Diplomarbeit

Suche und Extraktion relevanter Informationen aus dem Internet zur Ähnlichkeitsbewertung verschiedener Aspekte einer virtuellen Konferenz

Christian Schwering

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

11. August 1999

Betreuer:

Prof. Dr. Katharina Morik
Dipl. Inform. Stefan Haustein

Nichts auf der Welt
ist so mächtig
wie eine Idee, deren
Zeit gekommen ist.

VICTOR HUGO

Zusammenfassung

Thema dieser Diplomarbeit ist der Entwurf und die Implementierung eines Informationsagentensytems für die Beschaffung von personenspezifischen Daten aus dem World Wide Web. Die in ausgesuchten Datenquellen gefundenen und extrahierten Daten werden anschließend zur Wissensentdeckung genutzt. Es sollen Regeln entdeckt werden, die über das Verhältnis zwischen Personen Aufschluss geben. Im Vordergrund steht eine praktische Untersuchung über die Möglichkeiten und Schwierigkeiten von Agentensystemen im World Wide Web.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich bei dieser Diplomarbeit unterstützt haben.

Bei Dr. Stephan Busemann, von der DFKI und Prof. Dr. Bernhard Nebel, von der Universität Freiburg, für die Bereitschaft sich als Testperson zur Verfügung zu stellen.

Bei allen Mitarbeitern des Lehrstuhl VIII, insbesondere bei Peter Brockhausen, der mir zahlreiche Hinweise zur Benutzung von MOBAL gab.

Mein ganz besonderer Dank geht an Prof. Dr. Katharina Morik, die mir mit dem Thema dieser Diplomarbeit eine spannende und zukunftsorientierte Aufgabe gestellt hat und natürlich an Stefan Haustein, der immer ein offenes Ohr für meine kleinen und großen Probleme hatte.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Wir ertrinken in Informationen, aber hungern nach Wissen.

JOHN NAISBITT, MEGATRENDS

Ein leichter Zugang zu aktueller und hochwertiger Fachinformation ist für Forschung und Lehre unabdingbar. Das Internet bietet eine weite Palette von Informationsdiensten für eine Suche nach relevanten Informationen. Die bekanntesten sind Gopher, Archie, WAIS und das World Wide Web (WWW) [Scheller et al., 1994], wobei sich das WWW inzwischen als wohl wichtigster Dienst durchgesetzt hat. Die Informationssuche im Internet gestaltet sich jedoch immer schwieriger – einerseits auf Grund der Komplexität und Größe, andererseits wegen des relativ schnellen Wandels und Wachstums. John Naisbitt [Naisbitt, 1985] schrieb 1985 vorausschauend:

> Maß und Umfang an Informationen sind mit den gegenwärtigen Mitteln nicht mehr zu bewältigen. [...] Wissenschaftler, die mit technischen Daten überhäuft werden [...] erklären, mitunter sei es einfacher ein Experiment selber durchzuführen, [...] als in der Flut wissenschaftlicher Veröffentlichungen nachzuwühlen [...], ob dieses Experiment schon einmal geführt worden ist oder nicht.

1982 ging man davon aus, dass sich das Informationsvolumen alle 5 Jahre verdoppeln würde. 1988 verdoppelte es sich alle 2,2 Jahre und 1992 alle 1,6 Jahre [Davies und Weeks, 1995]. Die MIT Media Labs sprachen 1996 davon, dass sich das WWW alle 50 Tage verdoppelt und Bob Johnson, Analyst bei Dataquest Inc. sagte (zitiert in [Nwana, 1996]):

> »*in the future, it [agents] is going to be the only way to search the Internet, because no matter how much better the Internet may be organised, it can't keep pace with the growth in information ...*«

Um das Informationssystem WWW effizient nutzen zu können, müssen neue Wege gegangen werden, hier finden besondere Typen von Softwareagenten ihre Anwendung: die *Informationsagenten*.

Diese Informationsagenten sind auch Thema vorliegender Diplomarbeit. Wo liegen die Einsatzgebiete dieser Agenten? Welche Arbeit können sie dem Menschen abnehmen und wo können sie ihn unterstützen? Welche Schwierigkeiten gilt es zu beseitigen? Abschnitt 2.1, insbesondere 2.1.2 (Seite 19) erklärt, was unter Informationsagenten verstanden wird. Im Kapitel 3 (Seite 37) wird ein Informationsagentensystem vorgestellt, das personenbezogene Informationen aus bestimmten Quellen des WWW extrahiert. Diese Agenten versorgen ein Informationssystem, das als Informationsbörse einer Konferenz dient.

1.1 Motivation

Als Beispielsszenario für diese Informationsagenten dient eine Konferenz, die im Folgenden beschrieben wird:

> In der Zukunft liegt eine Konferenz, die ihre Besucher in »zwei Welten« erwartet – der realen und der virtuellen.
>
> Die Konferenzteilnehmer melden sich zu dieser Konferenz an. Sie sind entweder aktive Teilnehmer, die Vorträge halten oder Workshops veranstalten, einfach nur Besucher der Konferenz – oder beides. Jeder Besucher erhält an den Konferenztagen einen kleinen Apparat zum Umhängen – den sogenannten »*wearable Parrot*[1]« – der natürlichsprachlich von aktuellen Ereignissen berichtet, auf interessante Veranstaltungen hinweist, oder Verabredungen mit anderen Gästen vorschlägt – alles entsprechend einem persönlichen Profil, dass für den Teilnehmer erstellt wurde und unter Anderem Ähnlichkeiten zwischen den Personen repräsentiert.
>
> Dem Konferenzteilnehmer ist sicherlich bewusst, dass viele Helfer für eine Konferenz nötig sind, die zum großen Teil ungesehen im Hintergrund arbeiten. Ist ihm aber auch bewusst, dass eine ganze Reihe von »virtuellen Helfern« im digitalen Hintergrund für den reibungslosen Ablauf dieser Konferenz sorgen? Jeder Konferenzteilnehmer hat einen persönlichen Agenten, der ihn in der »anderen Welt« vertritt. Diese Agenten helfen den Teilnehmern bei der Planung und Einhaltung von Terminen und Treffen.

Diese Konferenz ist Teil des europäischen Projekts COMRIS, das seit 1997 die derzeitigen Möglichkeiten zur Kombination dieser beiden Welten unter-

[1] Tragbarer Papagei

sucht. Verschiedene Organisationen und Institutionen in ganz Europa arbeiten an diesem Projekt. Der Lehrstuhl für künstliche Intelligenz an der Universität Dortmund ist mit dem Aufbau des unterstützenden Informationssystems – dem sogenannten *Information Layer* betraut. Dieses System wird in Abschnitt 2.3 genauer erklärt.

1.2 Ziele

Ziel dieser Arbeit ist, für das beschriebene Szenario die Informationsagenten zu implementieren. Sie sollen personenbezogene Daten über die Konferenzteilnehmer sammeln und diese Informationen in einem zentralen Informationssystem ablegen. Durch die gesammelten Daten soll ein Teilnehmerprofil entstehen, mit dem die Ähnlichkeit der Personen bewertet werden kann (siehe Abbildung 1.1). Diese Bewertung ist natürlich direkt abhängig von der Art der gefundenen Daten. Nur wenn diese Daten bekannt sind, kann entschieden werden, welche Aspekte verfüg- und bewertbar sind.

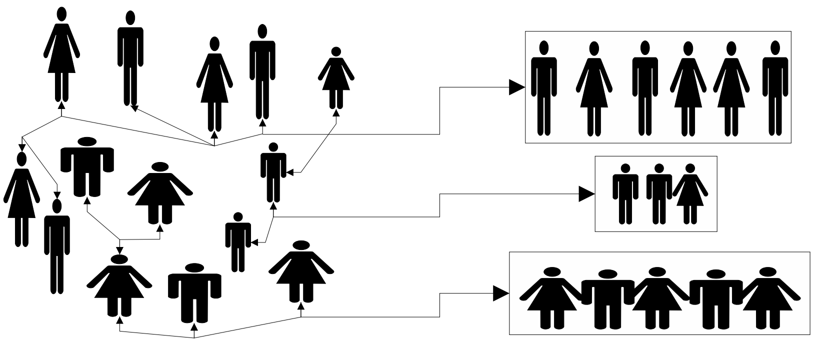


Abbildung 1.1: Ähnlichkeitsbewertung der Konferenzteilnehmer. Die Pfeile symbolisieren Aspekte, die bestimmte Teilnehmer verbinden. Diese Aspekte können aus verschiedenen Dimensionen sein, z. B. wäre Körperform ein Vergleichsmerkmal.

Schließlich sollte das System jedem Konferenzteilnehmer Empfehlungen bezüglich seiner Kontakte machen können. Idealerweise in der Art eines Orakels, das alle Beziehungen kennt und aus diesen die Empfehlungen generiert.

Die Aufgabe ist also, verschiedene Datenquellen strukturiert in einem zentralistischen Informationssystem zusammenzuführen und zur Wissensentdeckung zu benutzen – *Data Mining* auf dem WWW.

Zusammenfassend ergeben sich folgende Fragen:

- Welche personenspezifischen Daten sind im WWW verfügbar und relevant?
- Welche Aspekte ergeben sich aus diesen Daten?

- Wie können diese Aspekte genutzt werden, um Schlüsse zu ziehen bzw. Regeln zu lernen?

1.3 Gliederung

Diese Diplomarbeit lässt sich grob in zwei Teile gliedern:

Der erste Teil dieser Diplomarbeit befasst sich mit der Realisierung eines Agentensystems zur Suche und Extraktion relevanter, personenbezogener Informationen aus bestimmten Quellen des WWW. Im folgende Kapitel werden zunächst Grundlagen erklärt, die zum Verständnis dieser Arbeit wichtig sind. Abschnitt 2.1 erläutert, was unter Agenten und Agentensystemen verstanden wird. Da es in der Literatur keine einheitliche Definition dieses Begriffes gibt, wird ein kleines Spektrum der gängigen Klassifizierungen gezeigt. Abschnitt 2.2.4 (Seite 30) beschreibt MOBAL, ein System, das verschiedene Tools zum Aufbau und zur Pflege von Wissensbasen bereiststellt.

Kapitel 3 (Seite 37) enthält den groben Entwurf dises Systems und in Kapitel 4 (Seite 49) werden Details zur Realisierung dieses Entwurfes gegeben. Alle Ergebnisse werden schließlich in Kapitel 5 (Seite 61) zusammengefasst.

Zunächst werden relevante, personenbezogener Informationen aus bestimmten Quellen des WWW extrahiert und in das Informationssystem eingetragen. Die gefundenen Informationen werden mit dem Informationssystem *Information Layer* synchronisiert, d. h. Daten werden entweder neu erfasst bzw. ergänzt. Der Information Layer dient dem Informations-Management des EG Projekts COMRIS und bildet die Brücke zwischen dem realen und virtuellen Informationsraum. Abschnitt 2.3 (Seite 33) erklärt die zum Verständnis dieses Systems nötigen Grundlagen.

Den zweiten Teil bildet die Ähnlichkeitsbewertung. Ähnlichkeitsbewertung heißt hier: Analyse bestimmter Merkmale der von den Informationsagenten gefundenen Informationen. Diese Informationen werden in eine geeignete Repräsentation gebracht, um mit maschinellen Lernverfahren auf eine »Entdeckungsreise« zu gehen. Das in dieser Arbeit benutzte Lernverfahren wird in Abschnitt 2.2.3 (Seite 29) vorgestellt. Mit Hilfe von Trainingsdaten soll untersucht werden, ob Regeln gelernt werden können, die sich auf Relationen zwischen den Konferenzteilnehmern beziehen und so ein Ähnlichkeitsmaß ausdrücken.

Im Anhang B (Seite 81) werden Details zu den einzelnen Datenquellen der Agenten angegeben.

Kapitel 2

Grundlagen

Dieses Kapitel führt die für das Verständnis grundlegenden Techniken ein. Abschnitt 2.1 gibt einen Überblick des derzeitigen Agentenbegriffes. In der Literatur werden Agenten und Agentensysteme häufig durch ihre Funktion und Anwendung charakterisiert. Exakte Definitionen sind hier schwierig, wie auch keine exakte Definition des Begriffes »Künstliche Intelligenz« möglich ist.

Im Abschnitt 2.2 wird der Bereich des maschinellen Lernens beschrieben, der in dieser Arbeit Anwendung findet.

Der Information Layer, in den das Agentensystem dieser Arbeit integriert ist, wird in Abschnitt 2.3 erläutert.

2.1 Agentensysteme

Der folgende Abschnitt 2.1.1 erläutert zunächst kurz, was unter Software-Agenten zu verstehen ist, Abschnitt 2.1.2 (Seite 19) betrachtet dann die Teilmenge der Informationsagenten. Einige Beispiele runden das Kapitel in Abschnitt 2.1.3 ab.

2.1.1 Agenten

Für [Minsky, 1985] ist ein Agent nicht intelligent – aber die Summe dieser nicht intelligenten Agenten macht Intelligenz aus: Das Ganze ist mehr als die Summe seiner Teile. Inzwischen wird der Begriff »Agent« in den verschiedenen Forschungsbereichen, wie verteilte Systeme, Software Engineering und der künstlichen Intelligenz (KI) unterschiedlich gebraucht.

In der KI sind es vor allem die Arbeiten von [Wooldridge und Jennings, 1995], [Bradshaw, 1997] und [Nwana, 1996], die die Thematik einführen und grundlegend sind. Doch selbst in diesem Bereich herrscht keine Einigkeit bei einer genauen Definition.

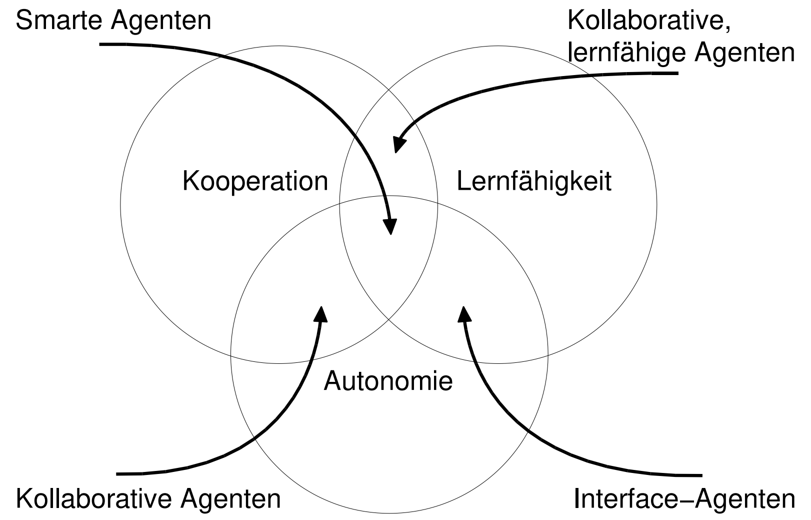


Abbildung 2.1: Einteilung verschiedener Agentenarten nach [Nwana, 1996].

Das Wort »Agent« löst bei jedem Menschen direkt eine intuitive Vorstellung von der Bedeutung aus. Diese Vorstellung kann sehr facettenreich sein und ist von Mensch zu Mensch sicherlich verschieden. Man denkt z. B. an Versicherungsagenten, Agenten von Schauspielern oder an James Bond. All diesen Personen ist gemein, dass sie für andere aktiv werden, indem sie Aufträge erledigen oder auf andere Weise assistierend auftreten. Sie sind auf bestimmte Dinge spezialisiert, wissen z. B. wo Flüge günstig gebucht werden können, oder haben spezielle Kontakte, die es ihnen ermöglichen, gewisse Sachen zu organisieren oder zu beschaffen. So gibt es die Reiseagentur und die Nachrichtenagentur. Die Nachrichtenagentur beschäftigt Agenten, die sich um Nachrichten, also um Informationen kümmern. Diese Agenten (in diesem Zusammenhang auch Korrespondenten genannt) sind in der Regel auf der ganzen Welt verteilt, pflegen ihrerseits Kontakte zu sogenannten *Informanten* und sorgen dafür, dass aktuelle Informationen zur Agentur geleitet werden, wo sie an entsprechende Abnehmer weiterverteilt werden.

Die Tätigkeiten von Agenten der wirklichen Welt werden nun in die virtuelle Welt übertragen. Der Begriff »Agent« wird als eine Metapher für Programme verwendet, »*deren Funktion und Art der Ausführung in einer offenen und verteilten Umgebung mit bestimmten menschlichen Eigenschaften wie autonomes, zielgerichtetes Verhalten, soziale Kooperation, Intelligenz und Lernen verbunden ist* ([Klusch und Benn, 1998]).«

Durch Einordnung in Aufgabenbereiche und -felder werden in der Literatur Agenten definiert. [Nwana, 1996] entwirft eine Typologie, die es erlaubt, Agenten je nach Eigenschaft zu klassifizieren (Abbildung 2.1).

Grundlegende Eigenschaften sind demnach:

1. Die Fähigkeit zur Kooperation,

2. Lernfähigkeit und
3. Autonomie.

Diese Eigenschaften können durchaus unterschiedlich ausgeprägt sein, je nach Aufgabe des Agenten. Kollaborative, lernfähige Agenten brauchen natürlich die ersten beiden Eigenschaften, Interface-Agenten sollten lernfähig (adaptiv) und autonom sein. Damit ein Agent wirklich *smart*[1] ist sollte er alle drei Eigenschaften kombinieren.

Autonomie und Kooperation sind auch nach [Foner, 1997] zwei der wichtigsten Kriterien, die aus einem Stück Software einen Agenten machen. Er fordert außerdem die Erfüllung verschiedener weiterer Merkmale, die theoretisch sicherlich sinnvoll sind, in bestehenden Systemen jedoch kaum Verwendung finden. So z. B. die Vorstellung, zwischen Risiko und Vertrauen abwägen zu müssen. D. h. man muss seinem Agenten trauen, dass er auch die eigenen Interessen vertritt und uns nicht betrügt.

Nwana nennt außerdem noch andere Kriterien, mit denen Agenten unterschieden werden können. Sind sie mobil oder statisch? Bewegen sie sich z. B. in einem Netzwerk, oder befinden sie sich immer auf einem Rechner? Haben sie die Fähigkeit zum selbsttätigen Handeln oder reagieren sie nur auf Änderungen ihrer Umgebung? Selbsttätiges Handeln setzt voraus, dass die Agent über eine interne Symbolik und ein Schlußfolgerungsmodell verfügen.

