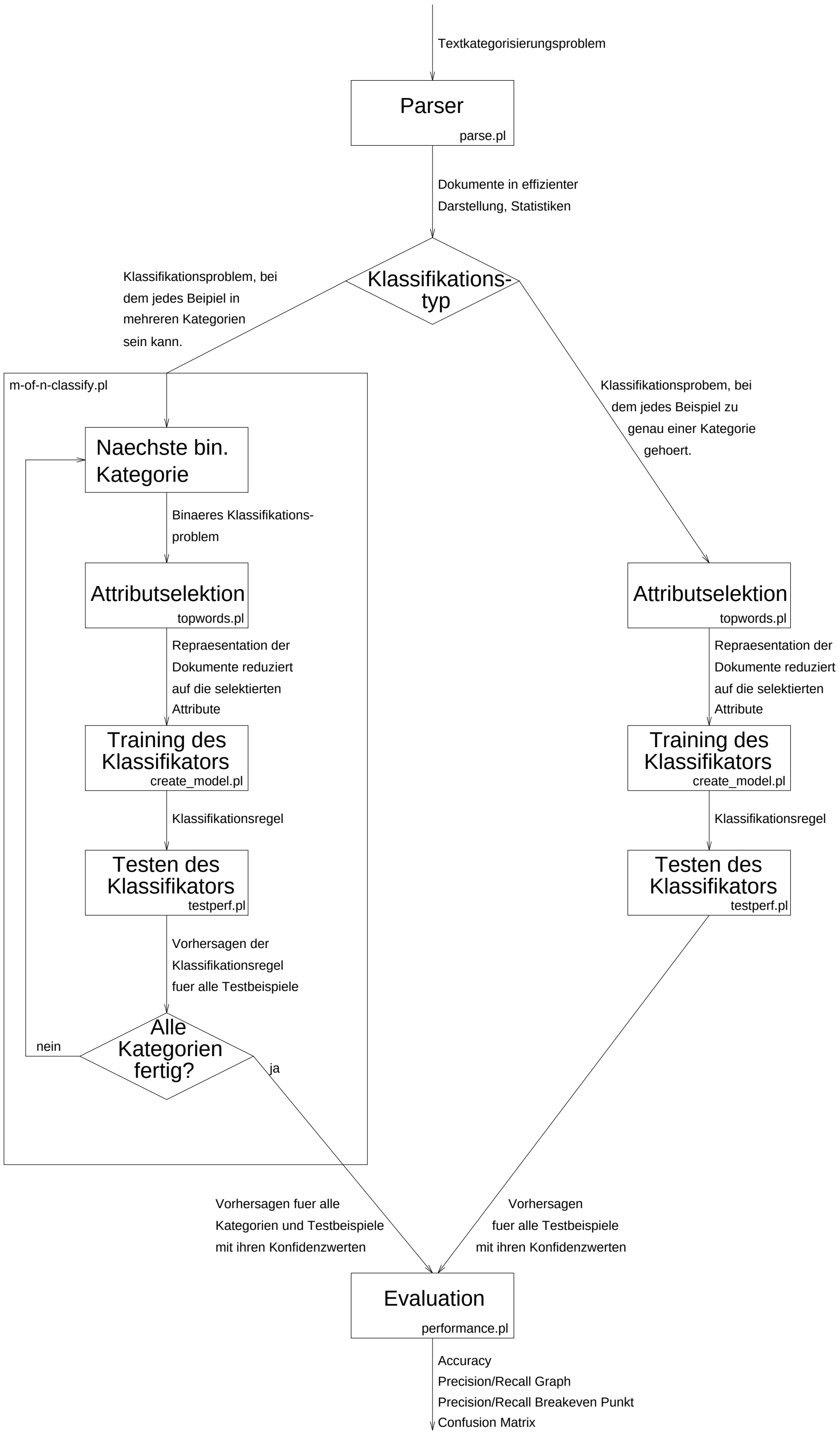

Abbildung 22: Ablaufschema des Textkategorisierungssystems.

Zahlen benutzt. Dadurch lassen sich Dokumente effizient in (spärlich besetzten) Matrizen repräsentieren. Vom Parser werden außerdem Statistiken, wie z. B. die Vorkommenshäufigkeiten von Termen, erstellt, die von den folgenden Stufen benutzt werden. Weiterhin teilt `parse.pl` die Menge der Dokumente zufällig in Test-/Trainingsbeispiele auf.

Zwei verschiedene Typen von Kategorisierungsproblemen können mit dem System bearbeitet werden. Ist jedes Dokument der betrachteten Kollektion in genau einer Klasse, so ist dem rechten Ast in Schaubild 22 zu folgen. Ansonsten wird das Problem in mehrere binäre Klassifikationsprobleme zerlegt (linker Ast) (`m-of-n-classify.pl`). Jedes binäre Problem wird getrennt betrachtet und anschließend werden die Ergebnisse zusammengeführt.

Nach dem Parsing werden mittels `topwords.pl` relevante Attribute selektiert. Das Ergebnis dieser Stufe ist die gleiche Form von Matrixdarstellung der Dokumente, wie sie von `parse.pl` geliefert wurde. Allerdings werden die Zahlen für Terme nicht mehr zufällig vergeben, sondern anhand der Bewertung des Attributselektionskriteriums. Der Term mit der höchsten Bewertung erhält die Nummer 1, der mit der zweithöchsten die Nummer 2, etc. Durch diese Organisation kann leicht mit verschiedenen Attributanzahlen experimentiert werden, ohne die Attributselektion erneut durchführen zu müssen.

Bis hierhin war der Prozeß unabhängig von der verwendeten Lernmethode. Mit dem Programm `create_model.pl` wird nun aus den Trainingsdaten eine Hypothese gelernt. Die implementierten Lernmethoden sind in Abschnitt 3.4 beschrieben. Die Ausgabe von `create_model.pl` ist eine Datei mit der gelernten Hypothese.

`testperf.pl` liest diese Hypothese ein und wendet sie auf die Testbeispiele an. Die Vorhersagen der Hypothese für jedes Testbeispiel werden in einer Datei gespeichert. `performance.pl` vergleicht die Vorhersagen mit den korrekten Klassifikationen und erstellt Precision/Recall-Graphen, Precision/Recall-Breakeven-Point, Accuracy/Coverage-Graphen und Accuracy(k)-Graphen. Die Ergebnisse werden in einer Log-Datei gespeichert.