Nach Nwana gibt es insgesamt sieben Agententypen:

- kollaborative Agenten,
- Interfaceagenten,
- mobile Agenten,
- Informations-/Internetagenten,
- reagierende Agenten,
- hybride Agenten und
- smarte Agenten.

[Franklin und Graesser, 1997] schlagen eine Einteilung nach Abbildung 2.2 vor. Autonome Agenten bilden die Wurzel eines Baumes und alle anderen Agententypen sind Unterarten dieses Typs. Unterhalb dieser Wurzel befinden sich Viren auf einer Ebene mit aufgabenspezifischen und Unterhaltungsagenten. Auf Grund der vielen Definitionen die es für Agenten gibt, nennen auch sie einige Eigenschaften, die Agenten ausmachen:

- Reaktiv: Agenten beobachten ihre Umwelt und reagieren gegebenenfalls.

[1] Klug, intelligent.

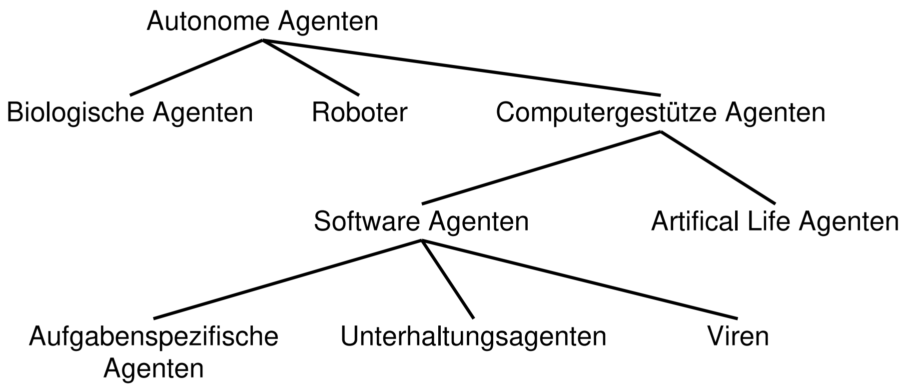


Abbildung 2.2: Agententaxonomie nach [Franklin und Graesser, 1997].

- Autonom: Sie haben die Kontrolle über ihre Handlungen.
- Ziel-orientiert: Agenten handeln nicht nur einfach als Reaktion auf die Umgebung.
- Kommunikativ: Sie kommunizieren mit anderen Agenten oder Menschen.
- Anpassungsfähig: Sie ändern ihr Verhalten auf Grund vergangener Erfahrungen.
- Mobil: Agenten können den Rechner wechseln.

Da eine Aufgabe verschiedene Anforderungen an Agenten stellen kann, nennen [Beale und Wood, 1994] sechs Attribute die Agenten ausmachen und die charakteristisch für einige Aufgabetypen sind:

- Lernenfähigkeit: Der Agent passt sich der Aufgabe an und kann so seine Leistung steigern.
- Forschungsfähigkeit: Eine Aufgabe kann schlecht umrissen sein, der Agent muss verschiedene Möglichkeiten ausprobieren, bis er die Aufgabe bewältigen kann.
- Vorführend: Der Agent zeigt einem Anwender wie ein bestimmtes Programm zu bedienen ist.
- Ratgebend: Der Agent kooperiert mit dem Benutzer um ihm bei der Erledigung einer Aufgabe zu helfen.
- Autonomie: Siehe oben.
- Asynchronität: Die Aufgabe kann Verzögerungen unterliegen. Es ist z. B. möglich, dass ein Netzservice ausfällt und die Aufgabe aufgeschoben werden muss.

Agentenbasiertes *Data Mining* stellt einen Aufgabentyp dar, der viele dieser Attribute verwendet. Dabei wird versucht, Datenbanken im Hinblick auf Beziehungen zu untersuchen, die zwischen Variablen, oder Ursachen und Wirkungen bestehen. Einige Probleme machen diese Aufgabe nicht-trivial. Z. B. ist es schwierig, eine Datenbank zu durchsuchen, die viele verschiedene Variablen in verschiedenen Dimensionen beinhaltet, denn es lassen sich nie mehr als drei Dimensionen zur gleichen Zeit graphisch dargestellt [Beale und Wood, 1994]. Inzwischen ist es kein Problem mehr, riesige Datenmengen zu speichern, im Gegenteil – je größer der Datenbestand wird, umso wahrscheinlicher ist es, dass Beziehungen innerhalb dieser Datenbasis vorhanden sind.

2.1.2 Informationsagenten

Gegenwärtig ist die wissensbasierte Suche nach relevanten Informationen im Internet durch intelligente Informationsagenten sehr attraktiv. Die Suchmaschinen der Gegenwart werden längst nicht mehr den Wünschen der Anwender gerecht. In der Zukunft wird der Bedarf an *Informations-Management* nicht mehr mit herkömmlichen Strategien bewältigt werden können [Liebermann, 1998]. Die Aufgabe des Suchens, Organisierens, Verwaltens und des Präsentierens sollte im Auftrag des Anwenders von Informationsagenten übernommen werden.

Traditionell ist Informations-Management eine Aufgabe des *Information Retrieval* (IR). Doch IR setzt sich mit relativ statischen Informationsquellen auseinander. Viele Informationen im Web sind nicht textbezogen und daher versagen hier die klassischen IR-Techniken. Andererseits basieren viele Suchmaschinen auf IR: AltaVista, Lycos, Excite, Infoseek und HotBot benutzen alle die Häufigkeit von Stichwörtern um das Thema eines Dokuments zu bestimmen. Hier wird auf die Fähigkeit des Anwenders gesetzt, eine gezielte Anfrage zu stellen, und gegebenenfalls diese Anfrage nach und nach zu verfeinern [Salton, 1989]. Informationsagenten ersetzen nicht die IR-Techniken – sie integrieren sie.

Unter einem Informationsagenten versteht man also einen Agenten, der in einem externen *Informationsraum* tätig ist und dessen Aufgabe es ist, dem Anwender beim Zugriff und der Verwaltung dieser Informationsquelle und der von ihr bereitgestellten Informationen zu unterstützen. Diese Informationsräume sind z. B. durch das Web verbundene Datenbanken. Die Informationen, die im WWW verbreitet werden sind aber höchst unstrukturiert, z. B. natürlichsprachlicher Text oder Bilder – nur für den Menschen verständlich. Das Spektrum für den Einsatz von Informationsagenten ist daher so weit wie die Arten von Informationen und so verschieden wie der Informationsbedarf der Anwender.

Ein Informationsagent soll z. B. für einen Benutzer aktiv und autonom nach relevanten Informationen in einer Menge von Informationsquellen suchen, die für ihn verfügbar sind. In der Praxis ist es oft so, dass für jede Daten-

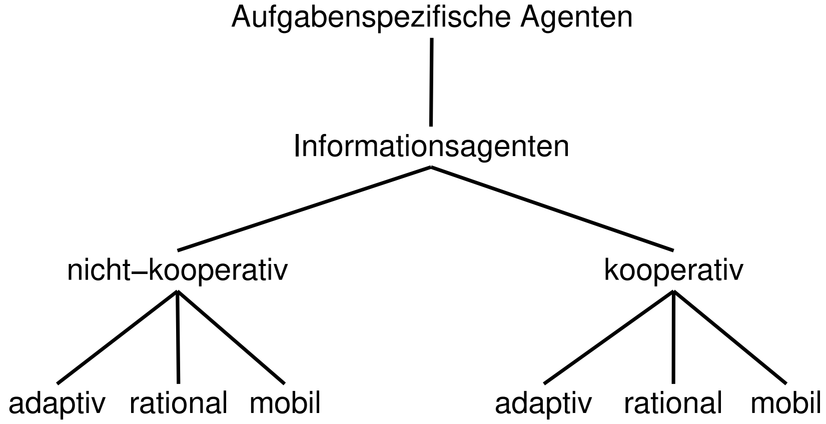


Abbildung 2.3: Mögliche Klassifizierung von Informationsagenten.

quelle ein speziell »abgerichteten« Agenten eingesetzt wird, so dass mehrere Agenten kollaborativ zusammenarbeiten. Dies wird dann Agentensystem oder Multi-Agentensystem genannt [Kinny und Georgeff, 1997].

Die Einteilung von [Franklin und Graesser, 1997] (Abbildung 2.2) wird von [Klusch und Benn, 1998] noch weiter verfeinert, indem sie an das Blatt der aufgabenspezifischen Agenten noch einen Unterbaum für Informationsagenten hängen (Abbildung 2.3).

Sie unterteilen die Klasse der Informationsagenten in kooperative und nicht kooperative und innerhalb dieser Klassen jeweils in adaptive, rationale und mobile Agenten.

Nicht-kooperative Informationsagenten

Diese Agenten kooperieren nicht untereinander um die Aufgabe des Anwenders zu erfüllen. Die Voraussetzungen für eine erfolgreiche Suche ist ein umfassendes Wissen über die Struktur und Bedeutung der Informationsquellen.

Kooperative Informationsagenten

Hier wird jeder Informationsquelle ein Agent zugeordnet. Handelt es sich bei der Quelle um eine heterogene Datenbank, können beim Zugriff Standards wie CORBA [Redlich, 1996], ODBC [2] oder JDBC [3] verwendet werden. Anwendungsabhängige Kooperationen erfolgen mit Hilfe dieser Agenten. Die be-

[2] Diese Abkürzung steht für *Open DataBase Connectivity*. Es handelt sich dabei um ein genormtes Protokoll, mittels dessen Anwendungen auf Datenbanksysteme zugreifen können. Als Abfragesprache dient SQL (*Structured Query Language*).

[3] JDBC steht für *Java DataBase Connectivity* und ist genau wie ODBC ein Protokoll zum Zugriff auf Datenbanksysteme. Der Unterschied zu ODBC ist lediglich, dass JDBC-Treiber in Java programmiert und damit plattformunabhängig sind.

sondere Eigenschaft dieses Agententyps ist die Fähigkeit, bei Bedarf mit anderen Agenten zusammen zu arbeiten.

Adaptive Informationsagenten

Ein adaptiver Agent besitzt die Fähigkeit, sich seiner Umgebung und seinen Anforderungen je nach Aufgabengebiet anzupassen, beispielsweise der wechselnden Relevanz der Informationen für einen Anwender. Die Adaptivität wird in der Regel durch den Einsatz von maschinellen Lernverfahren erreicht [Weiß und Sen, 1996]. Hier stellen sich folgende Fragen:

- Welches Lernverfahren soll für welche Anwendung eingesetzt werden?
- Soll das Lernen auf einen einzelnen Agenten oder auf das ganze Agentensystem angewendet werden?
- Welche Art von Wissen über sich und die Umgebung sind für den Agenten wichtig?
- Können Methoden der Wissensentdeckung in Datenbanken eingesetzt werden?

Rational kooperative Informationsagenten

Dieser Agententyp erwartet »Lohn« für seine Dienste. Durch die fortschreitende Kommerzialisierung des WWW werden diese Agenten in Zukunft eine sehr wichtige Bedeutung erlangen und nach dem Motto »Geld gegen Information« agieren. Solche Systeme stellen eine geeignete Softwaretechnologie für den elektronischen Handel dar [Klusch und Benn, 1998].

Mobile Informationsagenten

Das sind Informationsagenten, die sich physikalisch im WWW von einem lokalen Informationssystem zum anderen bewegen [Bradshaw, 1997]. Gegenwärtig spielt dieser Agententyp keine große Rolle da unter anderem der Sicherheitsaspekt noch Klärung bedarf. Ein mobiler Agent macht technisch keinen Unterschied zu einem Virus.

Match-Making-Agenten

Informationen sind jedoch auch immer mit den Personen verknüpft, die sie produzieren oder eine Interesse daran haben. So ist es manchmal wichtig Personen zu finden, die ein Intersse teilen. Die Suche nach etwas »Passendem« wird als *match-making* bezeichnet. Und so treten Informationsagenten bisweilen auch als Match-Making-Agenten auf. Diese Agenten stellen z.

B. fest, welche Interessen Mitglieder einer Gruppe verbinden, oder finden fremde Personen, die ein Interesse teilen.

2.1.3 Beispiele

Die bekannteste Art von Informatinsagenten sind gegenwärtig die sogenannten *Softbots* [Etzioni und Weld, 1994] – einzelne Agenten, die dem Anwender bei seiner Suche im WWW assistieren, dabei aber nicht mit anderen Agenten kooperieren. Tabelle 2.1 enthält die URLs einiger Agenten[4].

Name	URL
Amalthaea	http://lcs.www.media.mit.edu/projects/amalthaea/
bizzyB	http://zeus.gmd.de/cobra/cobra4.html
ELFI	http://zeus.gmd.de/projects/elfi.html
Kasbah	http://kasbah.media.mit.edu/
Letizia	http://lieber.www.media.mit.edu/people/lieber/ Lieberary/Letizia/Letizia.html
NewsWeeder	http://www.wisewire.com/
Pleiades	http://www.cs.cmu.edu/˜softagents/pleiades/index.html
RETSINA	http://www.cs.cmu.edu/˜softagents/retsina/retsina.html
Warren	http://www.cs.cmu.edu/˜softagents/warren/
WebWatcher	http://www.cs.cmu.edu/˜webwatcher
Yenta	http://foner.www.media.mit.edu/people/foner/ yenta-brief.html

Tabelle 2.1: Die URLs einiger im WWW ausführbarer Informationsagenten bzw. Agenten-Toolkits.

ELFI und bizzyB

Information Brokering steht in direktem Zusammenhang mit Informationsagenten. [Koenemann und Thomas, 1998] definieren Information Brokering wie folgt:

> *The mediation of information transactions between providers and consumers in order to fulfill participants' goals.*

Sie benutzen die Begriffe *Informationsbedarf* und *Informationsobjekte*, um ihr System genauer zu beschreiben. So ist bekannt, dass die Anfragen eines Anwenders nicht mit seinem eigentlichen Infromationsbedarf übereinstimmen und dieser sich im Laufe des Suchprozesses durchaus ändern kann [Belkin und Coft, 1992]. Demnach ist Information Brokering kein einzelner

[4] In der Regel sind hier nur Dienste verfügbar, nicht die Agenten selbst.

Match-Making-Prozess, sondern ein interaktives Ereignis – technische Lösungen müssen diese Dynamik beachten, wenn sie erfolgreich sein wollen. Ein Brokering-System hat Daten seiner Nutzer in Form von Profilen und Fallbeschreibungen. Die Informationsobjekte bestehen zum einen aus den eigentlichen Daten (*source data*) und zum anderen aus Beschreibungen dieser Daten (*meta data*). Meta Daten eines Dokuments sind z. B. Titel, Autor und *abstract*. Der Informations-Brokering-Prozess setzt sich aus verschiedenen Unterprozessen zusammen, von denen einige in den Bereich der Informationsagenten fallen. Beim *broadcasting* wird eine Anfrage an alle bekannten Informationsanbieter übertragen. Die *collection* befasst sich mit der darauf folgenden Sammlung der Informationsobjekte, die anschließend durch *filtering* auf die Bedürfnisse des Anwenders angepasst werden.

[Koenemann und Thomas, 1998] unterscheiden drei Aufgaben, die durch den Einsatz von Agenten gelöst werden können:

1. Die Ausführung von Routine-Aufgaben, z. B. das Überwachen von bestimmten Web-Ressourcen und entsprechendes Aktualisieren von Informationen.
2. Die Ausführung von speziellen aber nebensächlichen Aufgaben. Beispielsweise hat ein Agent spezielles Wissen über den Zugriff einer bestimmten Online-Datenbank.
3. Zur Hilfe und Anleitung eine bestimmte Aufgabe zu lösen.

ELFI ist eine Information Brokering-Umgebung im Bereich öffentliche Mittel. Es geht darum, dass Universitäten und andere Forschungsbetriebe oft Geldgeber suchen und deswegen einen Information Broker beschäftigen. Dieser Broker sucht verschiedene Server nach neuen Geldquellen ab, sammelt gefundene Informationen, verteilt sie an Abnehmer und unterstützt die Antragsstellung. Die Agenten müssen dabei entscheiden wann eine Information neu ist, sie müssen neue Geldquellen identifizieren und entscheiden, ob ein bestimmtes Förderprogramm für einen bestimmen Fall relevant ist.

Die Informationsobjekte in ELFI sind textuelle Beschreibungen von Programmen zur Forschungsförderung, Aufrufen zur Antragstellung, Bewilligungen, etc. Diese Objekte werden von den Anbietern zum zentralen Broker übertragen, wo sie weiter verarbeitet werden (Indizierung, Bildung der Metadaten). Agenten überwachen die Anbieter, um die zentrale Datenbank aktuell zu halten. Einige Agenten unterstützen einen menschlichen Broker bei der Kategorisierung der Daten und der Erzeugung der Metabeschreibungen. Schließlich werden die Informationen agentenbasiert verteilt.

bizzyB ist eine Broker-Umgebung des »Milan Chamber of Commerce«, die auf Geschäftsinformationen und Akkreditierung[5] spezialisiert ist. Die In-

[5] Kreditbeschaffung.

itiatoren erwarten, dass durch die Einführung des Brokers die Qualität und Aktualität der Informationen, die den Kunden bereitgestellt werden, erhöht wird. Informationsobjekte sind Daten über Betriebe – Name, Größe, Ort, Rechtsform und die Art des Angebots.

RETSINA

RETSINA [Klusch und Benn, 1998] ist die Bezeichnung für ein Multiagentensystem, in dem verschiedene Typen von Agenten zusammenarbeiten um relevante Informationen im Internet zu finden. Es gibt Interface-, Aufgaben- und Informationsagenten. Die Interfaceagenten stellen den Zugang für Anwender bereit, Aufgabenagenten planen und kontrollieren die Ausführung von Aufgaben, die sie von den Interfaceagenten erhalten. Die Informationsagenten bearbeiten die Anfragen die sie von den Aufgabenagenten erhalten. Hierbei kann es sich insbesondere um Matchmaking-Agenten handeln. Beispiele für RETSINA-Agentensysteme sind Warren – Handel und Informationen über Aktien an verschiedenen Börsen, Pleiades – Terminplanug und Büroautomation und Thales – Vorhersage der Position bzw. Sichtbarkeit eines Satelliten. Adaptive Techniken aus dem Gebiet des maschinellen Lernens werden gegenwärtig vor allem bei nicht-kooperativen Informationsagenten eingesetzt. Ein Beispiel hierfür ist WebWatcher.