Um Serien von Experimenten zu automatisieren, kann das Programm `run.pl` verwendet werden. Es kann benutzt werden, um verschiedene Parametersätze und Lernverfahren zu vergleichen. Die Ergebnisse werden automatisch über verschiedene Test-/Trainingsaufteilungen gemittelt.

B.3 Modulhierarchie

Abbildung 23 stellt die Abhängigkeiten der Module des Textkategorisierungssystems dar. Die einzelnen Module werden im folgenden kurz in ihrer Funktionalität beschrieben.

- `Experiment.Config`: Konfigurationsfile, in dem alle Parameter für einen Kategorisierungsdurchlauf angegeben werden (z. B. Verzeichnis mit Bei-

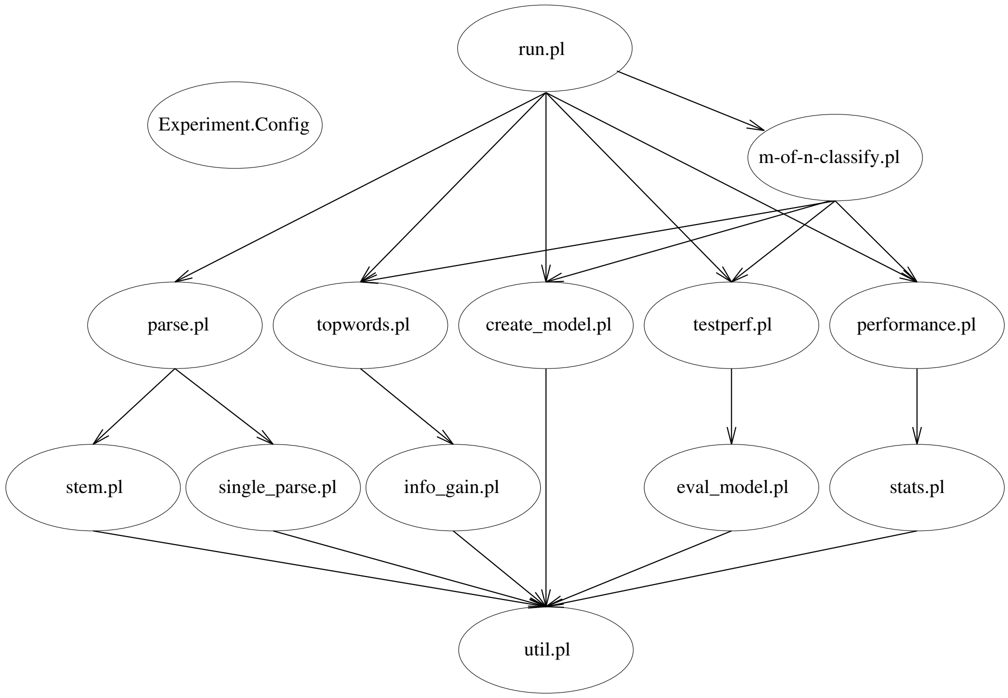

Abbildung 23: Abhängigkeiten der Programmodule des Textkategorisierungssystems.

spielen, Lernmethode, Stammformreduktion?, etc.). Jedes Modul hat Zugriff auf dieses Konfigurationsfile.

- `run.pl`: Steuert den Ablauf einer Serie von Experimenten. Beliebig viele Parametersätze können angegeben werden, für die jeweils über eine Anzahl von verschiedenen Test-/Trainingsmengen gemittelt wird. `run.pl` modifiziert `Experiment.Config` automatisch.

- `parse.pl`: Parsing von Dokumenten und Überführung in eine kompakte Repräsentation. Erstellung von Statistiken über Vorkommenshäufigkeiten von Klassen und Termen.

- `topwords.pl`: Attributselektion. Terme können anhand von drei Kriterien ausgewählt werden. (1) Terme mit einer Document-Frequency von mindestens n. (2) Terme, die nicht zu den m häufigsten gehören. (3) Mutual Information.

- `create_model.pl`: Lernen einer Klassifikationshypothese. Die oben vorgestellten Lernalgorithmen stehen zur Auswahl. Für jeden Lernalgorithmus enthält dieses Modul die zum Lernen verwendeten Prozeduren.

- `testperf.pl`: Anwendung der gelernten Klassifikationshypothese auf die Testbeispiele.

- `performance.pl`: Aufbereitung der Klassifikationsergebnisse zu Precisi on/Recall-Graphen, Precision/Recall-Breakeven-Point, Accuracy/Cover age-Graphen und Accuracy(k)-Graphen.
- `stats.pl`: Hilfsroutinen zur Berechnung der Performanzmaße.
- `eval_model.pl`: Für jeden Lernalgorithmus enthält dieses Modul die zur Klassifikation von Testbeispielen verwendeten Prozeduren.
- `info_gain.pl`: Funktionen zur Berechnung der Mutual Information für die Attributselektion.
- `single_parse.pl`: Prozeduren zum Parsing eines einzelnen Dokuments.
- `stem.pl`: Stemming Algorithmus nach Porter [Porter, 1980].
- `util.pl`: Hilfsprozeduren zum Zugriff auf Datenstrukturen.

C Implementation des Reinforcement-Learning-Ansatzes

Die Implementation des Reinforcement-Learning-Ansatzes erfolgte in Allegro Common Lisp 4.2. Einige untergeordnete Programmteile, wie z. B. ein Parser für HTML, wurden in C mit Hilfe von Flex realisiert.

Im folgenden wird wieder zuerst das Eingabeformat für Beispiele beschrieben (Abschnitt C.1). Dann wird in Abschnitt C.2 ein Überblick über den Programmablauf gegeben und abschließend werden die einzelnen Programmodule mit ihren gegenseitigen Abhängigkeiten erläutert (Abschnitt C.3).

C.1 Eingabeformat

WebWatchers Tourprotokolle werden zuerst mit einem Konvertierungsprogramm in eine Form gebracht, die eine effiziente Durchführung von Experimenten zuläßt. Auf das Konvertierungsprogramm wird nicht näher eingegangen, sondern es wird im folgenden die hier beschriebene Repräsentation als gegeben vorausgesetzt.