WebWatcher

WebWatcher (URL in Tabelle 2.1) [Armstrong et al., 1995] ist ein sogenannter *Tour-Guide*. Er unterstützt den Anwender bei der Suche im WWW, indem er ihm Links vorschlägt, von denen WebWatcher glaubt, sie könnten den Anwender interessieren. Der Anwender kann diese Vorschläge bewerten und so WebWatcher Rückmeldung über seine Leistung geben. So gesehen ist WebWatcher ein Interface-Agent. Für jeden Hyperlink in einer HTML-Seite entscheidet WebWatcher, ob der Benutzer diesem Link folgen wird oder nicht. Diese Entscheidung ist abhängig vom Interesse des Benutzers. Dabei dienen Links, denen der Anwender gefolgt ist als positive Beispiele.

Letizia

Ein anderer Informationsagent ist Letizia (Tabelle 2.1)[Liebermann, 1998]. Liebermann klassifiziert seinen Agenten als einen »Erkundungs«-Agenten. Er betrachtet jeden Link einer Webseite als unbekanntes Territorium und das Verfolgen dieses Links als einen Sprung ins Ungewisse, zumindest was den Informationsgehalt der gelinkten Seite betrifft. Ein Informationsagent sorgt hier dafür, dass dem Anwender interessante Links vorgeschlagen werden. Letizia wird zusammen mit einem Web-Browser betrieben und nutzt maschinelles Lernen, um ein Benutzerprofil zu erstellen, damit es vernünftige Vorschläge machen kann. Der Agent merkt sich die Dokumente die der Anwender liest und unterstellt ein Interesse an diesen Dokumenten ohne dabei eine Bewertung des Nutzers zu erwarten. Letizia nutzt die Links auf Webseiten aus, da diese oft die Relevanz von anderen Dokumenten repräsentieren. Miteinander verknüpfte Seiten stellen eine semantische Nachbarschaft dar. Allerdings bietet Letizia keine Unterstützung, wenn man nach bestimmten Informationen sucht. Hier wird eine Suchmaschine bessere Ergebnisse liefern, die dann mit Hilfe von Letizia effizienter durchsucht werden können.

Amalthaea

Alex Moukas [Moukas, 1996] verbindet hier eine ganze Reihe von Informationsagenten-Techniken (URL in Tabelle 2.1). Vorschläge, die sich mit dem Interesse des Nutzers decken, resultieren aus den Interaktionen mehrerer interner Informationsagenten. Diese internen Agenten teilen sich auf in Entdecker-Agenten (*discovery agents*), die Vorschläge machen, und Filteragenten, die bestimmte Vorschläge aussuchen. Amalthaea verbindet also Agenten wie Letizia oder WebWatcher mit Multi-Agentensystemen. Amalthaea integriert, wie WebWatcher, die Bewertung des Anwenders in seine Vorschläge. Der Benutzer richtet seine Anfrage an einen oder mehrere Filteragenten. Diese Anfrage wird mit Hilfe von IR-Verfahren in einen Vektor von numerisch gewichteten Stichworten übersetzt. Die Entdeckungsagenten versuchen dann mit Hilfe dieser Vektoren auf den Suchmaschinen im WWW Dokumente zu finden, die für die Anfrage der Filteragenten relevant sind. Die Filteragenten suchen sich unter den Kandidaten der Entdeckungsagenten wiederum die relevantesten Dokumente aus, die dann dem Benutzer präsentiert werden. Der Benutzer bewertet das Ergebnis seiner Anfrage und so kann sich das System von Mal zu Mal verbessern.

Kasbah-Agenten

Wie der Name schon sagt[6], handelt es sich hier um Informationsagenten, die eine profitorientierte Suche im Internet durchführen. Ein »virtueller Marktplatz« bildet die Umgebung und An- und Verkaufangebote sind die relevanten Informationen dieser Agentenart. Jeder Agent bietet eine bestimmte Ware zum Verkauf an und/oder versucht eine bestimmte Ware für seinen Benutzer zu kaufen. Ein Handel zwischen zwei Agenten kommt zustande, nn beide ihre gegenseitigen Angebote akzeptieren. Dabei wird die gesamte Kommunikation vom »Marktplatz« organisiert. Anfragen werden an ihn gestellt, Angebote werden ihm mitgeteilt.

In diesem Zusammenhang ist das sogenannte »Fischmarkt«-Problem zu erwähnen: Ein Verkäufer macht ein Angebot und wartet eine gewisse Zeit, ob sich ein Käufer meldet. Gibt es keinen Käufer verringert er den Verkaufspreis und wartet erneut. Die Prozedur dauert solange, bis sich ein Käufer meldet.

[6] Kasba[h] oder Kasabi – arabisches Viertel in nordafrikanischen Städten.

Verkauft er die Ware nicht, kommt sie vom Tisch. Die interessierten Käufer wollen die angebotene Ware mit den geringsten Kosten einkaufen. Warten sie jedoch zu lange, kann es sein, das sich ein anderer Käufer die Ware schnappt. Ein Agent muss also lernen, im richtigen Moment zuzugreifen. Wann ist aber dieser Moment gekommen?

2.2 Maschinelles Lernen

Einige Agenten (z. B. WebWatcher) haben die Eigenschaft, dass sie adaptiv, bzw. lernfähig sind. Diese Lernfähigkeit wird mit Verfahren des maschinellen Lernens erreicht. Das maschinelle Lernen versucht nicht menschliches Lernen auf Rechner zu übertragen, obwohl es sich im Prinzip ähnelt. Vielmehr geht es darum, eine Lernaufgabe so zu modellieren, das sie in einer geeigneten Repräsentation maschinell gelernt werden kann. Der Mensch hat eine eigene Weise sein Wissen zu repräsentieren und die Maschine repräsentiert ihr *Wissen* auf ihre Art. Die Psychologen benutzen ganz bestimmte Modelle, um Wahrnehmung und Lernen beim Menschen zu erklären. Diese Modelle können bis zu einer gewissen Komplexität mit Rechnern modelliert und nachempfunden werden. So kommt es, dass die Psychologen inzwischen die streng formalisierten Modelle der Informatiker benutzen, um die menschliche Kognition zu erklären [Metz-Göckel, 1997].

Richard Bandler[7] und John Grinder[8] [Bandler und Grinder, 1996] haben ganz bestimmte Verhaltensweisen von Menschen modelliert. Dabei hatten sie folgende Idee: Warum soll man jahrelang durch Versuch und Irrtum Erfahrungen und Fertigkeiten erwerben, wenn man die Essenz dieser Lehrjahre auch dadurch gewinnen kann, indem man die Handlungsmuster von Menschen nachmacht, die bereits über die gewünschten Fähigkeiten verfügen? Mit ihrem Ansatz konnten sie erstaunliche Erfolge erzielen. So haben sie z. B. die Handlungsmuster einer bekannten Familientherapeutin[9] modelliert und sie an ihre Studenten weitergegeben. Sie wendeten diese Muster an und konnten die gleichen Ergebnisse wie diese Therapeutin erzielen, auch wenn sie nicht deren jahrelange Erfahrung besaßen.

Lernen wird in der Psychologie wie folgt definiert [Metz-Göckel, 1997]:

Definition 2.1 *Lernen ist ein Prozess, der zu relativ dauerhaften Veränderungen des Verhaltens und Erlebens bzw. zu neuen Verhaltens- und Erlebnisweisen führt.*

Eine Definition für das maschinelle Lernen kommt von [Michalski, 1986]:

Definition 2.2 *Lernen ist das Konstruieren oder Verändern von Repräsentationen von Erfahrungen.*

[7] Mathematiker, Gestalttherapeut und Computerfachmann
[8] Linguist
[9] Virginia Satier

Beiden Definitionen ist gemein, dass sie von Konstruktion oder Veränderung von Erfahrungen handeln. Der Lernprozess wird in beiden Fällen durch Beobachtungen beeinflusst. Die veränderte Repräsentation von Erfahrungen sorgt beim Menschen dafür, dass er sein Verhalten bzw. sein Erleben verändert.

[Mitchell, 1997] definiert maschinelles Lernen so, dass es anwendbar wird:

Definition 2.3 *Ein Programm* lernt *aus Erfahrung* `E` *in Bezug auf Aufgabe* `T` *und einem Leistungsmaß* `P`*, wenn es seine Leistung* `P` *in Bezug auf Aufgabe* `T` *durch die Erfahrung* `E` *verbessert.*

Beispielsweise verbessert ein Datenbanksystem die Anwortzeit (`P`) auf Anfragen (`T`) auf Grund der Erfahrungen (`E`), die es aus den Aktualisierungen der Benutzer hat.

Bei einer Lernaufgabe müssen also `E`, `T` und `P` klar definiert sein. Im Kontext dieser Diplomarbeit bezeichnet:

`T` das Lernen von Regeln für die Vorhersage eines Zielbegriffes.

`P` die Bewertung der gelernten Regeln.

`E` die Beispiele, an Hand derer die Regeln gelernt werden sollen.

Da es verschiedene Lernalgorithmen gibt, die jeweils bei verschiedenen Lernaufgaben vorteilhaft bzw. weniger vorteilhaft sind, muss zunächst herausgefunden werden, von welcher Art diese Aufgabe ist. Der schwierigste Teil besteht darin, eine geeignete Repräsentation für die Daten zu finden, denn diese legt den Lernalgorithmus fest. Während der Modellierung stellen sich dem Anwender folgende Fragen [Morik et al., 1993]:

- Was bedeuten die Aussagen in dieser Repräsentation? Wie ist ihre Semantik?
- Welche Folgerungen kann das System ziehen? Sind diese richtig und genügen sie den Anforderungen der Semantik?
- Werden diese Folgerungen in einer angemessenen Zeit geschlossen?

2.2.1 Modellierung eines Sachgebiets

Die Modellierung eines Sachgebiets (*Domäne*) durchläuft grundsätzlich drei Phasen [Morik, 1989].

In der *ersten Phase* wird die Struktur des Modells festgelegt. Welche Aspekte der Domäne sind relevant und gehören ins Modell? Man wählt das Vokabular für die Beschreibung – die Repräsentation (die natürlich auch immer vom benutzen Lernvefahren abhängt). Semantische Beziehungen zwischen

Konzepten, Eigenschaften und Zuständen werden als Grundannahmen gemacht.

Die Frage nach der Relevanz wird durch die Ergebnisse der Agenten determiniert. Es gibt nur Informationen über gemeinsame Publikationen und gemeinsame Projekte.

Der Schritt von der Domäne zum Modell, insbesondere der Entwurf der ersten Repräsentation, wird von keinem System unterstützt. In der Regel ist das Arbeit, die auf dem Papier vorbereitet wird und erst anschließend mit dem Rechner getestet wird.

In der *zweite Phase* werden dem Modell Fakten und Regeln zugefügt. Je mehr Beobachtungen (Beispiele) in das Modell einfließen, umso mehr semantische Beziehungen entstehen. Stellt man nun fest, dass sich eine Beobachtung nicht ins Modell einpassen lässt (eine Regel leitet nicht mehr diese Beobachtung ab), muss das Modell bzw. die Repräsentation revidiert werden.

Die *dritte Phase* kann auch als Testphase betrachtet werden. Hier werden Experimente entworfen, die einerseits das Modell testen, andererseits aber auch Konflikte und Lücken des Modells aufzeigen, die mit den gegebenen Informationen nicht aufgelöst werden können. Diese Konflikte können dann behoben und Lücken gefüllt werden. Das Auswerten eines Modells mit Experimenten zieht auch meistens eine Revision des Modells nach sich.

Es gibt Systeme, die dem Anwender einen Teil dieser 3-phasigen Modellierung abnehmen. Betrachtet man maschinelles Lernen als automatisches modellieren und Wissenserwerb als Modellieraufgabe, dann ist es naheliegend maschinelles Lernen in den Wissenserwerb zu integrieren.

2.2.2 Lernaufgabe

Die Modellierung der Lernaufgabe in dieser Diplomarbeit erfolgt in Prädikatenlogik. Lernverfahren, die mit dieser Art der Repräsenation umgehen können, gehören in die Klasse der *ILP* – der *induktiven logischen Programmierung.*

Hier geht es darum, für eine gegebene Menge von Daten die speziellste oder generellste Verallgemeinerung zu finden. Die gegebenen Daten (Beobachtungen oder Beispiele) sollen auf interne Beziehungen und zusammenfassende Beschreibungen untersucht werden. Wie die meisten maschinellen Lernverfahren wird auch hier vom Speziellen auf das Allgemeine geschlossen. Das wird als *induktiver Schluss* bezeichnet (Definition aus [Morik, 1999]).

Definition 2.4 (Induktiver Schluss) *Gegeben eine Menge* E *von Grundbeispielen der Form* $c_{m+1} \leftarrow c_1, \ldots, c_m$. *Die Hypothese* H *folgt induktiv aus* E *und dem Hintergrundwissen* B, *genau dann wenn:*

- $B \cup H \vdash E$,

- $\mathtt{B} \not\models \mathtt{E}$ *und*
- $\mathtt{B} \cup \mathtt{E} \not\models \neg \mathtt{H}$.

Geschrieben: $\mathtt{E} \cup \mathtt{B} \prec \mathtt{H}$.

In der induktiven logischen Programmierung werden zwei Lernaufgaben unterschieden: das Begriffslernen und das Regellernen.

2.2.3 Wissensentdeckung

Im Kontext dieser Diplomarbeit sollen Regeln gelernt werden, die Aufschluss über das Verhältnis zwischen Personen geben. Diese Lernaufgabe fällt in den Bereich der Wissensentdeckung.

Die formale Fassung der Lernaufgabe verwendet den logischen Begriff des minimalen Modells. Die folgenden Definitionen sind aus [Morik, 1999] und [Klingspor, 1998].

Definition 2.5 *Eine Interpretation* $\mathtt{I}$ *ist ein Modell einer Theorie* $\mathtt{T}$, $\mathcal{M}(\mathtt{T})$, *wenn sie für jede Aussage in* $\mathtt{T}$ *wahr ist.*

Definition 2.6 *Eine Interpretation* $\mathtt{I}$ *ist ein minimales Modell einer Theorie* $\mathtt{T}$, *geschrieben* $\mathcal{M}^+(\mathtt{T})$, *wenn* $\mathtt{I}$ *ein Modell von* $\mathtt{T}$ *ist und es keine Interpretation* $\mathtt{I}'$ *gibt, die auch ein Modell von* $\mathtt{T}$ *ist und* $\mathtt{I}' \subset \mathtt{I}$.

Dann lässt sich das Regellernproblem (Wissensentdeckung) wie folgt definieren:

Definition 2.7 (Regellernproblem) *Gegeben: Eine Menge* $\mathtt{E}$ *(Beobachtungen) in der Sprache* $\mathtt{LE}$ *mit* $\mathtt{E} \subset \mathtt{LE}$, *und eine Menge* $\mathtt{B}$ *(Hintergrundwissen) in der Sprache* $\mathtt{LB}$ *mit* $\mathtt{B} \subset \mathtt{LB}$.

Gesucht wird die Menge von Hypothesen $\mathtt{H}$ *(Regeln) in einer Hypothesensprache* $\mathtt{LH}$ *mit* $\mathtt{H} \subset \mathtt{LH}$, *für die gilt:*

1. $\mathcal{M}^+(\mathtt{B} \cup \mathtt{E}) \subseteq \mathcal{M}(\mathtt{H})$, *d. h.* $\mathtt{H}$ *ist in allen minimalen Modellen von* $\mathtt{B}$ *und* $\mathtt{E}$ *war.*
2. *Für alle* $\mathtt{h} \in \mathtt{H}$ *gibt es ein* $\mathtt{e} \in \mathtt{E}$, *so dass* $\mathtt{B}, \mathtt{E} \setminus \{\mathtt{e}\} \not\models \mathtt{e}$ *und* $\mathtt{B}, \mathtt{E} \setminus \{\mathtt{e}\}, \mathtt{h} \models \mathtt{e}$, *d. h. alle Hypothesen enthalten neue Informationen über die Beobachtungen.*
3. *Für jedes* $\mathtt{h} \in \mathtt{LH}$, *das die Bedingungen 1. und 2. erfüllt, gilt auch* $\mathtt{H} \models \mathtt{h}$, *d. h. alle in* $\mathtt{B}$ *und* $\mathtt{E}$ *wahren Hypothesen folgen aus* $\mathtt{H}$.
4. $\mathtt{H}$ *ist minimal.*

Regellernen kann als Suche in einem großen, vordefinierten Raum von potentiellen Hypothesen eingeordnet werden. Eine Teilordnung dieses Hypothesenraumes ist eine für die Suche sehr sinnvolle Struktur. Der Hypothesenraum ist natürlich abhängig von der Hypothesensprache und wird in der Regel begrenzt. Die Technik dieser Begrenzung ist abhängig vom jeweiligen Lernverfahren und wird für das hier benutzte auf der Seite 32 in Abschnitt 2.2.4 beschrieben.

2.2.4 Das System MOBAL

MOBAL ist eine Umgebung, die es dem Anwender erlaubt auf vergleichsweise komfortable Weise eine Wissensbasis aufzubauen, sie zu pflegen und zu verändern. Verschiedene Lernalgorithmen sind integriert und können auf der Wissensbasis ausprobiert und getestet werden. Im Folgenden werden die für diese Arbeit benutzten Tools vorgestellt. Allerdings ist es im Vorfeld sicherlich wichtig, einige grundlegende Begriffe zu klären, die bei der Arbeit mit MOBAL gebraucht werden (aus [Sommer et al., 1996]).

Fakten geben Relationen, Eigenschaften von Objekten und die Zugehörigkeit zu Konzepten an. Das klingt im ersten Moment recht abstrakt, wird aber durch ein Beispiel sicherlich deutlich:

```
person(felix).
persons_temperature(felix, 36.5).
```

Damit wird zum Ausdruck gebracht, dass `felix` dem Konzept `person` angehört und dass es eine Beziehung zwischen `felix` und der Zahl `36.5` gibt.