In einem Basisverzeichnis befinden sich die folgenden vier Dateien.

```
<Basisverzeichnis>/traces
<Basisverzeichnis>/visited_urls
<Basisverzeichnis>/urls
<Basisverzeichnis>/examples
```

`traces` enthält die Kennungen der Tourprotokolle, die in dem Datensatz enthalten sind. `visited_urls` enthält die URLs der Seiten, die auf diesen Touren besucht wurden. `urls` enthält zusätzlich die URLs in den Hyperlinks auf den besuchten Seiten. Weiterhin bildet `urls` alle URLs auf die Zahlen von 1 bis n ab. Dadurch brauchen anstelle von URLs im folgenden nur noch Zahlen betrachtet zu werden, was zu einer erheblichen Effizienzsteigerung beiträgt.

`examples` enthält eine Zeile für jedes Beispiel, das aus den Touren generiert wurde. Jede Zeile ist von der folgenden Form:

```
4776 81156 (81172 81173) "trace.17864" 3394
|    |     |             |            |
|    |     |             |            Nummer der Datei mit der
|    |     |             |            Seitenbeschreibung
|    |     |            Kennung des Tourprotokolls
|    |    Ausgewaehlte Hyperlinks
|   Von Seite
Beispiel-ID
```

Die obige Zeile beschreibt, daß auf der Tour mit der Kennung "trace.17864" der Benutzer von Seite 81156 den Hyperlinks zu den Seiten 81172 und 81173 gefolgt ist. Die Beschreibung der Seite 81156 befindet sich in der Datei mit dem Namen 3394, die in dem Verzeichnis

```
<Basisverzeichnis>/features/pfd
```

enthalten ist. Sie enthält den Titel der Seite sowie für jedes Hyperlink auf der Seite dessen Anchor-Text und dessen URL-Kennung. Außerdem befindet sich im Verzeichnis

```
<Basisverzeichnis>/features/taf
```

für jedes Beispiel eine Datei, benannt nach der Beispiel-ID, welche die vom Benutzer eingegebene Interessenbeschreibung enthält. Das Verzeichnis

```
<Basisverzeichnis>/features/pt
```

enthält für jeden URL in `visited_urls` die normalisierte TFIDF-Vektorraumdarstellung der entsprechenden Seite.

C.2 Ablauf

Der Ablauf eines Experiments ist in Abbildung 24 dargestellt. Ausgehend von dem oben beschriebenen Eingabeformat werden zuerst die in `traces` enthaltenen Tourprotokolle anhand eines vorgegebenen Verhältnisses in Test- und Trainingstouren aufgeteilt. Die Seiten, die während der Trainingstouren mindestens einmal besucht wurden, werden markiert. Abhängig von der zu verwendenden Methode wird nun gelernt.

Für den Bayes'schen Klassifikator werden für alle Seiten mit Trainingsbeispielen die entsprechenden Wahrscheinlichkeiten geschätzt. Soll später auf Seiten klassifiziert werden, für die es keine Trainingsbeispiele gibt, so werden während der Klassifikation die Wahrscheinlichkeiten anhand des Anchor-Textes geschätzt.

Für die Reinforcement-Learning-Methode werden zuerst die Belohnungsfunktionen $R_w(s)$ geladen. Sie ergeben sich direkt aus den Dateien im Verzeichnis `pt`. Danach wird für die Anchor-Texte und die Seitentitel ein invertierter Index angelegt, der die schnelle Bestimmung der k nächsten Nachbarn zur Funktionsapproximation erlaubt[25]. Nun werden die Q-Funktionen mit Hilfe von synchroner dynamischer Programmierung bestimmt. Pro Hyperlink (s, a) existiert ein Vektor, der die Werte von $Q_w(s, a)$ für alle w zusammenfaßt. Diese Vektoren sind nur spärlich besetzt. Deshalb wird zu ihrer effizienten Speicherung ein spezieller Datentyp verwendet, der diese Eigenschaft berücksichtigt.

Nach dem Lernen wird die jeweilige Methode auf die Testbeispiele angewandt und die erzeugten Rankings werden in einer Datei gespeichert. Durch Analyse dieser Datei können anschließend verschiedene Performanzmaße berechnet werden.

[25] Im Grunde ist die Bestimmung der nächsten Nachbarn genau das, was ein Text-Retrieval-System im Sinne von SMART für eine Anfrage macht.

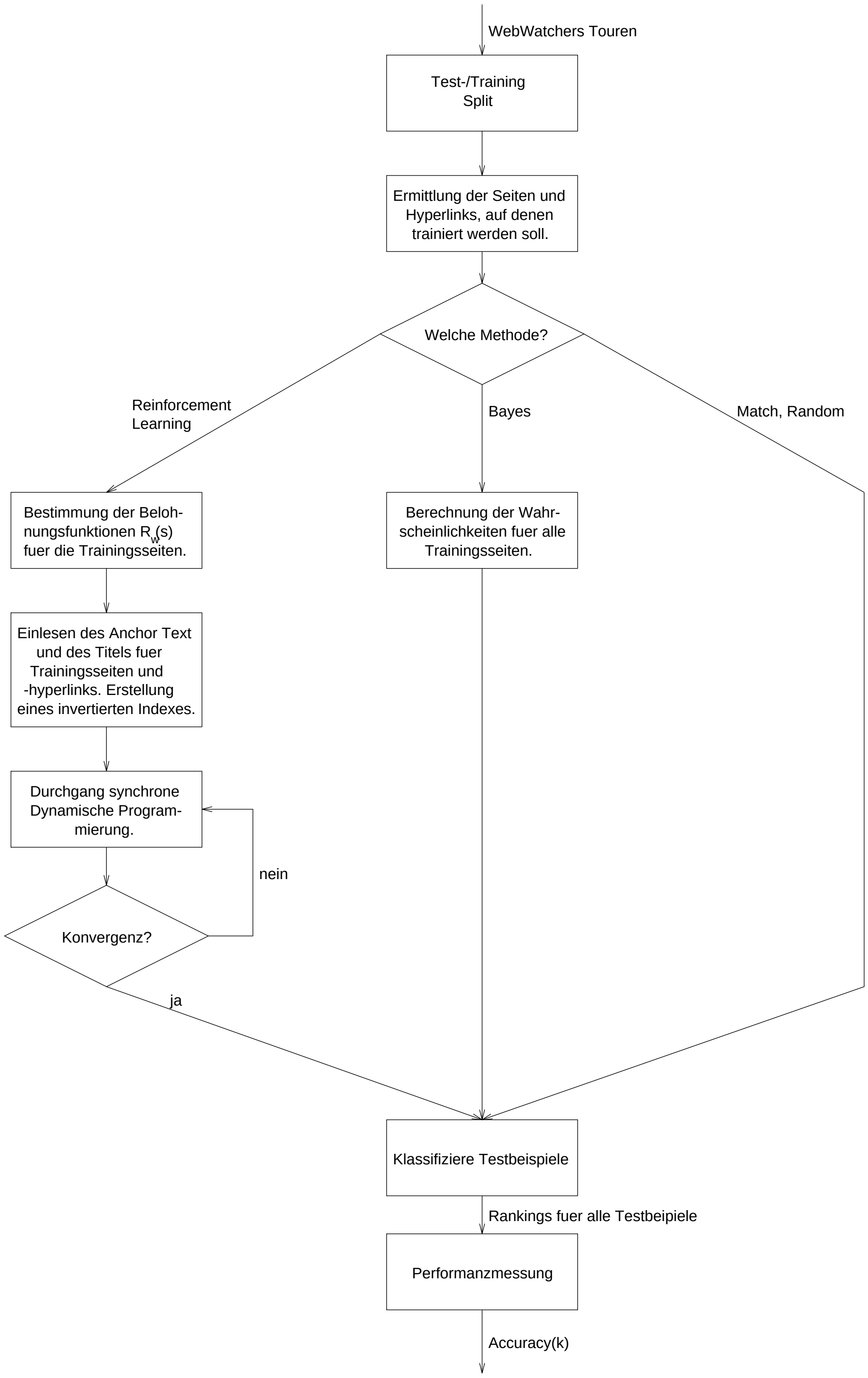

Abbildung 24: Ablaufschema für ein Experiment.

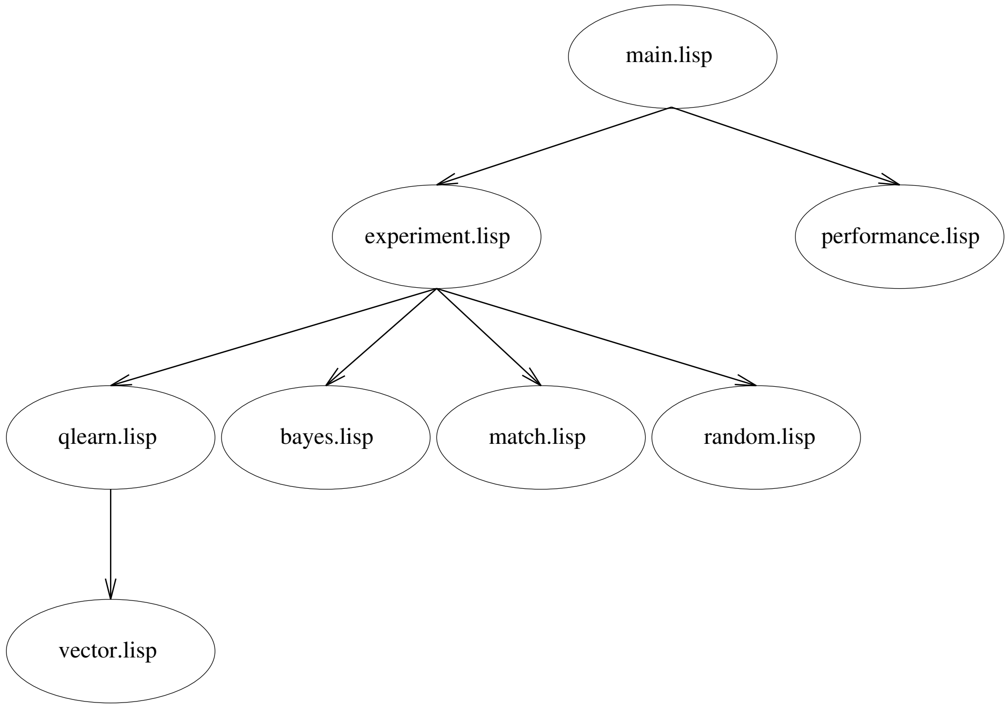

Abbildung 25: Abhängigkeiten der Programmodule des Reinforcement-Learning-Systems.

C.3 Modulhierarchie

Im folgenden werden kurz die einzelnen Module des Programmpakets beschrieben, das für die Reinforcement Learning Experimente verwendet wurde. Abbildung 25 zeigt die Abhängigkeitsbeziehungen zwischen den Modulen.

- `main.lisp`: Ähnlich wie `run.pl` können mit diesem Programmteil Serien von Experimenten spezifiziert und automatisch durchgeführt werden. Es kann eine beliebig lange Folge von verschiedenen Parametersätzen angegeben werden. Für jeden Parametersatz wird die Performanz über eine Reihe von verschiedenen Test-/Trainingsbeispielmengen gemittelt. Hierbei werden für alle Parametersätze die gleichen Test-/Trainingsbeispielmengen verwendet, so daß Signifikanztests, die paarweise Daten voraussetzen, eingesetzt werden können.

- `experiment.lisp`: Dieses Modul enthält die Funktionen, die zum Ablauf eines einzelnen Experiments benötigt werden. Im wesentlichen wird eine Funktion von diesem Modul exportiert, die für einen Parametersatz und eine vorgegebene Aufteilung in Test- und Trainingsbeispiele die $Accuracy(3)$ liefert. Außerdem produziert diese Funktion eine Datei, in der alle Vorhersagen des Lernalgorithmus auf den Testdaten in einem Standardformat gespeichert sind. Anhand dieser Datei können auch andere Performanzmaße nachträglich berechnet werden.

- `qlearn.lisp`: Hier sind alle Funktionen, die mit der Reinforcement-Learning-Methode in Verbindung stehen, zusammengefaßt. Das Modul exportiert eine Funktion *learn* und eine Funktion *rank* nach `experiment.lisp`. Für eine Menge von Trainingsbeispielen lernt *learn* die Q-Funktionen. Mit *rank* kann dann ein Testbeispiel klassifiziert werden.

- `bayes.lisp`: Enthält die Funktionen für den naiven Bayes'schen Klassifikator. Auch hier werden wieder zwei Funktionen *learn* und *rank* exportiert.