Jedem Faktum wird ein Wahrheitswert *evidence point* zugeordnet, der den Wahrheitsgehalt wiedergibt. Positive Fakten haben den Wert *true*, negative Fakten den Wert *false*. Ebenso sind aber auch die Werte *both* und *unknown* möglich, um sowohl einen Widerspruch zu repräsentieren (*both*), als auch um auszudrücken, dass über das zugehörige Faktum bezüglich des Wahrheitsgehalts keine Aussage gemacht werden kann. Die allgemeine Syntax eines Faktums ist also $\mathtt{p}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$.

Prädikate – eigentlich deren Deklaration – bestimmen die *Sorten* der Wissensbasis.

```
person/1: <name>.
persons_temperature/2: <name>, <temperatur>.
```

deklariert die Sorte `name` als das erste (und in diesem Beispiel einzige) Argument der Prädikate `person` und `persons_temperature`. `temperatur` wird als zweites Argument des Prädikats `persons_temperature` bestimmt.

Dabei gibt die Zahl die Anzahl der Stellen des Prädikats an. Wird ein Prädikat nicht deklariert, generiert MOBAL Sortennamen.

Sorten drücken die hierarchische Struktur der Objekte innerhalb der Wissensbasis aus. Das *Sort Taxonomy Tool* (Seite 33 in diesem Abschnitt) berechnet diese Sorten und kann sie graphisch darstellen.

Regeln Als Regeln bezeichnet man Klauseln der Art:

$$L_1 \& \ldots \& L_n \rightarrow L_{n+1}$$

Die Klausel besteht mindestens aus zwei Literalen, hat einen *Kopf* und einen *Rumpf*, wobei im Kopf die Schlussfolgerung (*conclusion*) enthalten ist, und der Rumpf aus einer oder mehreren Prämissen besteht. Ein einfaches Beispiel ist:

```
small(X) --> not(large(X)).
```

Regeln enthalten Variablen (hier `X`) und werden auf alle passenden Fakten angewendet. Aus den Fakten:

```
small(maus).
small(laus).
```

mit dem Wahrheitswert *true* erzeugt obige Regel die beiden neuen Fakten:

```
not(large(maus)).
not(large(laus)).
```

Mit Hilfe von Regeln lassen sich komplexe Zusammenhänge beschreiben, etwa, dass eine Person ab einer bestimmten Temperatur Fieber hat.

```
persons_temperature(P, T) & gt(T, 37.5) --> has_fever(P).
```

drückt genau das aus. Dazu stellt MOBAL sogenannte *build-in*-Prädikate bereit – in diesem Fall `gt(_,_)` (*greater than*) für einen Größenvergleich.

Das *Rule Discovery Tool* (Seite 32) sucht in der Wissenbasis nach solchen Regeln. Dazu ist es jedoch notwendig, mit sogenannten *Metaprädikaten* ein Regelschema vorzugeben.

Metaprädikate sind – Salopp gesagt – Regeln für Regeln. Ein Metaprädikat beschreibt, welche Form eine gelernte Regel haben kann. Mit Hilfe dieses Schemas ist es möglich, die Hypothesen zu beschreiben, von denen erwartet wird, dass die das Zielprädikat formen. Anders ausgedrückt: Die Metaregeln bestimmen Form und Anzahl der im Körper einer Regel benutzten Prädikate. Ein sehr einfaches Regelmodell ist

```
mp(P, Q) : P(X) --> Q(X)
```

Das würde erlauben, von allen einstelligen Prädikaten auf andere einstellige Prädikate zu schließen, deren Instantiierung dieselbe ist.

RDT – Rule Discovery Tool

Diese Metaprädikate sind Voraussetzung für das Rule Discovery Tool – kurz: RDT. RDT [Kietz und Wrobel, 1992] ist ein sogenanntes *Wissenentdeckungs-Lernverfahren* das in MOBAL dazu dient, Regularitäten in den Fakten zu finden. An Hand der Fakten und einem Regelmodell versucht RDT Regeln zu finden, die je nach *Akzeptanzkriterium* positive oder negative Beobachtungen (Beispiele) aus der Faktenmenge abdecken. Die Beispiele müssen in den Fakten enthalten sein. RDT benutzt dabei das sogenannte *closed-loop learning*. Diese Lernmethode der Prädikatenlogik nutzt das Regelmodell, um alle mögliche Instantiierungen systematisch zu testen. Das Regelmodell ermöglicht eine Begrenzung des Hypothesenraumes.

Verschiedene Parameter beeinflussen das Akzeptanzverhalten von MOBAL. Dieses Akzeptanzkriterium teilt sich in *confirmation*- und *pruning*-Kriterium auf. Beide Kriterien sorgen dafür, dass die Hypothesensuche an einer bestimmten Stelle abgebrochen wird:

1. Wenn das *pruning*-Kriterium erfüllt ist und damit die Hypothesen zu speziell würden, oder

2. wenn das *confirmation*-Kriterium erfüllt ist und damit die Hypothesen zu allgemein würden.

Insgesamt können fünf Systemwerte benutzt werden, um die Kriterien zu bestimmen. Diese Werte und alle einstellbaren Parameter befinden sich in Anhang C.1, Seite 91.

Die Größe des Hypothesenraums von RDT hängt von folgenden Werten ab: Von der Anzahl der Regelschemata r, der Anzahl der Prädikate p und die maximale Anzahl k der Literale in einem Regelschema. Sollen Konstanten gelernt werden kommt die Anzahl c der Argumente und die maximale Zahl i aller möglichen Belegungen für c hinzu. Diese Werte bestimmen die obere Schranke des Hypothesenraumes [Brockhausen und Morik, 1997]:

$$r \cdot (p \cdot i^c)^k \tag{2.1}$$

Da hier keine Konstanten gelernt werden ist $c = 0$, dadurch wird $i^c = 1$ und die obere Schranke wird nur von r, p und k bestimmt:

$$r \cdot p^k \tag{2.2}$$

STT – Sort Taxonomy Tool

Dieses Tool teilt die Domäne in Sorten bzw. Klassen ein. Als Eingabe erwartet es einfach die Fakten. Das Ergebnis ist eine Klassifizierung in Sorten in Form eines Netz- bzw. Gitterwerkes, die sogenannte *sort taxonomy*. Das Wort Taxanomie bedeutet die Einordnung von Konzepten in ein System – das Schöne daran ist, dass man diese Einteilung graphisch darstellen kann und so einen Überblick der Faktenbasis bekommt. Der Sortenname wird entweder vom Benutzer gewählt oder von STT generiert. Dieses Tools kann vom Anwender dazu benutzt werden, seine Faktenbasis zu inspizieren. Das System kann mit Hilfe dieser Taxonomie neue Fakten und Regeln auf Kompatibilität überprüfen.

Die Parameter des Sort Taxonomy Tools sind in Anhang C.2, Seite 92 abgebildet.

Der Algorithmus zur Konstruktion der Taxonomie beruht im Wesentlichen darauf, eine Menge konstanter Ausdrücke zu bilden, die mit Hilfe ihres Auftretens an bestimmten Argumentstellen in den Prädikaten gefunden werden.

Eine detailierte Beschreibung der Theorie, auf der MOBAL basiert, findet sich in [Morik et al., 1993].

2.3 Der Information Layer

Das Forschungsprojekt COMRIS (*cohabited mixed reality information spaces*) untersucht die Möglichkeiten zwei Informationsräume – den realen mit dem virtuellen – zu verbinden. Der Ansatz beruht auf einem Software-Agenten-Konzept, dass auf eine enge Zusammenarbeit von realen und virtuellen Agenten setzt. Ein wesentlicher Bestandteil dieses Systems ist der Information Layer [Haustein, 1999]. Er ist der Dreh- und Angelpunkt für den Informationsaustausch zwischen den beiden Informationsräumen.

Seine Besonderheiten sind vor allem die verschiedenen Services (siehe Abschnitt 2.3.2) und die Verwaltung der Daten innerhalb einer anpassbaren, objekt-orientierten Ontologie[10]. Diese Ontologie setzt ihre Objekte in Beziehung zueinander und macht den Aufbau des Information Layer offen und modular.

Die Kommunikation erfolgt nach dem *XML*[11]-Standard [Bray et al., 1998]. Das hat verschiedene Vorteile: XML-Parser sind für verschiedene Progammiersprachen verfügbar und es gibt bereits verschiedene XML-Tools – zukünftige Webbrowser werden diesen Standard unterstützen. XML erlaubt es, die Daten und die impliziten Informationen über diese Daten zu erwei-

[10]Der Begriff Ontologie kommt eigentlich aus der Philosohie und bedeutet: Lehre von den Ordnungs-, Begriffs- und Wesensbestimmungen des Seienden und geht bis auf Aristoteles zurück. »on« heißt Seiendes, »logos« Lehre [Ferber, 1998].

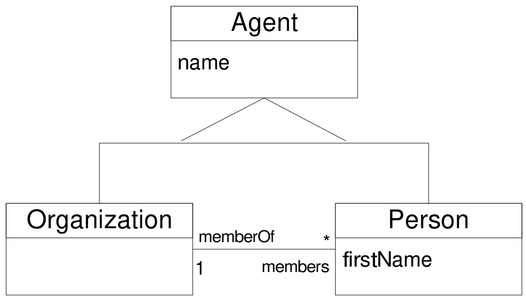
[11]*Extensible Markup Language*

tern oder zu verändern.

In dieser objekt-orientierten Umgebung werden nicht nur reine Fakten gespeichert, sondern auch Beziehungen, Relationen zwischen diesen Fakten. Objekte werden als Klassen definiert und teilen sich Eigenschaften mit anderen Klassen oder vererben sie an Unter-Klassen.

2.3.1 Definition der Klassen

Zur Definition der Ontologie werden Klassen definiert, die miteinander in Beziehung stehen. Abbildung 2.4 verdeutlicht diesen objekt-orientierten Ansatz.




```
<entity id="Agent">
    <defAttr id="name"/>
<entity/>

<entity id="Person" isa="Agent">
    <defAttr id="firstName"/>

<entity id="Organization" isa="Agent">
    <defRelation id="members"
        type="Person"
        inverse="memberOf"
        imax="1"/>
<entity/>
```

Abbildung 2.4: Modellierung von drei Klassen innerhalb des Information Layer.

Im linken Teil ist ein Klassendiagramm in der *Unified Modelling Language* (*UML*) [Oestereich, 1997] modelliert. Daneben befindet sich die entsprechende Strukturdefinition des Information Layer im XML-Format. Insgesamt werden drei Klassen abgebildet: `Agent`, `Person` und `Organization`. Der Pfeil symbolisiert, dass sowohl `Organization`, als auch `Person` Unterklassen von `Agent` sind. Die Eigenschaft `name` der Klasse `Agent` vererbt sich an alle Unterklassen – sowohl eine Organisation hat einen Namen, als auch eine Person. Außer der Eigenschaft `name` hat die Klasse `Person` auch noch das Attribut `firstName`, was der menschlichen Konvention entspricht, einer Person Vor- und Nachnamen zu geben.

Zwischen Personen und Organisationen gibt es Beziehungen, die in der Abbildung durch einfache Linien (sogenannte *Assoziationen*) symbolisiert sind. So hat eine Organisation mehrere Mitglieder und umgekehrt ist eine Person Mitglied einer Organisation. Das geht aus den Beschriftungen der Assoziationen hervor. `memberOf` ist mit einer »1« versehen, `members` mit einem Stern (»*«). Die Assoziation `memberOf` begrenzt die Mitgliedschaft einer Person auf genau eine Organisation (1:1-Relation), während der Stern bedeutet, das eine Organisation prinzipiell unbegrenzt viele Personen als Mitglieder zählen kann (1:n-Relation).

Beide Relationen werden innerhalb der Klasse `Organization` definiert. Zu dieser Definition gehört die Angabe des assoziierten Objekts (hier die Klasse `person`) und die Deklaration der inversen Relation (hier `memberOf`). Diese inverse Relation wird durch den Wert des Attributs `imax` auf ein Objekt der Klasse `Organization` begrenzt. Diese Strategie ermöglich es, die Ontologie an neue Situationen anzupassen.

Diese Klassendefinition wird in einer Datei gespeichert und ist so von anderen Programmen bzw. Agenten abrufbar.

2.3.2 Module

Wie oben erwähnt bietet der Information Layer mehrere Dienste, die dafür sorgen, Informationen zu verteilen. Tabelle 2.2 zeigt die verschiedenen Module, die die Funktionalität des Information Layer realisieren.

MODUL	BESCHREIBUNG
Server	Der Server gewährleistet, dass andere Module auf Datenquellen zugreifen können. Datenquellen können XML-Dateien, SQL-Datenbanken oder andere Server sein.
Instance Browser	Das ist ein graphisches Interface, um die Inhalte des IL zu inspizieren.
ACL Server	Der ACL Server verbindet den IL mit der COMRIS Infrastruktur.
HTTP Server	Dieses Modul ermöglicht die Darstellung der Inhalte des IL auf einer Webseite.

Tabelle 2.2: Die Module des Information Layer.

Für den Testbetrieb ist der Instance Browser die ideale Schnittstelle zum Information Layer.

2.3.3 Datenaustausch mit dem Information Layer

Der Server sorgt für den Austausch von Objekten zwischen den Informationsagenten und dem Information Layer. Im Dialog mit dem IL-Server treten die Agenten als Clients auf, die einzelne Klassen oder Auflistungen bestimmter Instanzen dieser Klassen anfordern können.

Für die Darstellung in HTML benutzt der Information Layer sogenannte *templates*[12]. Die Ausdrücke, mit denen diese Templates gestaltet werden, können dazu benutzt werden, die mit MOBAL gelernten Regeln in das System zu integrieren.

[12] Schablonen

Kapitel 3

Systementwurf

Wie schon in der Einleitung (Seite 13) dargestellt, ist das Ziel dieser Diplomarbeit ist die Suche und anschließende Ähnlichkeitsbewertung von Informationen. Aber welche Informationen sind verfügbar und können bewertet werden? – Welche persönlichen Aspekte können überhaupt auf Ähnlichkeit verglichen werden?

Zwei Menschen können sich in verschiedenen Aspekten ähneln. Z. B. teilen zwei Menschen die Vorliebe für japanisches Essen oder gehen gerne Angeln. Aber woher kann ein System diese Art von Informationen bekommen?

Ein Ansatz war, die privaten Webseiten von Konferenzteilnehmern nach diesen Informationen zu durchsuchen. Jedoch zeigten zahlreiche Stichproben, dass die Gestaltung von Webseiten und das Layout von Firmensites sehr individuell ausfallen. Die wenigsten Webseiten verraten Hobbys oder persönliche Interessen. Außerdem ist die Struktur der von Websites so unterschiedlich, dass es eine eigene Untersuchung wert ist, hier nach Informationen zu suchen. [Freitag, 1998] untersuchte die Schwierigkeiten einer solchen Extraktion aus HTML-Seiten. Sie ging dabei von Webseiten aus, denen eine gewisse Regularität im Aufbau gemein ist, da sie von einer Institution stammen. Selbst bei diesen Voraussetzungen hatte sie Schwierigkeiten bei der Namensextraktion von Personen, da die HTML-Strukturen bei persönlichen Webseiten, im Vergleich zu Webseiten die Kurse ankündigten, recht unterschiedlich waren.

Ähnlich sind sich zwei aber auch Personen, wenn sie in der gleichen Fachrichtung forschen. An Hand der Veröffentlichungen oder Projektbeteiligungen ist z. B. sehr gut ablesbar, welcher Wissenschaftler wann und mit wem veröffentlicht hat. Was kann daraus geschlossen werden, dass Teilnehmer bereits zusammen veröffentlicht haben? Man könnte davon ausgehen, dass sie sich kennen. Wenn sie schon über Jahre hinweg zusammen veröffentlicht haben, kennen sie sich dann gut (zumindest lange)? Die gleichen Fragen stellen sich für Projektpartner.

Da die Extraktion relevanter Informationen aus persöhnlichen Webseiten auf Grund der fehlenden Struktur nicht möglich ist, beziehen wird die Informationen aus strukturierten Quellen. Für Publikationen und Projekte sind solche Quellen im WWW verfügbar, so können die oben beschriebenen Probleme umgangen werden.

In diesem Kapitel wird die Modellierung des Gesamtsystems vorgestellt und die nötigen Definitionen der Ontologie des Information Layer erläutert. Im Wesentlichen sind zwei Aufgaben zu lösen:

1. Der Entwurf von Informationsagenten für die Suche und Extraktion personenbezogener Daten und

2. eine Ähnlichkeitsbewertung dieser Daten mit einem maschinellen Lernverfahren.

Die erste Aufgabe bildet eine typische Anwendung für Informationsagenten. Bei der Suche handelt es sich um einen immer wiederkehrenden Prozess, der unabhängig von einem Anwender automatisch gestartet wird. Die Agenten sind dabei jeweils für eine ganz bestimmte Datenquelle innerhalb des World Wide Web zuständig. Sie speichern die gefundenen und extrahierten Daten für die Weiterverwendung im Information Layer. *Autonomie* (siehe Abschnitt 2.1, Seite 15) ist also eine Eigenschaft, die den *aufgabenspezifischen* Agenten eigen sein soll.

Für die zweite Aufgabe müssen die relevanten Daten aus dem Information Layer in einer geeigneten Repräsentation nach MOBAL exportiert werden. Mit dem integrierten Lernverfahren RDT wird dann versucht, Regularitäten in diesen Daten zu finden. Die gelernten Regeln werden über Templates mit den Daten des Information Layer kombiniert und stehen dann anderen Anwendungen zur Verfügung. Diese Anwendungen können wiederum von Agenten bedient werden. Auch hier gilt das Gleiche wie für die Informationsagenten – das Agentensystem ist beliebig erweiterbar.

Das System könnte dahingehend erweitert werden, dass die Benutzer die Möglichkeit haben, falsche oder schlechte Vorhersagen zu kennzeichnen, damit die Regeln verfeinert werden können. In der jetzigen Version werden die Regeln jedoch nur einmal generiert und mit Hilfe der Templates in die HTML-Ausgabe integriert.

Abbildung 3.1 verdeutlicht die Architektur des Systems.

Die gestrichelten leeren Kästchen in Abbildung 3.1 sollen die Skalierbarkeit des Systems verdeutlichen. Es kann durch weitere Agenten beliebig erweitert werden.