- `match.lisp`: Implementiert die nicht lernende Methode *Match*. Deshalb wird auch nur eine Funktion *rank* exportiert.

- `random.lisp`: Exportiert eine Funktion *rank*, die ein zufälliges Ranking der Hyperlinks eines Testbeispiels liefert.

- `performance.lisp`: Dieses Modul enthält die Funktionen zur Performanzmessung. Basierend auf der Datei mit den Klassifizierungen der Testbeispiele, die von `experiment.lisp` erzeugt wird, können verschiedene Performanzmaße berechnet werden. Im Moment sind Accuracy(k)-Graphen und Accuracy/Coverage-Graphen implementiert.

- `vektor.lisp`: Datentyp zur effizienten Verwaltung von spärlich besetzten Vektoren. Der Datentyp wird im Modul `qlearn.lisp` verwendet.

Literatur

[Agosti et al., 1992] Agosti, M., Gradenigo, G. und Marchetti, P. (1992). A Hypertext Environment for Interacting with Large Textual Databases. *Information Processing and Management*, 28(3):371–387.

[Altavista, 1996] Altavista (1996). http://altavista.digital.com/.

[Apté und Damerau, 1994] Apté, C. und Damerau, F. (1994). Automated Learning of Decision Rules for Text Categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

[Armstrong et al., 1995] Armstrong, R., Freitag, D., Joachims, T. und Mitchell, T. (1995). WebWatcher: A Learning Apprentice for the World Wide Web. In *AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments.*

[Ashmore] Ashmore, L. University of Virginia. Persönliche Kommunikation.

[Balabanovic und Shoham, 1995] Balabanovic, M. und Shoham, Y. (1995). Learning Information Retrieval Agents: Experiments with Automated Web Browsing. In *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments.* AAAI-Press.

[Bartell et al., 1992] Bartell, B., Cottrell, G. und Belew, R. (1992). Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling. In *Proceedings of the ACM SIGIR.*

[Barto et al., 1995] Barto, A. G., Bradtke, S. J. und Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138.

[Bates, 1986] Bates, M. (1986). An Exploratory Paradigm for Online Information Retrieval. In Brookes, B., Herausgeber, *Intelligent Information Systems for the Information Society*, Seiten 91–99. North-Holland, Amsterdam.

[Belkin et al., 1993] Belkin, N., Marchetti, P. und Cool, C. (1993). Braque: Design of an Interface to Support User Interaction in Information Retrieval. *Information Processing and Management*, 29(3):325–344.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming.* Princeton University Press.

[Berry et al., 1995] Berry, M., Dumais, S. und Shippy, A. (1995). A Case Study of Latent Semantic Indexing. Technical Report CS-95-271, Computer Science Department, University of Tennessee at Knoxville.

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

[Blair, 1992] Blair, D. (1992). Information Retrieval and the Philosophy of Language. *The Computer Journal*, 35(3).

[Blum, 1995] Blum, A. (1995). Empirical Support for Winnow and Weighted-Majority Based Algorithms: Results on a Calendar Scheduling Domain. In *International Conference on Machine Learning (ICML)*.

[Bollmann et al., 1992] Bollmann, P., Raghavan, V. und Jung, G. (1992). On Probabilistic Notions of Precision as a Function of Recall. *Information Processing and Management*, 28(3):291–315.

[Boyan et al., 1996] Boyan, J., Freitag, D. und Joachims, T. (1996). A Machine Learning Architecture for Optimizing Web Search Engines. In *AAAI Workshop on Internet Based Information Systems*.

[Brachman und Schmolze, 1985] Brachman, R. und Schmolze, J. (1985). An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216.

[Campagnoni und Ehrlich, 1989] Campagnoni, F. und Ehrlich, K. (1989). Information Retrieval Using a Hypertext-Based Help System. *ACM Transactions on Information Systems*, 7(3):271–291.

[Caroll, 1972] Caroll, L. (1972). *The Complete Illustrated Works of Lewis Carroll.* Chancellor Press, London, UK.

[Caruana und Freitag, 1994] Caruana, R. und Freitag, D. (1994). Greedy Attribute Selection. In *International Conference on Machine Learning (ICML)*.

[Cassandra et al., 1994] Cassandra, A., Kaebling, L. und Littman, M. (1994). Acting Optimally in Partially Observable Stochastic Domains. In *National Conference on Artificial Intelligence*, Seattle.

[Cheeseman et al., 1988] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W. und Freeman, D. (1988). AutoClass: A Bayesian Classification System. In *Proceedings of the Fifth International Conference on Machine Learning (ICML)*, Seiten 54–56.

[Cohen, 1995a] Cohen, W. (1995). Learning to Classify English Text with ILP Methods. In *Advances in Inductive Logic Programming.* IOS Press.

[Cohen, 1995b] Cohen, W. (1995). Text Categorization and Relational Learning. In *International Conference on Machine Learning (ICML)*, Seiten 124–132, Lake Tahoe.

[Cohen, 1996] Cohen, W. (1996). Context-Sensitive Learning Methods for Text Categorization. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[Cooper, 1991] Cooper, W. (1991). Some Inconsistencies and Misnomers in Probabilistic Information Retrieval. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seiten 57–61.

[Cover und Thomas, 1991] Cover, T. und Thomas, J. (1991). *Elements of Information Theory*. Wiley.

[Croft et al., 1989] Croft, B., Lucia, T., Cringean, J. und Willett, P. (1989). Retrieving Documents by Plausible Inference: An Experimental Study. *Information Processing and Management*, 25(6):599–614.

[Croft und Turtle, 1993] Croft, B. und Turtle, H. (1993). Retrieval Strategies for Hypertext. *Information Processing and Management*, 29(3):313–324.

[Croft und Lewis, 1990] Croft, W. und Lewis, D. (1990). Term Clustering of Syntactic Phrases. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seiten 385–404.

[Crouch, 1988] Crouch, C. (1988). A Cluster-Based Approach to Thesaurus Construction. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seiten 309–320.

[de Kroon et al., 1996] de Kroon, E., Mitchell, T. und Kerckhoffs, E. (1996). Improving Learning Accuracy in Information Filtering. In *International Conference on Machine Learning (ICML), Workshop on Machine Learning meets Human Computer Interaction*, Seiten 41–58.

[Deerwester et al., 1990] Deerwester, S., Dumais, S., Furnas, G., Landauer, T. und Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

[Dieberger, 1996] Dieberger, A. (1996). Browsing the WWW by Interacting with a Textual Virtual Environment - A Framework for Experimenting with Navigational Metaphors. In *ACM Proceedings on Hypertext*, Seiten 170–179.

[Dumais, 1994] Dumais, S. (1994). Latent Semantic Indexing (LSI) and TREC-2. Technical Report TM-ARH-023878, Bellcore.

[Dunlop und van Rijsbergen, 1993] Dunlop, M. und van Rijsbergen, C. (1993). Hypermedia and Free Text Retrieval. *Information Processing and Management*, 29(3):287–298.

[Ellis, 1990] Ellis, D. (1990). A Behavioural Approach to Information Retrieval System Design. *Journal of Documentation*, 45(3):171–212.

[Fagan, 1987] Fagan, J. (1987). *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and non-Syntactic Methods*. PhD thesis, Department of Computer Science, Cornell University.

[Foltz, 1990] Foltz, P. (1990). Using Latent Semantic Indexing for Information Filtering. In *Conference on Office Information Systems*, Seiten 40–47.

[Foltz und Dumais, 1992] Foltz, P. und Dumais, S. (1992). Personalized Information Delivery: An Analysis of Information Filtering Methods. *Communications of the ACM*, 35:51–60.

[Frei und Schäuble, 1991] Frei, H. und Schäuble, P. (1991). Determining the Effectiveness of Retrieval Algorithms. *Information Processing and Management*, 27(2/3):153–164.

[Freitag et al., 1996] Freitag, D., Hirsh, H., Joachims, T., Kawamura, A., de Kroon, E., Loewenstern, D., McBarron, B., Mitchell, T. und Slattery, S. (1996). Experiments in Learning from Text. To be published.

[Frisse und Cousins, 1989] Frisse, M. und Cousins, S. (1989). Information Retrieval from Hypertext: Update on the Dynamic Medical Handbook Project. In *ACM Conference on Hypertext*, Seiten 199–212.

[Fuhr, 1989] Fuhr, N. (1989). Models for Retrieval with Probabilistic Indexing. *Journal of Information Processing and Management*, 25(1):55–72.

[Fuhr und Buckley, 1991] Fuhr, N. und Buckley, C. (1991). A Probabilistic Learning Approach for Document Indexing. *ACM Transactions on Information Systems*, 9(3):223–248.

[Fuhr et al., 1991] Fuhr, N., Hartmann, S., Lustig, G., Schwantner, M., Tzeras, K. und Knorz, G. (1991). AIR/X - a Rule-Based Multistage Indexing System for Large Subject Fields. In *RIAO*, Seiten 606–623.

[Fuhr und Knorz, 1984] Fuhr, N. und Knorz, G. (1984). Retrieval Test Evaluation of a Rule Based Automatic Indexing (AIR/PHYS). In van Rijsbergen, C., Herausgeber, *Research and Development in Information Retrieval: Proceedings of the Third Joint BCS and ACM Symposium*, Seiten 391–408. Cambridge University Press.

[Furnas, 1986] Furnas, G. (1986). Generalized Fisheye Views. In *Proceedings of Human Factors in Computing Systems (CHI)*, Seiten 16–23.

[Gordon, 1995] Gordon, G. (1995). Stable Function Approximation in Dynamic Programming. In *International Conference on Machine Learning (ICML)*.

[Hart et al., 1968] Hart, P., Nilsson, N. und Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems, Science, and Cybernetics*, 4:100–107.

[Hayes und Weinstein, 1990] Hayes, P. und Weinstein, S. (1990). CONSTRUE/TIS: a System for Content-Based Indexing of a Database of News Stories. In *Annual Conference on Innovative Applications of AI*.

[Hjerppe, 1986] Hjerppe, R. (1986). HYPERCATalog: Visions and Preliminary Conceptions of an Extended and Enhanced Catalog. In Brookes, B., Herausgeber, *Intelligent Information Systems for the Information Society*, Seiten 221–232. Elsevier, Amsterdam.

[Jacobson, 1991] Jacobson, T. (1991). Sense-Making in a Database Environment. *Information Processing and Management*, 27(6):647–657.

[James, 1985] James, M. (1985). *Classification Algorithms*. Wiley.

[Joachims, 1996] Joachims, T. (1996). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. Technical Report CMU-CS-96-118, School of Computer Science, Carnegie Mellon University.

[Joachims et al., 1995] Joachims, T., Mitchell, T., Freitag, D. und Armstrong, R. (1995). WebWatcher: Machine Learning and Hypertext. In Morik, K. und Herrmann, J., Herausgeber, *GI Fachgruppentreffen Maschinelles Lernen*. University of Dortmund.

[Jones und Cockburn, 1996] Jones, S. und Cockburn, A. (1996). A Study of Navigational Support Provided by Two World Wide Web Browsing Applications. In *ACM Proceedings on Hypertext*.

[Korf, 1990] Korf, R. (1990). Real-Time Heuristic Search. *Artificial Intelligence*, 42:189–211.

[Kreyszig, 1975] Kreyszig, E. (1975). *Statistische Methoden und ihre Anwendungen*. Vandenhoeck & Ruprecht, Göttingen, 5. Auflage.

[Lang, 1995] Lang, K. (1995). NewsWeeder: Learning to Filter Netnews. In *International Conference on Machine Learning (ICML)*.

[Larson, 1982] Larson, H. (1982). *Introduction to Probability Theory and Statistical Inference*. Wiley, New York.

[Lebowitz, 1987] Lebowitz, M. (1987). Experiments with Incremental Concept Formation: UNIMEM. *Machine Learning Journal*, 2:103–138.

[Lewis, 1992a] Lewis, D. (1992). An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[Lewis, 1992b] Lewis, D. (1992). *Representation and Learning in Information Retrieval*. PhD thesis, Department of Computer and Information Science, University of Massachusetts.

[Lewis und Ringuette, 1994] Lewis, D. und Ringuette, M. (1994). A Comparison of Two Learning Algorithms for Text Classification. In *Third Annual Symposium on Document Analysis and Information Retrieval*, Seiten 81–93.