Der nächste Abschnitt (3.1, Seite 39) enthält die Planung dieser Agenten. In Abbildung 3.1 bilden sie die Schnittstelle zwischen dem WWW und dem Information Layer. Abschnitt 3.2 (Seite 43) enthält die Modellierung der nötigen Objekte innerhalb der Ontologie des Information Layer.

Schließlich beschreibt Abschnitt 3.3 (Seite 47) die angestrebte Ähnlichkeitsbewertung mit Hilfe eines Regellernverfahrens. Die gelernten Regeln werden

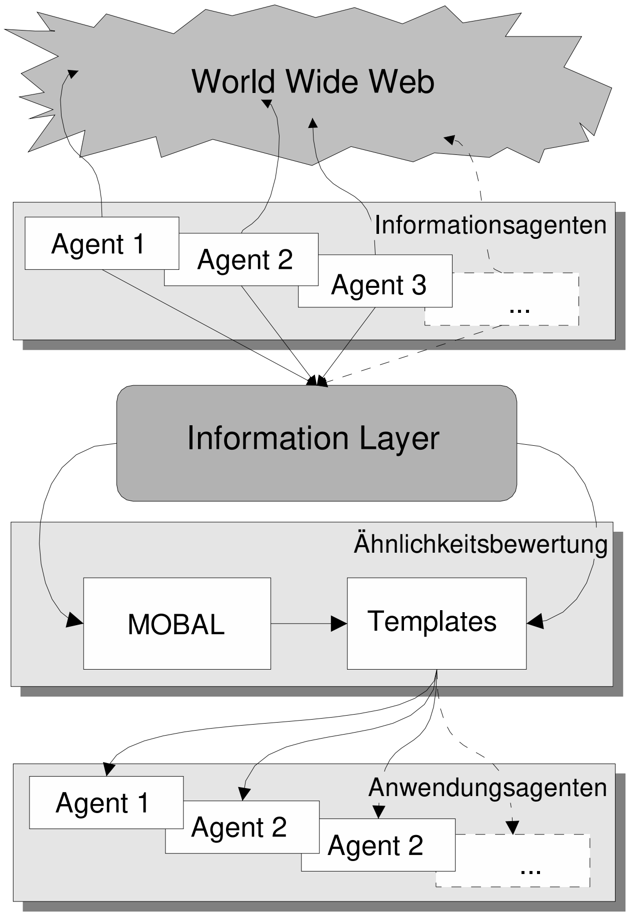


Abbildung 3.1: Das Agentensystem.

dann so in das System integriert, dass Konferenzteilnehmern Empfehlungen bezüglich Treffen gegeben werden.

3.1 Das Agentensystem

In diesem Agentensystem kooperieren verschiedene aufgabenspezifische Informationsagenten miteinander, um den Information Layer mit Informationen bezüglich gemeinsam veröffentlichter Dokumente und EG-Projektbeteiligungen zu füllen.

Grundsätzlich soll das System beliebig erweiterbar sein. D. h. man sollte Agenten hinzufügen oder entfernen können, ohne dabei das ganze System ändern zu müssen. Selbst ein einzelner Agent sollte angepasst werden können ohne Einfluss auf das System. Das Schema wird in Abbildung 3.2 verdeutlicht.

Die Informationsagenten kümmern sich um die Extraktion von Daten aus ganz bestimmten Quellen des World Wide Web. Diese Datenquellen werden

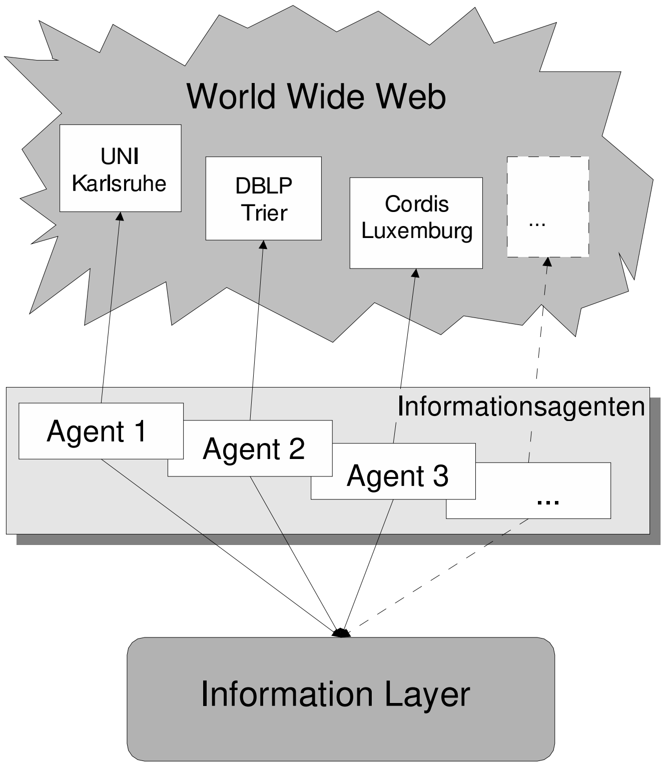


Abbildung 3.2: Die Informationsagenten.

im Anhang B genauer beschrieben. Die extrahierten Daten werden in den Information Layer eingefügt.

Die Agenten sind jeweils als Threads realisiert und arbeiten parallel. Sollte der Server einer Datenquelle nicht verfügbar sein, liefert der entsprechende Agent eine Fehlermeldung, die anderen arbeiten jedoch normal weiter. An den gefundenen, bzw. nicht gefundenen Informationen macht es sich bemerkbar, wie erfolgreich ein Agent war.

3.1.1 Modellierung der Agenten

Für jede Datenquelle gibt es einen speziellen Agenten, der die Besonderheiten dieser Quelle kennt. Neben diesen individuelle Anpassungen, sind einige der zu bewältigenden Aufgaben bei allen Agenten gleich ist. Die Datenquellen sind:

- Die Projektdatenbank von CORDIS in Luxemburg,
- die Informatik-Bibliographie *ACM Sigmoid DBLP* an der Universität Trier und
- die *Collection of Computer Science Bibliographies* an der Universität Karlsruhe.

Die URLs dieser Datenquellen und weitere Einzelheiten finden sich im Anhang B (Seite 81). Jede Datenquelle wird von genau einem Agenten abgefragt. Jeder Agent wird in einer eigenen Klasse modelliert:

- `DBLPCollector`
- `UKACollector`
- `CordisCollector`

Wie in Abbildung 3.3 dargestellt, sind diese Klassen Unterklassen der Klasse `Collector`.

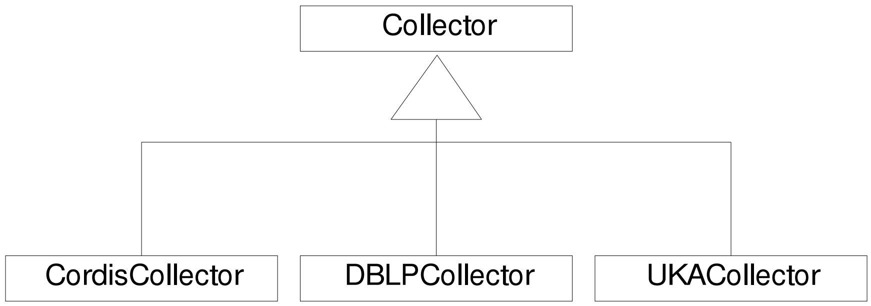


Abbildung 3.3: Vererbungshierarchie des Collectors.

Die objekt-orientierte Programmiersprache JAVA bietet die richtigen Voraussetzungen für die Realisierung dieser Agenten, da sie die *Vererbung* bei Objekten ermöglicht. Bei der Vererbung werden Eigenschaften einer *Superklasse* (hier der `Collector`) an die *Subklassen* (Unterklassen) weitergegeben.

Gesteuert werden diese Agenten vom `CollectorSupervisor`, der den Information Layer beobachtet und Neuzugänge unter den Personen zum Anlass nimmt, die Agenten auf die Suche zu schicken.

3.1.2 Der Basisagent

Der `Collector` ist der sogenannte Basisagent dieses Systems. Er stellt grundlegende Eigenschaften bereit, die von allen Agenten gebraucht werden. Alle Agenten unterhalten einerseits eine Verbindung zum Information Layer und unterstützen andererseits das HTT-Protokoll zur Extraktion der Informationen. Personen, Publikationen und Projekte haben innerhalb der Ontologie des IL eine ganz bestimmte Repräsentation (siehe Abschnitt 3.2). Da alle Agententypen auf diese Repräsentation zugreifen, ist die Funktion ebenfalls im Basisagenten – dem `Collector` – enthalten. Diese Eigenschaften vererben sich so automatisch an die Subklassen. Im Einzelnen sind das:

- Verbindungsaufbau zum Information Layer: Die Agenten bauen eine Verbindung zu dem Port auf, an dem der IL-Service läuft.

- Das Entfernen von HTML-Tags: Durch diese Tags wird ein HTML-Dokument zwar strukturiert, aber diese Tags enthalten keine relevanten Informationen. Da die Seiten der Informationsquellen immer den gleichen Aufbau haben, finden sich die relevanten Informationen immer an den gleichen Stellen im Dokument. So wird zwar die Informationssuche erleichtert, aber nach der Extraktion müssen diese Tags entfernt werden, da sie nicht zur eigentlichen Information zählen.

- Das Ersetzen von Umlauten: Die Datenquellen verfahren mit den Anfragen unterschiedlich. Umlaute müssen unter Umständen durch passende Zeichen ersetzt werden.

- Das Synchronisieren von Personen mit dem Information Layer: Die Agenten finden entweder Autoren, Editoren oder Projektpartner. Diese Personen sind entweder schon im IL vorhanden, oder müssen eingefügt werden.

- Das Synchronisieren von Veröffentlichungen: Da es zwei Agenten gibt, die publikationsbezogene Informationen extrahieren, wird auch diese Funktion im Basisagenten realisiert[1].

3.1.3 Die Publikations-Agenten

Eine typische wissenschaftliche Arbeit ist ein Dokument mit einem Titel, einer Liste von Autoren, einer Zusammenfassung, dem eigentlichen Text und einer Liste von Literaturangaben. Den Titel und die Autorenliste findet man auch in den Bibliotheken wieder. Manchmal werden auch die Zusammenfassung oder die Stichwörter gespeichert – das jedoch eher unregelmäßig. Die Agenten können sich also nur auf den Titel, die Autoren, und das Veröffentlichungsjahr verlassen.

Als Publikations-Agenten kommen zwei Agenten zum Einsatz. Sie erweitern die Funktionalität des Basisagenten um spezielle, an die Datenquellen angepasste Funktionen.

Die Agenten übermitteln den Namen einer Person als Anfrage an die Datenquelle und erhalten HTML-Seiten als Ergebnis. Die gefundenen Publikationen werden zunächst in einem eigenen Objekt verwaltet und schließlich mit dem Information Layer synchronisiert.

[1] Das entsprechende Synchronisieren der Projekte wird noch im speziellen Projekt-Agenten erledigt.

3.1.4 Der Projekt-Agent

Ein Agent beschafft Informationen bezüglich Projektbeteiligungen von Personen. Die Datenquelle ist die Projekt-Datenbank von CORDIS in Luxemburg (Details siehe Anhang B, Seite 81). Die Informationen werden hier jedoch nicht auf der ersten Ebene gefunden, wie bei den beiden Publikations-Agenten. Die Anfrage bei der Datenquelle liefert eine Liste von Hyperlinks auf Projekte zurück, an denen die Person beteiligt war (oder ist). Der Agent muss diesen Links folgen, um detailierte Angaben zu dem entsprechenden Projekt und den Projektpartnern zu finden.

Projekte bilden innerhalb des Information Layer eine eigene Klasse (siehe Abschnitt 3.2). Jedoch werden auch diese Daten zunächst in einer eigenen Klasse verwaltet, bevor sie mit dem IL synchronisiert werden.

3.1.5 Der Supervisor

Der Supervisor macht dieses System autonom. Er bestimmt, wann die Agenten aktiv werden sollen. Gegenwärtig tut er das nur, wenn er neue Konferenzteilnehmer im Information Layer entdeckt. Es soll aber auch noch ein Aktualisierungsmodus integriert werden, der dann zu bestimmten Zeiten die Agenten auf die Suche schickt.

3.1.6 Informationsverarbeitung

Während der Suche speichern die Agenten jede gefundene Publikation bzw. jedes gefundene Projekt in einem eigenen Objekt. Insbesondere die Ergebisse der Anfrage in Karlsruhe müssen darauf untersucht werden, ob es sich wirklich um eine relevante Publikation handelt. Es kann nämlich vorkommen, dass der Name einer Person weder als Autor noch als Editor einer Publikation erscheint, sondern z. B. im Titel vorkommt. Dann ist die gefundene Veröffentlichung keine Publikation dieser Person – sie handelt vielleicht *von* der Person. Erst wenn feststeht, dass eine Person als Autor oder Editor auftritt, werden die Publikationen mit dem Information Layer synchronisiert.

3.2 Definition der Klassen des Information Layer

Da die gefundenen Daten mit dem Information Layer synchronisiert werden sollen, muss die Ontologie des IL für diese Daten vorbereitet sein. Das geschieht einfach durch Definition von neuen Objekten. Hier werden die Objekte für

- Personen,
- Publikationen und

- Projekte

definiert.

Tabelle 3.1 und Abbildung 3.4 zeigen, wie diese Objekte als Klassen im Information Layer modelliert und damit definiert sind. Diese Klassendefinitionen sind in einer Datei (`entities.xml`) abgelegt und können so von Agenten oder anderen Applikationen abgerufen werden.

```
<entity id="Person" show="lastName + ', ' + firstName">
  <defAttr id="firstName"/>
  <defAttr id="lastName"/>
  <defAttr id="participant"/>
  <defAttr id="description" format="Memo"/>
</entity>

<entity id="Publication" show="title">
  <defAttr id="title"/>
  <defAttr id="year"/>
  <defAttr id="class"/>
  <defAttr id="journal"/>
  <defAttr id="booktitle"/>
  <defRelation id="author" type="Person" inverse="publications"/>
  <defRelation id="editor" type="Person" inverse="editorOf"/>
</entity>

<entity id="Project" show="title">
  <defAttr id="title"/>
  <defAttr id="projectreference"/>
  <defAttr id="status"/>
  <defAttr id="starts"/>
  <defAttr id="ends"/>
  <defRelation id="projectleader" type="Person" inverse="leaderof"/>
  <defRelation id="participant" type="Person" inverse="participation"/>
</entity>
```

Tabelle 3.1: Die Definition der Klassen `Person`, `Publication` und `Project` des Information Layer.

Jede Person hat Attribute (ID-Nummer, Vorname, Name, eine Kennzeichnung, ob diese Person Konferenzteilnehmer[2] ist und eine optionale Beschreibung) und Relationen zu anderen Klassen. Wie in Tabelle 3.1 zu sehen, werden die Relationen, die eine Person haben kann, in der Klasse bestimmt, zu der es eine Beziehung geben kann. D. h. die Definition der Klasse `Person` determiniert nicht die Verknüpfungen, die eine Instanz der Klasse haben kann. So ist die Ontologie einfach erweiterbar. Da jede Instanz eine eindeutige Kennzeichnung hat (`id`), sind auch die Relationen eindeutig.

[2] Achtung: Das Attribut `participant` darf nicht mit der Relation `participant` der Klasse `Project` verwechselt werden.

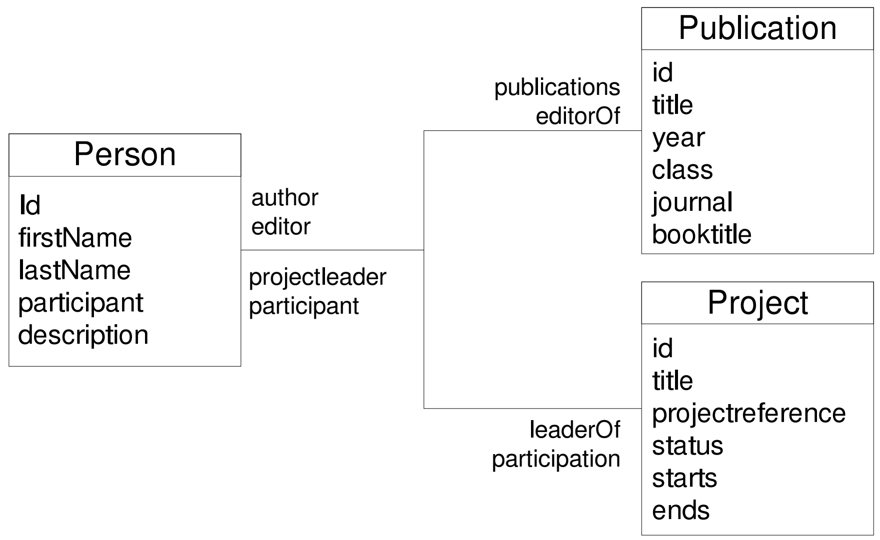


Abbildung 3.4: Klassendiagramm der Objekte `Person`, `Publication` und `Project`.

Publikationen werden durch Titel, Erscheinungsjahr, Dokumentenklasse[3] (siehe Tabelle 4.6, Seite 54), in welchem Journal oder welchem Buch sie erschienen sind, repräsentiert. Desweiteren gibt es Relationen zu Autoren und Editoren und umgekehrt. In Abbildung 3.4 sind die Kanten entsprechend beschriftet. Editoren und Autoren sind also Personen, deren Beziehungen zu einer oder mehreren Publikationen sie erst zu Autoren oder Editoren machen.

Projekte werden durch Titel, Projektreferenznummer, Status, Anfangs- und Enddatum repräsentiert. Auch hier sind Relationen zu Personen definiert, nämlich zu den Projektteilnehmern und dem Projektleiter. Ist eine Person Projektleiter erhält die entprechende Instanz die Relation `person.leaderOf`, die auf die entsprechende Projekt-Instanz verweist.

Tabelle 3.2 zeigt einen Ausschnitt der Datei `instances.xml`, die alle Instanzen enthält, die der Information Layer kennt.

Die Instanz der Klasse `Person` enthält Verweise auf Instanzen der Klassen `Publication` und `Project`. Die betreffende Person ist Autor und Editor mehrerer Publikationen (`<publications .../>`, `<editorOf .../>`) und Partner bei mehreren Projekten (`<participation .../>`)[4]. Die Identitätsnummern werden automatisch erzeugt, wenn eine neue Instanz einer Klasse angelegt wird.

[3]Wir werden später sehen, dass die Dokumentenklasse (Veröffentlichungsart) nur von einer Datenquelle bereitgestellt wird.

[4]Die Liste der dargestellten Instanzen ist gekürzt.

```
<Person id="d92b0ea421">
  <firstName>Katharina</firstName>
  <lastName>Morik</lastName>
  <participant>true</participant>
      ...
  <publications idref="d92b0f777e" hint="The {Real-Estate} Agent --- Mo
  deling Users by Uncertain Reasoning" type="Publication"/>
  <publications idref="d92b0faf53" hint="Sloppy modelling" type="Public
  ation"/>
      ...
  <editorOf idref="d92b0faf53" hint="Sloppy modelling" type="Publicatio
  n"/>
      ...
  <participation idref="25500" hint="Co-Habited Mixed-Reality Informati
  on Spaces" type="Project"/>
</Person>

<Project id="25500">
  <title>Co-Habited Mixed-Reality Information Spaces</title>
  <projectreference>25500</projectreference>
  <status>Execution</status>
  <starts>1997-10-01</starts>
  <ends>2000-09-30</ends>
  <projectleader idref="d92b0f64d3" hint="Steels, Luc" type="Person"/>
      ...
  <participant idref="d92b0ea421" hint="Morik, Katharina" type="Person"
  />
      ...
</Project>

<Publication id="d92b0faf53">
  <title>Sloppy modelling</title>
  <year>1989</year>
  <class>inproceedings</class>
  <journal>-</journal>
  <booktitle>Knowledge Representation and Organization in Machine Learn
  ing</booktitle>
  <author idref="d92b0ea421" hint="Morik, Katharina" type="Person"/>
  <editor idref="d92b0ea421" hint="Morik, Katharina" type="Person"/>
</Publication>
```


Tabelle 3.2: Ausschnitt aus der Datei `instances.xml`. Die Instanzen Person und Project sind gekürzt.

Es ist klar, dass eine Publikation in jedem Fall einen Autor haben muss, genauso wie ein Projekt in jedem Fall einen Leiter hat[5]. Die Anzahl der Autoren, Editoren und Projektpartner kann variieren.

Erst wenn eine Instanz der Klasse Person eine Relation zu einer Instanz einer anderen Klasse hat, wird die entsprechende Instanz um diese Relation erweitert. Da jede Instanz eine eindeutige Kennzeichnung hat (`id`), sind auch die Relationen eindeutig.

3.3 Ähnlichkeitsbewertung mit MOBAL

Für die Ähnlichkeitsbewertung müssen die Objekte `Person`, `Publication` und `Project` in das System MOBAL importiert werden. Dazu müssen sie von der objekt-orientierten Repräsentation im Information Layer in eine prädikatenlogische Repräsentation gebracht werden, mit der MOBAL lernen kann.

3.3.1 Die Lernaufgabe

So wird die Änlichkeitsbewertung zu einer Regellernaufgabe gemacht. Die Ähnlichkeit wird durch die Art der Beziehung zweier Personen zum Ausdruck gebracht. Die vier Beziehungsklassen lassen sich wie folgt charakterisieren:

1. bekannt,
2. gut bekannt,
3. freundschaftlich und schließlich das eigentliche Ziel:
4. man möchte sich treffen.

Den Konferenzteilnehmern werden jedoch nur die Personen präsentiert, von denen das System meint, dass sie sich treffen wollen.

Die anderen Beziehungen sollen der Aussagekraft des Zielprädikats dienen. Die Prädikate, die für die Modellierung gebildet werden heisßen entsprechend:

1. `known`,
2. `well_known`,
3. `friends` und
4. `wants_to_meet`.

[5] Es gibt tatsächlich Projekte, die nur einen Leiter haben und keine weiteren Vertragspartner.

3.3.2 Generierung der Daten

Um die Wissensbasis möglichst flexibel zu halten, werden die Fakten und Prädikate aus dem Inhalt des Information Layer generiert. Da die Klassendefinitionen des Information Layer abgerufen werden können, sollte es möglich sein, auch hier Agenten einzusetzen. In dieser Arbeit werden nur die Klassen `Person`, `Publication` und `Project` des IL nach MOBAL exportiert.

Das Progamm `MobalOut` erzeugt eine Textdatei, die von MOBAL eingelesen werden kann. Dazu werden die relevanten Elemente des Information Layer in eine Repräsentation gebracht, mit der MOBAL lernen kann. Hier könnte sicherlich auch der elegantere Weg über das MOBAL eigene *Programmer's Interface* gegangen werden.

3.3.3 Integration der gelernten Regeln

Die Integration der mit MOBAL gelernten Regeln geschieht manuell. Die Regeln werden *offline* gelernt und anschließend mit Templates (siehe Abschnitt 2.3.3, Seite 35) in den Information Layer eingebunden.

Kapitel 4

Implementierung

Dieses Kapitel beschreibt die konkrete Vorgehensweise in Bezug auf den im letzten Kapitel dargestellten Entwurf.

4.1 Agenten

Zur Implementierung des Agentsystems wurde die Programmiersprache JAVA benutzt ([Flanagan, 1998],[Morik, 1998]). Obwohl die Performanz mit einer anderen Sprache vielleicht etwas gesteigert werden könnte, bietet JAVA unübersehbare Vorteile, die insbesondere in diesem Einsatzgebiet überzeugend sind:

- JAVA ist objekt-orientiert,
- der Zugriff auf Internet-Ressourcen wird durch vorgefertigte Klassen vereinfacht.
- Der generierte Code ist plattformunabhängig, damit kann das Agentensystem auf nahezu allen Rechnern im Netz betrieben werden.
- Die Sprache ist frei verfügbar.

Insgesamt werden drei Agenten implementiert, die jeweils für eine Datenquelle zuständig sind.

1. der CORDIS-Agent (nächster Abschnitt),
2. der DBLP-Agent (Abschnitt 4.1.2) und
3. der UKA-Agent (Abschnitt 4.1.3).

Die Namen ergeben sich aus den abgefragten Datenquellen. UKA ist eine Abkürzun für die Universität Karlsruhe.

4.1.1 CORDIS-Agent

Der CORDIS-Agent sammelt Informationen über Projektbeteiligungen. Dazu übermittelt er seine Anfrage an den CORDIS-Server in Luxemburg.

Die Anfragen haben die Form:

```
http://apollo.cordis.lu/cordis-gci/srchidadb?CALLER=
         PROJADVANCESRCH&QZ_WEBSRCH=<Vorname>%20<Nachname>
```

Allerdings findet der Agent die Informationen nicht auf der ersten Ebene. Die zurückgelieferte Trefferseite (HTML-Format) enthält im Falle einer erfolgreichen Anfrage nur Links auf die gefundenen Projekte (siehe Abbildung B.7, Seite 88). Diesen Links folgt der Agent und kommt so zu den detailierten Projektdaten (Abbildungen B.8–B.10, Seiten 89–89). Die HTML-Struktur ist hier keine Hilfe beim Finden der Informationen. Es hat sich gezeigt, dass es in diesem Falle besser ist, zuerst die HTML-Tags zu entfernen, und dann einfach den Text zu durchsuchen. Tabellen 4.1 und 4.2 zeigen den Unterschied.

```
<P>
<TABLE><TR><TD>
<STRONG>Record Control Number :  </STRONG>38365
</TD><TD>
<STRONG>Quality Validation Date :  </STRONG>1997-11-05
</TD></TR></TABLE><P>
<STRONG>Update Date :  </STRONG>1998-05-04<P>
<STRONG>Project Acronym :  </STRONG><A HREF="/cordis-cgi/srchidadb?CALL
ER=ACROLINK&QM_EA_ACR_A=COMRIS&QM_EA_CAT=PROJECT&QM_EA_ORI=COMMISION&LI
NK_CALLER=PROJADVANCEDSRCH&LINK_SESSION=136891998-6-10">COMRIS</A><P>
<STRONG>Title :  </STRONG>CO-HABITED MIXED REALITY INFORMATION SPACES<P
>
```

Tabelle 4.1: Auschnitt aus einer HTML-Seite, die Projekdaten enthält.

```
Record Control Number :  38365
Quality Validation Date :  1997-11-05
Update Date :  1998-05-04
Project Acronym :  COMRIS
Title :  CO-HABITED MIXED REALITY INFORMATION SPACES
```

Tabelle 4.2: Derselbe Auschnitt ohne HTML-Tags.

Wie man hier sehr schön sehen kann, wird durch die Entfernung der Tags nicht nur der Text lesbarer, die Einträge sind auch für den Agenten einfach zu extrahieren.

Da es bis jetzt nur einen Agenten gibt der Projektdaten sammelt, und diese Daten aus einer Quelle kommen, gibt es keine unterschiedlichen Daten über Projekte. Das Synchronisieren wird hier auf folgendes Verfahren reduziert:

- Gibt es das Projekt bereits im Information Layer?

- JA, dann arbeite mit dem existierenden Projekt weiter.
- NEIN, dann erzeuge ein neues Projekt-Objekt im IL.

4.1.2 DBLP-Agent

Der DBLP-Agent extrahiert Informationen über Veröffentlichungen aus der Informatik-Bibliographie in Trier.

Der DBLP-Agent bekommt eine HTML-Seite als Ergebnis, die alle Dokumente einer Person innerhalb einer Tabelle angibt. Die Anfrage lautet einfach:

```
http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/
       <initial des nachnamens>/<Nachname>:<Vorname>.html
```

Entweder hat der Server eine Datei mit diesem Namen, oder nicht. Da sich der Dateiname dieser HTML-Seite aus dem Vor- und Nachnamen der Person zusammensetzt, muss das Ergebnis nicht weiter verifiziert werden. Die Autorenschaft aller in diesem HTML-Dokument aufgeführten Schriften liegt eindeutig bei der betreffenden Person.

Die strukurellen Gegebenheiten der `<table>`-Umgebung können für die Extraktion der relevanten Informationen ausgenutzt werden. Erscheinungsjahre stehen in einer Reihe, die über alle Spalten geht. In den Spalten einer Reihe finden sich die Publikationsdetails. Zuerst kommt eine Liste der Autoren (diese kann natürlich auch nur aus einem Autor bestehen) – danach der Titel (siehe Abbildung 4.3).

Da in der DBLP-Bibliographie keine Angaben über die Dokumentenklasse gemacht werden, liefert der DBLP-Agent nur Titel, Jahr und Autoren – grundsätzlich weniger Informationen über Veröffentlichungen als der UKA-Agent. Andererseits enthält die DBLP-Bibliographie Dokumente, die bei der UKA-Bibliographie nicht zu finden sind, so dass sich beide Quellen gut ergänzen. Die Qualität der Suchergebnisse wird in Abschnitt 5.1 (Seite 61) beschrieben.

Der DBLP-Agent kennzeichet den Eintrag der Dokumentenklasse derart, das der UKA-Agent erkennt, dass ein vorhandener Eintrag vom DBLP-Agenten stammt. In diesem Fall gilt die Sonderregelung, dass nicht auf Gleichheit der Dokumentenklasse geprüft wird, sondern es wird hier angenommen, dass ein Dokument mit gleichem Titel und gleichem Erscheinungsjahr identisch ist, und die Dokumentklasse wird dann vom UKA-Agenten beigesteuert.

4.1.3 UKA-Agent

Der UKA-Agent fragt die *Collection of Computer Science Bibliographies* an der Universität Karlsruhe ab – daher der Name. Der UKA-Agent erhält auf seine Anfrage eine HTML-Seite, die Angaben über Publikationen im BIBTEX-Format[1] enthält. Die Anfrage lautet:

[1] Eine gute Einführung in das BIBTEX-Format findet sich in [Lamport, 1995].

```
<table border=1>
.
.
.
<tr><th colspan=3 bgcolor="#FFFFCC">1996</th></tr>
<tr><td align="right" bgcolor="#FFCCCC">15</td><td> </td><td>Katha
rina Morik:
Induktion f&uuml;r alle F&auml;lle!
<a href="../../../journals/ki/ki10.html#Morik96">KI 10</a>(1): 38-39 (1
996)</td></tr>
<tr><td align="right" bgcolor="#FFCCCC">14</td><td> </td><td><a hr
ef="../k/Klingspor:Volker.html">Volker Klingspor</a>,
Katharina Morik,
<a href="../r/Rieger:Anke_D=.html">Anke D. Rieger</a>:
Learning Concepts from Sensor Data of a Mobile Robot.
<a href="../../../journals/ml/ml23.html#KlingsporMR96">Machine Learning
 23</a>(2-3): 305-332 (1996)</td></tr>
.
.
.
Demand and Requirements for Natural Language Systems-Results of an Inqu
iry.
<a href="../../../conf/ijcai/ijcai83.html#Morik83">IJCAI 1983</a>: 647-
649
</td></tr>
</table>
```

Tabelle 4.3: Auschnitt aus dem Quellcode einer DBLP-HTML-Seite.

```
http://liinwww.ira.uka.de/searchbib/index?query=
    <nachname>&case+on&partial=off&results=bibtex&maxnum=<number>
```

Hier wird nur der Nachname einer Person übermittelt, da die Vornamen nicht immer vollständig in den Einträgen enthalten sind. Die Trefferseite enthält nun alle Dokumente, in der der Text `nachname` irgendwo im BIBTEX-Eintrag vorkommt. D. h. hier muss der Agent noch die richtigen Einträge herausfiltern.

Im Beispiel in Tabelle 4.4 sieht man, dass die Angaben innerhalb der `pre`-Umgebungen der HTML-Seite stehen. Sie werden extrahiert und die HTML-Tags entfernt. Die Daten werden anschließend einem BIBTEX-Parser übergeben, der die Inhalte der einzelnen Felder in einer Hashtabelle speichert.

Nun überprüft der UKA-Agent, ob der `nachname` auch wirklich im Autor- oder Editorfeld vorkommt und ob die Vornamen gleich sind. Da es, wie gesagt, auch vorkommen kann, dass Vornamen abgekürzt werden, reicht hier die Gleichheit des Initials des Vornamens.

Eine bisher ungelöste Schwierigkeit bilden gleiche Dokumente mit unterschiedlich geschriebenen Titeln. Solange die Titel gleich geschrieben sind, ist es möglich, doppelte Einträge heraus zu filtern. Es hat sich aber gezeigt, dass Titel oft unterschiedlich geschrieben werden und eine Identifikation des Duplikats erschwert wird. Betrachten wir dazu Tabelle 4.5.

```
<pre>
@Article{Morik85,
  key =       "UM, Goals, Attitudes, Read";
  author =    "<a href="/searchbib/index?query=MorikK&partial=on&
case=on&result=bibtex&maximum=200" onMouseOver="window.status='Click to
 search for publications by this author!'; return true">Katharina Morik
</a> and <a href="..." onMouseOver="...">Claus-Rainer Rollinger</a>",
  title =     "The {Real-Estate} Agent --- Modeling Users by
              Uncertain Reasoning",
  journal =   "AIMag",
  year =      "1985",
  volume =    "6",
  pages =     "44--52",
  source =    "Have",
  notes =     "\input{/home/nyambi/kass/um/notes/morik85}",
}
</pre>
```

Tabelle 4.4: Auschnitt einer HTML-Seite.

Die beiden Einträge stammen aus unterschiedlichen Bibliographien, der erste aus der »Bibliography on Artificial Intelligence« und der zweite aus der »Bibliography of Research in Natural Language Generation« . Wie man sieht unterscheiden sich die beiden Einträge nicht nur in der Art (Unpublished und InProceedings) sondern auch in der Schreibweise des Titels. Im ersten Eintrag wird das Wort »Modeling« mit einem »l« geschrieben, im zweiten Eintrag mit zwei.

Während der Suche speichern die Agenten jede gefundene Publikation in einem Objekt. Insbesondere der UKA-Agent untersucht dann, ob es sich wirklich um eine relevante Publikation handelt. Erst wenn feststeht, dass alle Angaben korrekt sind und die Person, mach der gesucht wurde, entweder als Autor ider Editor in Erscheinung tritt, werden die Publikationen mit dem Information Layer synchronisiert. Wenn eine Publikation noch nicht im IL vorhanden ist, wird sie eingefügt, existiert sie bereits im IL gibt es folgende Möglichkeiten:

1. Die Publikationen sind absolut identisch: Dann wird für die Weiterverarbeitung (synchronisieren der Autoren bzw. Editoren) das im IL existente Publikationsobjekt benutzt.

2. Die neue Publikation hat weniger Daten als das Objekt im IL: Das existente Objekt wird benutzt.

3. Die neue Publikation hat mehr Daten als das Objekt des IL: Die Angaben des IL-Publikations-Objekts werden entsprechend ergänzt.

Hier wird davon ausgegangen, dass Publikationen dann identisch sind, wenn der Titel, das Erscheinungsjahr und die Dokumentenklasse (siehe Tabelle 4.6) gleich sind.

```
@Unpublished{Morik86,
  key =        "UM, NL, Read",
  author =     "Katharina Morik",
  title =      "Modeling the User's Wants",
  year =       "1986",
  note =       "Unpublished paper from UM86, the International
               Workshop on User Modelling, Maria Laach, West Germany",
  source =     "Have",
  notes =      "\input{/home/nyambi/kass/um/notes/morik86}",
}

@InProceedings{morik86,
  author =     "Katharina Morik",
  title =      "Modelling the user's wants",
  booktitle = "First International Workshop on User Modelling",
  year =       "1986",
  address =    "Maria Laach, West Germany",
  month =      aug,
}
```

Tabelle 4.5: Zwei unterschiedliche Einträge eines Dokuments. Der obere hat die Dokumentenklasse Unpublished, der untere InProceedings.

Die Untersuchung hat gezeigt, dass ein Dokument durchaus öfter veröffentlicht wird (siehe Beispiel in Tabelle 4.5, Seite 54). Dann sind aber in der Regel entweder das Erscheinungsjahr oder die Dokumentenklasse unterschiedlich. Ein Artikel wird z. B. als technischer Report (TechReport) von einer Universität veröffentlicht und später bei einer Konferenz vorgestellt. Dieser Artikel erscheint dann auch im Konferenzband (InProceedings). Die Angaben über Dokumentenklassen werden jedoch nur vom UKA-Agenten geliefert, da die DBLP-Bibliographie keine diesbezüglichen Angaben macht.