[Lieberman, 1995] Lieberman, H. (1995). Letizia: An Agent That Assists Web Browsing. In *International Joint Conference on Artificial Intelligence, Montreal*.

[Lin, 1991] Lin, L.-J. (1991). Programming Robots Using Reinforcement Learning. In *Ninth National Conference on Artificial Intelligence*, Seiten 781–786.

[Littlestone, 1987] Littlestone, N. (1987). Learning Quickly when Irrelevant Attributes Abound: A New Linear Threshold Algorithm. *Machine Learning Journal*, 2:285–318.

[Lucarella und Zanzi, 1993] Lucarella, D. und Zanzi, A. (1993). Information Retrieval from Hypertext: An Approach Using Plausible Inference. *Information Processing and Management*, 29(3):299–312.

[Lycos, 1996] Lycos (1996). http://www.lycos.com/.

[Mahadevan und Connell, 1992] Mahadevan, S. und Connell, J. (1992). Automatic Programming of Behaviour-Based Robots Using Reinforcement Learning. *Artificial Intelligence*, 55:311–365.

[Maron und Kuhns, 1960] Maron, M. und Kuhns, J. (1960). On Relevance, Probabilistic Indexing, and Information Retrieval. *Journal of the Association for Computing Machinery*, 7(3):216–244.

[Maron, 1961] Maron, M. E. (1961). Automatic Indexing: An Experimental Inquiry. *Journal of the Association for Computing Machinery*, 8:404–417.

[Masand et al., 1992] Masand, B., Linoff, G. und Waltz, D. (1992). Classifying News Stories using Memory Based Reasoning. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seiten 59–65.

[Meadow, 1992] Meadow, C. (1992). *Text Information Retrieval Systems*. New York Academic Press.

[Michie et al., 1994] Michie, D., Spiegelhalter, D. und Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

[Mitchell, 1996] Mitchell, T. (1996). Carnegie Mellon University. Persönliche Kommunikation.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

[Mitchell et al., 1994] Mitchell, T., Caruana, R., Freitag, D., McDermott, J. und Zabowski, D. (1994). Experience with a Learning Personal Assistent. *Communications of the ACM*, 37(7):81–91.

[Mitchell et al., 1985] Mitchell, T., Mahadevan, S. und Steinberg, L. (1985). LEAP: A Learning Apprentice for VLSI Design. In *Ninth International Joint Conference on Artificial Intelligence*.

[Moulinier und Ganascia, 1996] Moulinier, I. und Ganascia, J. (1996). Applying an Existing Machine Learning Algorithm to Text Categorization. In Wermter, S., Riloff, E. und Scheler, G., Herausgeber, *Connectionist, statistical, and symbolic approaches to learning for natural language processing*. Springer-Verlag.

[Moulinier et al., 1996] Moulinier, I., Raskinis, G. und Ganascia, J. (1996). Text Categorization: A Symbolic Approach. In *Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*.

[Nielsen, 1990] Nielsen, J. (1990). The Art of Navigating through Hypertext. *Communications of the Association of Computing Machinery*, 33(3):296–310.

[Pazzani et al., 1996] Pazzani, M., Muramatsu, J. und Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. In *AAAI Conference, Portland.*

[Pinkerton, 1994] Pinkerton, B. (1994). Finding What People Want: Experiences with the WebCrawler. In *Second International WWW Conference.*

[Pollard, 1993] Pollard, R. (1993). A Hypertext-Based Thesaurus as a Subject Browsing Aid for Bibliographic Databases. *Information Processing and Management*, 29(3):345–357.

[Porter, 1980] Porter, M. (1980). An Algorithm for Suffix Stripping. *Program (Automated Library and Information Systems)*, 14(3):130–137.

[Quinlan, 1986] Quinlan, R. (1986). Induction of Decision Trees. *Machine Learning Journal*, 1:81–106.

[Quinlan und Cameron-Jones, 1993] Quinlan, R. und Cameron-Jones, R. (1993). FOIL: A Midterm Report. In *European Conference on Machine Learning (ECML).*

[Raghavan et al., 1989] Raghavan, V., Bollmann, P. und Jung, G. (1989). A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance. *ACM Transactions on Information Systems*, 7(3):205–229.

[Robertson, 1977] Robertson, S. (1977). The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304.

[Rocchio, 1971] Rocchio, J. (1971). Relevance Feedback in Information Retrieval. In Salton, G., Herausgeber, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Seiten 313–323. Prentice-Hall Inc.

[Rosenblatt, 1962] Rosenblatt, F. (1962). *Principles of Neurodynamics.* Spartan Books, New York.

[Salton, 1971] Salton, G. (1971). *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall Inc.

[Salton, 1991] Salton, G. (1991). Developments in Automatic Text Retrieval. *Science*, 253:974–979.

[Salton und Buckley, 1987] Salton, G. und Buckley, C. (1987). Term Weighting Approaches in Automatic Text Retrieval. Technical Report 87-881, Department of Computer Science, Cornell University.

[Samuel, 1959] Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal on Research and Development*, Seiten 210–229.

[Savoy, 1991] Savoy, J. (1991). Spreading Activation in Hypertext Systems. Technical Report 761, Université de Montréal.

[Schütze et al., 1995] Schütze, H., Hull, D. und Pedersen, J. (1995). A Comparison of Classifiers and Document Representations for the Routing Problem. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[Sengers, 1996] Sengers, P. (1996). Socially Situated AI: What It Is and Why It Matters. In *AAAI-96 Workshop on AI & Entertainment*, Menlo Park, CA. AAAI Press.

[Shannon und Weaver, 1949] Shannon, C. und Weaver, W. (1949). *The Mathematical Theory of Communication.* University of Illinois Press, Urbana.

[Spark-Jones, 1973] Spark-Jones, K. (1973). Collection Properties Influencing Automatic Term Classification Performance. *Information Storage and Retrieval*, 9:499–513.

[Sutton, 1984] Sutton, R. (1984). *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, Amherst.

[Sutton, 1988] Sutton, R. (1988). Learning to Predict by the Method of Temporal Differences. *Machine Learning Journal*, 3:9–44.

[Tague, 1981] Tague, J. (1981). The Pragmatics of Information Retrieval Experimentation. In Spark-Jones, K., Herausgeber, *Information Retrieval Experiment.* Butterworths, London.