Eintrag	Beschreibung
Article	Artikel, der in einem Journal oder Magazin erschienen ist.
Book	Ganzes Buch.
InProceedings	Artikel in einem Konferenzband
InCollection	Artikel, der in einer Art Buch veröffentlicht wurde.
TechReport	Report, der von einer bestimmten Institution veröffentlicht wurde.
Unpublished	Zu diesem Dokument gibt es zwar Titel und Autor, aber keinen formellen Verleger (*publisher*).

Tabelle 4.6: Der Eintrag bestimmt die Dokumentenklasse des BIBTEX-Formats (Liste unvollständig).

4.2 Anwendung von MOBAL

In Abschnitt 2.2.4 wurde das System MOBAL bereits vorgestellt. Hier wird dieses System nun praktisch eingesetzt, um Regeln zu lernen, die Beziehungen zwischen Personen anleiten. Ein Zielprädikat soll den Konferenzteilnehmern Treffen mit anderen »passenden« Teilnehmern empfehlen. Diese Zielprädikat heißt »`wants_to_meet`«.

Außer dem Zielprädikat `wants_to_meet` werden noch drei weitere »versteckte« Prädikate betrachtet: `known`, `well_known` und `friends`.

Damit mit MOBAL (RDT) gelernt werden kann, müssen die Daten des Information Layer in die MOBAL-Syntax übertragen werden. Wie die verschiedenen Relationen repräsentiert werden wird in Abschnitt 4.2.1 beschrieben. Diese Faktenbasis wird mit einigen einfachen Regeln um Hintergrundwissen erweitert (Abschnitt 4.2.2). Mit Hilfe von Metaprädikaten werden dann im Abschnitt 4.2.3 (Seite 58) die erwarteten Hypothesen beschrieben.

Die Verknüpfungen der Daten, die im Information Layer durch die Ontologie gegeben ist, müssen in eine andere Repräsentation gebracht werden. Im Information Layer ist das Objekt `Person` mit verschiedenen anderen Objekten verknüpft. Relevant sind hier die Objekte `Publication` und `Project`. Diese Verknüpfungen müssen explizit dargestellt werden. Dazu werden (bis auf die Personen) die Identifikationsnummern der Objekte benutzt. Doch nicht nur die Verknüpfungen, auch die Daten brauchen eine andere Repräsentation. Wenn z. B. die Publikationsjahre Einfluss auf das Lernergebnis haben sollen, müssen sie so repräsentiert werden, dass sie Teil der Hypothese sind. Außerdem sind Anpassungen zur Darstellung von Konstanten in MOBAL notwendig. So dürfen Terme eines Faktums keine Leerstellen oder andere Sonderzeichen haben.

4.2.1 Die Relationen zwischen Personen

Aus allen Beziehungen, die zwei Personen in der hier betrachteten Domäne zueinander haben können, werden Prädikate generiert. Folgende Relationen sind möglich:

1. Relationen durch gemeinsame Publikationen
   (a) Autor – Autor
   (b) Autor – Editor
   (c) Editor – Editor
   (d) Editor – Autor
2. Relationen durch gemeinsame Projekte
   - Projektpartner – Projektpartner

```
autor_autor(person1, person2, startyear, endyear, frequence).
editor_author(person1, person2, startyear, endyear, frequence).
autor_editor(person1, person2, startyear, endyear, frequence).
editor_editor(person1, person2, startyear, endyear, frequence).
pp(person1, org1, project, person2, org2, startyear, endyear).

sum_of_pubs(person1, person2, startyear, endyear, sumofpubs).
frequence_of_pubs(person1, person2, startyear, endyear, frequence).
period_of_pubs(person1, person2, startyear, endyear, numberofyears).
```

Tabelle 4.7: Prädikate die die Relationen zwischen Personen beschreiben.

Daraus ergeben sich folgende Prädikate (Detaildarstellung in Tabelle 4.7):

- `autor_autor`

  Dieses Prädikat drückt die Koautorenschaft aus. Enthalten sind außerdem das Jahr der ersten und letzten gemeinsamen Publikation als Koautoren und die Häufigkeit (Publikationsdurchschnitt pro Jahr).

- `editor_autor`

  Eine Person ist Editor einer Veröffentlichung, in der eine andere Person als Autor auftritt. Auch hier wird wieder das erste und letzte Jahr dieser Relation und deren Frequenz ausgedrückt.

- `autor_editor`

  Das umgekehrte Verhältnis.

- `editor_editor`

  Zwei Personen sind Koeditoren.

- `pp`

  Zwei Personen sind Projektpartner. Dieses Prädikat enthält neben dem gemeinsame Projekt noch die Organisationen der beiden Partner, und das Start- und Enddatum des Projekts.

Diese Prädikate modellieren die einzelnen Beziehungen zwischen den Personen. Was fehlt sind Angaben über alle gemeinsamen Publikationen – also eine Zusammenfassung der Relationen. Dies wird mit einige *Hilfsprädikaten* ausgedrückt:

- `sum_of_publications`

  Enthält die Summe aller gemeinsamen Publikationen. Es zählt das Jahr der alleresten und allerletzten gemeinsamen Veröffentlichung, egal in welcher Relation die beiden Personen in dieser Publikation zueinander stehen.

- `period_of_pubs`

  Repräsentiert die Anzahl der Jahre in denen gemeinsam publiziert wurde.

- `frequence_of_pubs`

  Beschreibt die durchschnittliche Veröffentlichungsfrequenz. Wenn z. B. in einem Jahr 5 Dokumente mit einem bestimmten Koautor veröffentlicht wurden (und zwar mit einer Frequenz von 5), und 10 Jahre später eine Person als Editor einer Publikation auftritt, bei der die andere Person Autor ist (Frequenz = 1), dann ergibt sich eine Gesamtfrequenz von 0,6.

4.2.2 Modellierung des Hintergrundwissens

Diese Fakten allein reichen jedoch noch nicht zum Lernen aus. Das Hintergrundwissen legt zusätzlich einige Größen fest, die in den Fakten nicht enthalten sind. Wir kennen zwar die Publikationsfrequenz, möchten aber ausdrücken, ob diese Frequenz besonders hoch oder niedrig ist. Dazu werden folgenden Regeln verwendet:

```
frequence_of_pubs(P1,P2,Y1,Y2,F) & leq(F,1)
                 --> pub_seldom(P1,P2).
frequence_of_pubs(P1,P2,Y1,Y2,F) & leq(F,2) & gt(F,1)
                 --> pub_middel(P1,P2).
frequence_of_pubs(P1,P2,Y1,Y2,F) & gt(F,2)
                 --> pub_frequent(P1,P2).
```

Die Publikationsfrequenz wird so in drei Gruppen aufgeteilt, die Aussagen über die Höhe der Frequenz machen:

- Selten, wenn im Schnitt einmal oder weniger pro Jahr veröffentlich wurde,
- mittel, wenn mehr als eine, aber weniger als zwei Veröffentlichungen pro Jahr stattfanden, und
- häufig, wenn mehr als 2-mal pro Jahr publiziert wurde.

Auch über den Zeitraum der Publikationen werden Aussagen hinzugefügt:

```
period_of_pubs(P1,P2,Y1,Y2,P) & leq(Y2,1989)
              --> period_long_ago(P1,P2).
period_of_pubs(P1,P2,Y1,Y2,P) & leq(Y2,1996) & gt(Y2,1989)
              --> period_medium(P1,P2).
period_of_pubs(P1,P2,Y1,Y2,P) & gt(Y2,1996)
              --> period_current(P1,P2).
```

Diese Rollen modellieren, wie lange der Zeitraum gemeinsamer Veröffentlichungen zurückliegt, bzw. ob er gegenwärtig ist.

- Lange her, wenn der Zeitraum vor 1989 endet und danach keine gemeinsame Veröffentlichung war,
- mittel lange her, wenn das Publikationsintervall nach 1989 beginnt, aber vor 1996 endet und
- gegenwärtig, wenn der Zeitraum nach 1996 endet.

Schließlich soll die Anzahl der Publikationen auch eine Rolle spielen und wird durch folgende Regeln modelliert:

```
number_of_pubs(P1,P2,Y1,Y2,N) & eq(N,1)
             --> one_publication(P1,P2).
number_of_pubs(P1,P2,Y1,Y2,N) & gt(N,1) & le(N,5)
             --> some_publication(P1,P2).
number_of_pubs(P1,P2,Y1,Y2,N) & gt(N,5)
             --> many_publication(P1,P2).
```

Entweder man hat:

1. genau eine gemeinsame Veröffentlichung,
2. einige Veröffentlichungen (zwischen 2 und 5) oder
3. viele (mehr als 5).

Durch die Einführung dieser Rollen, wird aus den Grundfakten das Hintergrundwissen abgeleitet, das dadurch ebenfalls in Form von Fakten vorliegt. Diese Fakten sind jedoch abhängig von den Grundfakten und sind deshalb veränderlich.

4.2.3 Hypothesenraum

RDT braucht nicht nur eine Faktenbasis zum Lernen, sondern auch ein Regelmodell, das beschreibt, wie mögliche Regeln auszusehen haben. Mit anderen Worten, dieses Modell bestimmt die möglichen Hypothesen – es legt die Hypothesensprache fest.

Folgende Metaprädikate werden verwendet:

```
mp1(P, Q) : P(X, Y) --> Q(X, Y).
mp11(P, Q) : P(X, Y, A, B, C) --> Q(X, Y).
mp21(P1, P2, Q) : P1(X, Y) & P2(X, Y) --> Q(X, Y).
mp22(P1, P2, Q) : P1(X, Y) & P2(X, O1, PRO, Y, O2, Y1, Y2)
                  --> Q(X, Y).
mp3(P1, P2, Q) : P1(X, Y, A, B, C) & P2(X, Y) --> Q(X, Y).
```

mp1 erlaubt Regeln, die von einem 2-stelligen Prädikat auf ein anderes 2-stelliges Prädikat schließen. Damit wird das Hintergrundwissen in den Hypothesenraum aufgenommen. mp11 passt auf jedes 5-stellige Prädikat und erlaubt die Ableitung eines 2-stelligen. Die Regeln, die dem Schema mp22 entsprechen müssen das Prädikat pp (für Projektpartner) enthalten, da es die einzigen 7-stelligen Fakten sind.

Die Zielprädikate sind ebenfalls 2-stellig und können so zu Literalen von Regeln werden, die sich aus den Metaprädikaten ergeben.

4.2.4 Lernbeispiele

Die Daten von vier Personen wurden extrahiert und die Beziehungen von diesen Personen persönlich klassifiziert um Testdaten zu erhalten. Drei Personen bildeten die Trainingsmenge, die Daten der vierten Person diente zum Test.

Es wurden sowohl positive als auch negative Beispiele verwendet:

- `known(person1,person2)` und `not(known(person1,person2))`,
- `well_known(p1,p2)` und `not(well_known(p1,p2))`,
- `friends(p1,p2)` und `not(friends(p1,p2))`,
- `wants_to_meet(p1,p2)` und `not(wants_to_meet(p1,p2))`.

known soll ausdrücken, dass sich person1 und person2 kennen. well_known repräsentiert gute Bekanntschaft und friends ein freundschaftliches Verhältnis in dem Sinne, dass man mit diesen Personen gerne zusammenarbeitet und sie ideale Kandidaten für zukünftige, gemeinsame Projekte geeignet wären. wants_to_meet – das eigentliche Zielprädikat – soll schließlich die Treffen vorhersagen. Die ersten drei Prädikate haben eine Teilmengenbeziehung, die in Abbildung 4.1 skizziert ist.

Freunde sind auf jeden Fall gute Bekannte – gute Bekannte auf jeden Fall Bekannte. Vielleicht ist diese Ordnung ja interessant für die Ableitung des Prädikats wants_to_meet.

4.2.5 Akzeptanzkriterien

Die Akzeptanzkriterien bestimmen, wann eine Regel vom System angenommen wird. Die Möglichkeiten dieses Kriterium in RDT zu bestimmen, wurden in Abschnitt 2.2.4 (Seite 32) eingeführt. Mit Hilfe der Parameter von RDT (siehe Anhang C.1) lässt sich dieses Kriterium einstellen und so der Hypothesenraum einschränken.

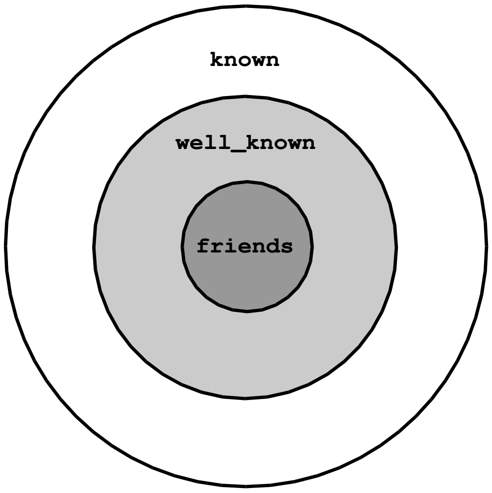


Abbildung 4.1: Mengenordnung der Prädikate.

Die Regeln wurden mit drei unterschiedlichen Akzeptanzkriterien, die im nächsten Kapitel (Abschnitt 5.2.2, Seite 69) dargestellt sind, gelernt. Dort werden auch die Lernergebnisse präsentiert.

Kapitel 5

Ergebnisse

In diesem Kapitel werden die Ergebnisse des Agentensystems (Abschnitt 5.1) und die Lernergebnisse (Abschnitt 5.2, Seite 66) präsentiert. Der Abschnitt 5.3 (Seite 72) beendet das Kapitel.

5.1 Agentenperformanz

Die Performanz des Agentensystems ist abhängig vom Netzverkehr und der Auslastung der angesprochenen Server. 15 Suchläufe des Systems wurden protokolliert und sind in Abbildung 5.1 graphisch dargestellt. Die längste Suche dauerte demnach 160 Sekunden und die kürzeste war nach 2 Sekunden beendet.

Die längste Suche dauerte demnach 160 Sekunden und die kürzeste war nach 2 Sekunden beendet. Im Schnitt sind die Agenten nach ca. 44,3 Sekunden mit ihrer Suche fertig. Das überrascht, wenn hier die Ergebnisse des Projekt-Agenten verglichen werden (siehe Abbildung 5.5). In der Regel werden aber sehr wenig Projekte gefunden, was sich natürlich positiv auf die Gesamtlaufzeit auswirkt

5.1.1 Publikations-Agenten

Die Datenquellen enthielten zum Testzeitpunkt (Januar–Juli 1999) keine Publikationen aus dem Jahr 1999. Als Maß wurden die Angaben auf den Webseiten der betreffenden Personen genommen. Gab es dort keine Angaben über Publikationen, wurden die Angaben der übergeordneten Organisationen bzw. Institutionen über Publikationen herangezogen. Die Methodik war dabei wie folgt: Die eigenen Angaben über Veröffentlichungen wurden mit den über die Datenquellen verfügbaren und den von den Agenten gefundenen verglichen.

Untersuchungen der eigenen Angaben zeigten, dass unter den Veröffentli-

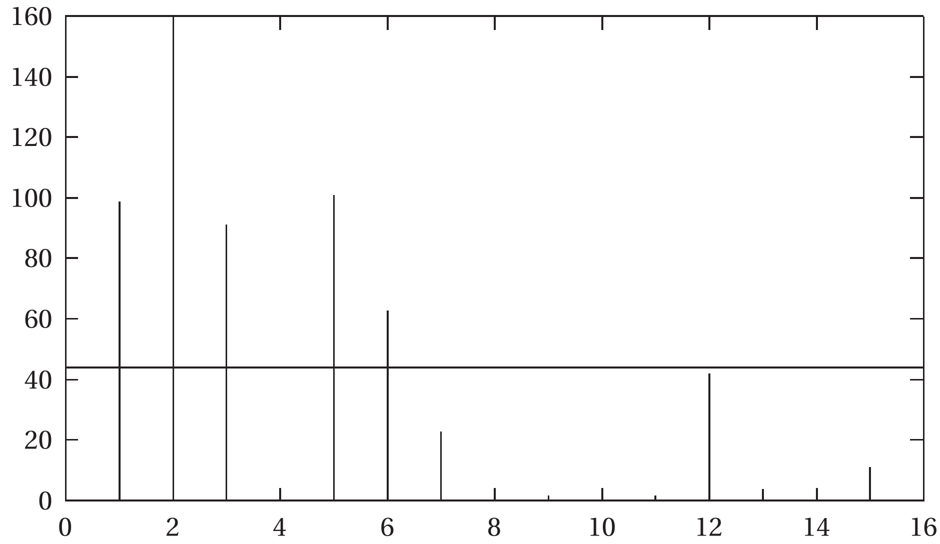


Abbildung 5.1: Einige Laufzeiten des Agentensystems und durchschnittliche Suchdauer (y-Achse = Sekunden).

chungen auch Diplomarbeiten und andere interne Berichte angegeben wurden. Diese Art von Publikationen werden sicherlich nicht immer an die Organisationen, die ihre Angaben über Veröffentlichungen den Bibliotheken zur Verfügung stellen, übermittelt und können deshalb auch nicht gefunden werden. Es kann auch vorkommen, dass Artikel in Informatik-fremden Organen erschienen sind. Da die Datenquellen auf das Gebiet der Informatik spezialisiert sind, ist es verständlich, wenn diese Publikationen dort nicht gefunden werden können. Drei Auswertungen sind beispielhaft in Tabelle 5.1 dargestellt.

Die Zeile »Gefundene Dokumente« gibt die Anzahl der Veröffentlichungen an, die die entsprechenden Quellen (AGENT, QUELLE, EIGENE ANGABEN) ausweisen. Im oberen Beispiel haben die Agenten also 10 Veröffentlichungen gefunden, in den Datenquellen werden insgesamt 13 angegeben und nach eigenen Angaben – auf der Homepage der Person – wurden 18 Dokumente veröffentlicht. Demnach hätten die Agenten nur 55,5 % der Dokumente gefunden, aber die Datenquellen hätten auch nur 72,2 % hergegeben. Schließlich können die Agenten nur die Daten der ihnen bekannten Quellen finden.