[Tague-Sutcliffe, 1992] Tague-Sutcliffe, J. (1992). Measuring the Informativeness of a Retrieval Process. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seiten 23–36.

[Tesauro, 1992] Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. *Machine Learning Journal*, 8:257–277.

[Thrun, 1994] Thrun, S. (1994). Learning to Play the Game of Chess. In *Advances in Neural Information Processing Systems (NIPS).*

[Thrun, 1995] Thrun, S. (1995). An Approach to Learning Mobile Robot Navigation. *Robotics and Autonomous Systems*, 15:301–319.

[Tzeras und Hartmann, 1993] Tzeras, K. und Hartmann, S. (1993). Automatic Indexing Based on Bayesian Inference Networks. In *Proceedings of the ACM SIGIR*, Seiten 22–34.

[van Rijsbergen, 1977] van Rijsbergen, C. (1977). A Theoretical Basis for the Use of Co-Occurrence Data in Information Retrieval. *Journal of Documentation*, 33(2):106–119.

[van Rijsbergen, 1979] van Rijsbergen, C. (1979). *Information Retrieval.* Butterworths, London, 2. Auflage.

[Vapnik, 1982] Vapnik, V. (1982). *Estimation of Dependencies Based on Empirical Data.* Springer Series in Statistics. Springer-Verlag.

[Wang et al., 1992] Wang, Z., Wong, S. und Yao, Y. (1992). An Analysis of Vector Space Models Based on Computational Geometry. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seiten 152–160.

[Watkins, 1989] Watkins, C. (1989). *Learning from Delayed Rewards.* PhD thesis, Cambridge University.

[Watkins und Dayan, 1992] Watkins, C. und Dayan, P. (1992). Technical Note: Q-Learning. *Machine Learning Journal*, 8(3):279–292.

[Webcrawler, 1996] Webcrawler (1996). http://www.webcrawler.com/.

[Weiss und Indurkhya, 1993] Weiss, S. und Indurkhya, N. (1993). Optimized Rule Induction. *IEEE Expert*, 8(6):61–69.

[Wiener et al., 1995] Wiener, E., Pedersen, J. und Weigend, A. (1995). A Neural Network Approach to Topic Spotting. In *Annual Symposium on Document Analysis and Information Retrieval (SDAIR).*

[Wilbur und Sirotkin, 1992] Wilbur, J. und Sirotkin, K. (1992). The Automatic Identification of Stop Words. *Journal of Information Science*, 18:45–55.

[Witten et al., 1996] Witten, I., Nevill-Manning, C. und Maulsby, D. (1996). Interacting with Learning Agents: Implications for ML from HCI. In *International Conference on Machine Learning (ICML), Workshop on Machine Learning meets Human Computer Interaction*, Seiten 51–58.

[Wolfram et al., 1996] Wolfram, D., Volz, A. und Dimitroff, A. (1996). The Effect of Linkage Structure on Retrieval Performance in a Hypertext-Based Bibliographic Retrieval System. *Information Processing and Management*, 32(5):529–541.

[Yahoo!, 1996] Yahoo! (1996). http://www.yahoo.com/.

[Yang, 1995] Yang, Y. (1995). Noise Reduction in a Statistical Approach to Text Categorization. In *Proceedings of the ACM SIGIR.*

[Yang, 1996] Yang, Y. (1996). An Application of Statistical Learning to News Story Categorization. To be published.

[Yang und Chute, 1993] Yang, Y. und Chute, C. (1993). Words or Concepts: the Features of Indexing Units and their Optimal Use in Information Retrieval. In *Annual Symposium on Computer Applications in Medical Care (SCAMC)*, Seiten 685–689.

[Yang und Pedersen, 1996] Yang, Y. und Pedersen, J. (1996). Feature Selection in Statistical Learning of Text Categorization. To be published.

[Yang und Wilbur, 1996] Yang, Y. und Wilbur, J. (1996). Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, 47(5):357–369.

Abkürzungsverzeichnis

$\Pr(A)$	Wahrscheinlichkeit von A
$\Pr(A\|B)$	Wahrscheinlichkeit von A gegeben B
$p(A)$	Dichtefunktion für A
$p(A\|B)$	Dichtefunktion für A gegeben B
$\hat{\Pr}(A)$	Schätzwert einer Wahrscheinlichkeit
$\hat{p}(A)$	Schätzwert einer Dichtefunktion
$E(A)$	Entropie von A
$\Re$	Menge der reellen Zahlen
$\|D\|$	Kardinalität der Menge D
$\|\|\vec{a}\|\|$	Länge des Vektors $\vec{a}$
$d(\vec{a}, \vec{b})$	Euklidscher Abstand von $\vec{a}$ nach $\vec{b}$
$\angle(\vec{a}, \vec{b})$	Winkel zwischen den Vektoren $\vec{a}$ und $\vec{b}$
$\cos(\angle(\vec{a}, \vec{b}))$	Kosinus des Winkels zwischen den Vektoren $\vec{a}$ und $\vec{b}$
$\max_{x \in X} f(x)$	Liefert den maximalen Wert $f(x)$ für $x \in X$ maximal wird
$\arg\max_{x \in X} f(x)$	Liefert den Wert $x \in X$, für den $f(x)$ maximal wird
$w, w_1, w_2, ...$	Worte
d, d', d''	Dokumente
$\vec{d}, \vec{d'}, \vec{d''}$	Dokumente in Vektorrepräsentation
D	Menge von Trainingsdokumenten
D_+	Menge von positiven Trainingsdokumenten
D_-	Menge von negativen Trainingsdokumenten
$C, C_1, C_2, ...$	Kategorien
$\mathcal{C}$	Menge der Kategorien, $\{C_1, C_2, ...\} = \mathcal{C}$
$T(d)$	Zielkonzept
$H(d)$	Hypothese/Entscheidungsregel
F	Menge der Attribute
$OCC(w, d)$	Liefert 1, wenn Wort w in Dokument d. Ansonsten 0.
$TF(w, d)$	Term Frequency von Wort w in Dokument d
$DF(w)$	Document-Frequency von Wort w
$IDF(w)$	Inverse-Document-Frequency von Wort w
$R(s)$	Belohnung im Zustand s
$Q(s, a)$	Q-Funktion für Aktion a in Zustand s
o. B. d. A.	ohne Beschränkung der Allgemeinheit