Dieses Ergebnis relativiert sich, wenn man sich die »eigenen Angaben« dieser Person genauer ansieht. So ist ein Artikel zweimal veröffentlicht worden, einmal in einem Konferenzband und ein anderesmal in einem Journal. Für die Agenten reicht Gleichheit des Titels und der Autoren nicht aus. Und wenn ein Artikel in einem Konferenzband erschienen ist, und später noch einmal

	AGENT	QUELLE	EIGENE ANGABEN
Gefundene Dokumente	10	13	18
Mehrfache Veröffentlichungen	2	2	2
Identität	1	4	
Interne Veröffentlichung			2
Diplomarbeit			1
Veröffentlichung 1999			5
Relevante Dokumente	7	7	8
Ergebnis	87,5 %	87,5 %	
Gefundene Dokumente	11	16	11
Mehrfache Veröffentlichungen	1	3	
Identität		3	
Unveröffentlicht			1
Relevante Dokumente	10	10	10
Ergebnis	100 %	100 %	
Gefundene Dokumente	24	34	31
Mehrfache Veröffentlichungen	6	6	
Identität		10	
Fremde Organe			3
Veröffentlichung 1999			1
Relevante Dokumente	18	18	27
Ergebnis	66,6 %	66,6 %	

Tabelle 5.1: Auswertungsbeispiele.

in einem Buch, dann sind das für den Agenten zwei Veröffentlichungen (was ja auch tatsächlich stimmt!).

Wie oben erwähnt enthalten die Datenquellen noch keine Angaben über Publikationen aus dem Jahr 1999. Die Person des ersten Tabelleneintrages hat bereits achtmal publiziert. Desweiteren befindet sich unter den eigenen Angaben auch die Diplomarbeit. Es wird nicht erwartet, dass Bibliographien Angaben über diese Dokumente haben, sie zählen vielmehr zu den internen Publikationen, von denen es auch noch andere Arten gibt. Nicht alle Arbeiten werden auch von offiziellen Organen veröffentlicht und sind somit für die Bibliographien verfügbar.

Nach Berücksichtigung dieser Kriterien und Abzug der entsprechenden Dokumente ergeben sich die Zahlen der Reihe »Ergebnisse«. Und hier zeigt sich, dass die Agenten sieben von acht Publikationen gefunden haben – das sind 87.5 %. Bei der fehlenden Publikation war die Person Editor, sie war den Datenquellen aber nicht bekannt.

Da die Bibliographien auch eine gewisse Schnittmenge haben, sind auch hier Abzüge zu machen. Im Beispiel sind insgesamt vier Publikationen identisch – der Agent entdeckt aber leider nur drei. Das liegt in den meisten Fällen an

Tippfehlern. So werden insbesondere bei den BIBTEX-Einträgen gerne Fehler in der Klammerung gemacht, die von den Agenten nicht bemerkt werden.

Im letzten dargestellten Beispiel in Tabelle 5.1 liegt die Trefferquote nur bei 66,6 %.

Tabelle 5.2 und Abbildung 5.2 enthalten die Ergebnisse von 15 Testpersonen.

TESTPERSON	VERÖFFENTLICHUNGEN gefunden	VERÖFFENTLICHUNGEN eigene Angaben	ABDECKUNG
Person1	7	8	87,5 %
Person2	3	4	75 %
Person3	10	10	100 %
Person4	4	6	66,67 %
Person5	2	8	25 %
Person6	2	7	28,57 %
Person7	18	27	66,67 %
Person8	24	30	80 %
Person9	5	11	45,45 %
Person10	5	18	27,78 %
Person11	4	6	66,67 %
Person12	13	13	100 %
Person13	6	9	66,67 %
Person14	15	18	83,33 %
Person15	29	33	87,88 %
		Schnitt	67,15 %

Tabelle 5.2: Von den Agenten gefundene Veröffentlichungen im Vergleich mit den Angaben auf der persönlichen Webseite.

Es ergibt sich also eine durchschnittliche Abdeckung von 66,77 %. Dieses Ergebnis läßt sich durch die Ergänzung neue Informationsagenten sicherlich verbessern.

Abbildungen 5.3 und 5.4 zeigen Laufzeiten der Publikations-Agenten.

Die Laufzeiten des UKA-Agenten sind offensichtlich unabhängig von der Anzahl der gefundenen Dokumente, da alle gefundenen Dokumente auf einer HTML-Seite dargestellt sind. Das sollte jedoch beim DBLP-Agenten genauso sein. Auch hier finden sich alle Dokumente auf einer HTML-Seite. Eine Erklärung ist, dass das Parsen der HTML-Seite mehr Zeit in Anspruch nimmt als das Parsen von im BIBTEX-Format vorliegenden Einträgen.

5.1.2 Projekt-Agent

Dieser Agent braucht für seine Suche länger als die Publikations-Agenten, da für jedes gefundene Projekt einem Hyperlink gefolgt und eine eigene HTML-

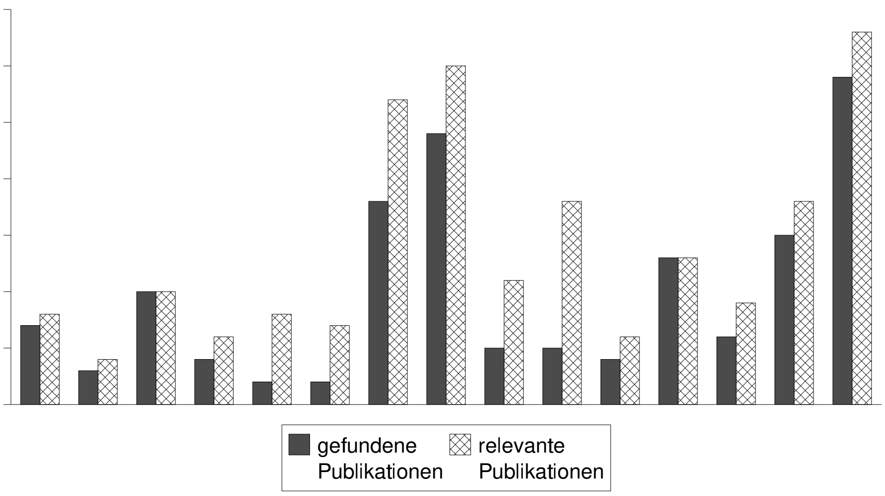


Abbildung 5.2: Verhältnis zwischen relevanten und gefundenen Veröffentlichungen.

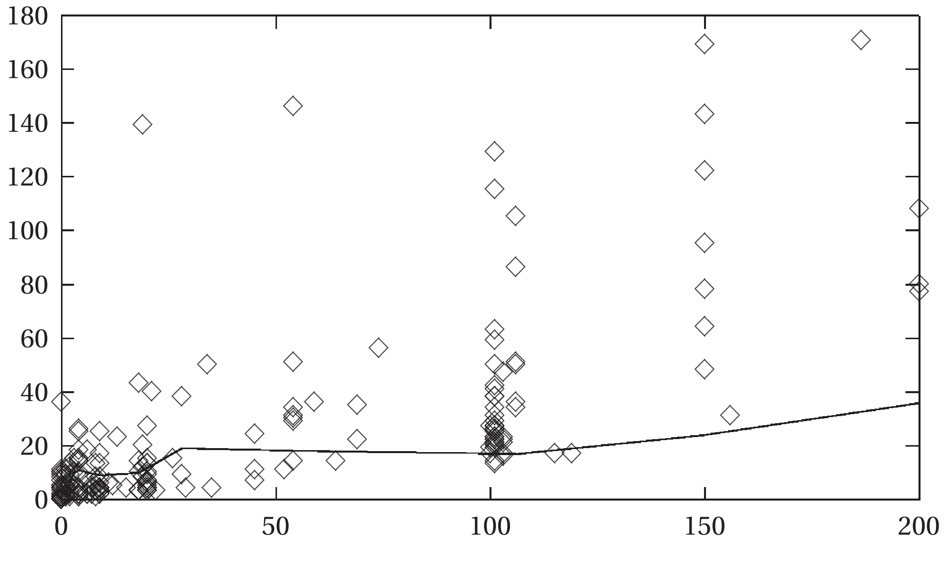


Abbildung 5.3: Laufzeiten des UKA-Agent (x-Achse = gefundene Publikationen, y-Achse = Sekunden).

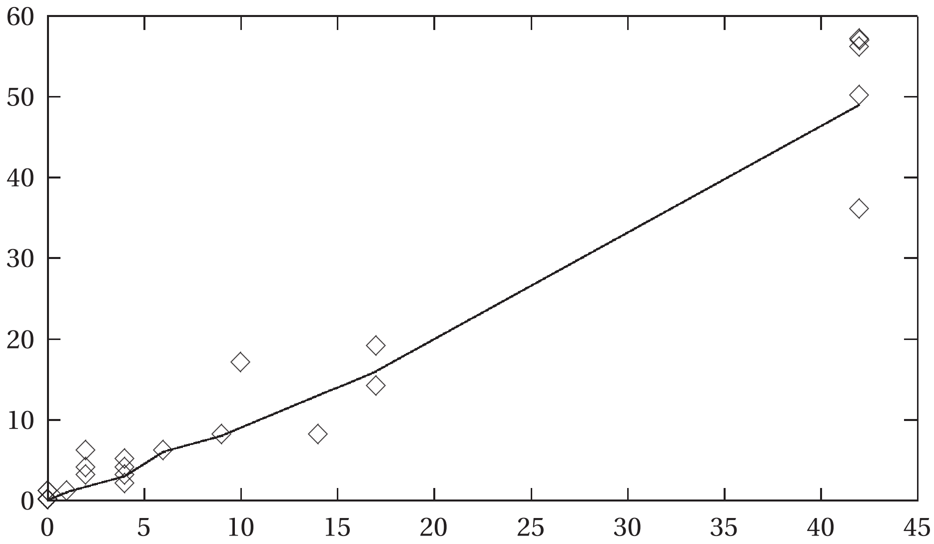


Abbildung 5.4: Laufzeiten des DBLP-Agent (x-Achse = gefundene Veröffentlichungen, y-Achse = Sekunden).

Seite geparst werden muss. Die Antwortzeiten des CORDIS-Servers sind auch ein wenig höher, als die der anderen Quellen, was dazu führt, dass eine erfolglose Suche ebenfalls um einiges länger dauert, als bei den anderen Agenten.

Da die Datenquelle nur Daten von Projekten anbietet, die die EU fördert oder gefördet hat, können natürlich keine Projekte gefunden werden, die von anderen Institutionen unterstützt oder geleitet werden, z. B. Projekte der Deutschen Forschungsgesellschaft (DFG). Leider enthalten die Webseiten der DFG (noch) keine Angaben über abgeschlossene, laufende oder geplante Projekte. Auch die Webseiten des Bundesministeriums für Bildung und Forschung (BMBF) geben keine derartigen Auskünfte.

Abbildung 5.5 zeigt die Laufzeit dieses Agenten. Die Laufzeit nimmt proportional mit den gefundenen Projekten zu.

5.2 Lernergebnisse

Die Trainingsmenge bestand aus drei Beispielen mit insgesamt 2631 Fakten. Davon waren 668 Fakten Zielkonzepte. Die Testdaten bestanden aus insgesamt 1506 Fakten. 163 Personen waren vorher von der Testperson entsprechend den Zielprädikaten klassifiziert worden. Die Regeln für die vier Konzepte wurden mit unterschiedlichen Abhängigkeiten gelernt:

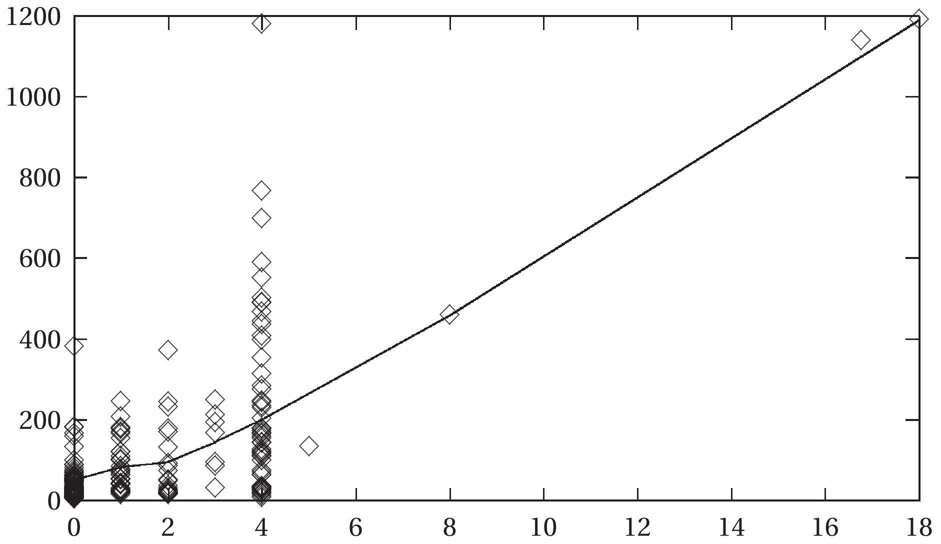


Abbildung 5.5: Laufzeiten des CORDIS-Agent (x-Achse = gefundene Projekte, y-Achse = Sekunden).

1. Alle Zielbegiffe,
2. ohne `friends`,
3. ohne `friends` und `well_known`,
4. nur abhängig von `known` und
5. nur das Zielkonzept `wants_to_meet`.

Das wichtigste Zielkonzept war `wants_to_meet`. Es wurden 12 Lernläufe durchgeführt, die sich durch die Einbindung der »versteckten« Prädikate und den Akzeptanzkriterien (AK1[1]–AK3) unterschieden.

	Kriterium	
	confirmation	*pruning*
AK1	`pos>=3 & (pos>=0.5*total & (neg=<3 & neg=<0.1*total))`	`pos<3`
AK2	`pos>=3 & (pos>=0.8*total & neg=<0.1*total)`	`pos<3`
AK3	`pos>=3 & neg<=0.3*total`	`pos<3`

Jedes dieser Akzeptanzkriterien besteht aus zwei Komponenten, dem *confirmation*- und dem *pruning*-Kriterium. Das erste sorgt dafür, dass die Regeln

[1] Default-Parameter von RDT.

nicht zu allgemein werden – es verhindert das sogenannte *overfitting*; das zweite sorgt dafür, dass die Regeln nicht zu speziell werden.

5.2.1 Performanzmaße

Die Bewertung des Gelernten erfolgt im Bereich des maschinellen Lernens nach bestimmten Gesichtspunkten. Am Besten lassen sich diese Maße mit Hilfe einer sogenannten *contingency table*, die in Tabelle 5.3 gezeigt ist, erklären (aus [Lavrac et al., 1999]).

	E	$\overline{E}$
R	A	B
$\overline{R}$	C	D

Tabelle 5.3: Contingency Table.

Die Spalte E enthält alle positiven Beispiele – die Mengen A und C –,die Spalte $\overline{E}$ enthält alle negativen Beispiele. Die Zeile R enthält alle von der Regel als positiv vorhergesagten Beispiele, Zeile $\overline{R}$ alle als negativ vorhergesagten. Die vier Mengen (A–D) können wie folgt charakterisiert werden (Tabelle 5.3):

A Richtig klassifiziert. Beispiel- und Regelklassifikation stimmen überein.

B Falsch klassifiziert. Die Regeln leiten negative Beispiele ab.

C Falsch klassifiziert. Die Regeln leiten positive Beispiele nicht als solche ab.

D Richtig klassifiziert. Die Regeln sagen die negativen Beispiele ($\overline{E}$) als nicht wahr vorher.

Diese Kategorien wurden zur Messung der Lernperformanz benutzt, für die es verschiedene Methoden gibt.

Accuracy ist die typischste Art im Bereich des maschinellen Lernens die Performanz zu messen.

$$Accuracy = \frac{A + D}{A + B + C + D} \tag{5.1}$$

$A+D$ ist die Anzahl korrekt klassifizierter Personen und $A+B+C+D$ die Zahl aller Klassifikationen. Ein Wert von 1 (= 100 %) ist traumhaft. Beispielsweise hat das System 10 von 100 Personen als Menschen klassifiziert, die man gerne treffen möchte – die Person selbst nennt aber nur 6 dieser 10 Personen. D. h. A = 6, B = 4, C = 0 und D = 90. Daraus ergibt sich eine Accuracy von 96/100 = 0,96 = 96 %. Ein hoher Wert für die Accuracy ist also immer erstrebenswert und gibt Aufschluss über die Qualität der gelernten Regeln.

Correctness ist das Verhältnis zwischen den vorhergesagten positiven Beispielen und allen als positiv vorhergesagten Beispielen – die *Richtigkeit* der Regeln. Im Information Retrieval nennt man diesen Wert *Precision*.

$$\mathsf{Correctness} = \frac{\mathsf{A}}{\mathsf{A} + \mathsf{B}} \tag{5.2}$$

Coverage ist das Verhältnis von vorhergesagten positiven Instanzen, zu allen positiven Instanzen. Wie der Name ausdrückt: Die *Abdeckung* der Beispielmenge durch die Ableitungen der Regeln. Im IR wird dieser Wert *Recall* genannt.

$$\mathsf{Coverage} = \frac{\mathsf{A}}{\mathsf{A} + \mathsf{C}} \tag{5.3}$$

In unserem Beispiel – $\mathsf{A} = 6$, $\mathsf{B} = 4$, $\mathsf{C} = 0$ und $\mathsf{D} = 90$ – ergibt sich eine Correctness von $6/10 = 0{,}6 = 60$ % und eine Coverage von $6/6 = 1 = 100$ %. D. h. es werden zwar alle positiven Beispiele aus der Testmenge durch Regeln abgedeckt, aber auch einige negative Beispiele (40 %).

Es kommt jedoch auch immer darauf an, mit welchen Kosten diese Klassifikationen verbunden ist. Wenn es z. B. darum geht Patienten als geheilt zu entlassen, ist es schlimmer einen kranken Menschen nach Hause zu schicken, als den Krankenhaustagessatz für einen Gesunden zu bezahlen. Es ist also nicht schlimm, wenn die Regeln mehr Treffen empfehlen, als von den Personen gewünscht wird. Es sollte also Wert auf eine hohe Coverage gelegt werden. Ein paar falsche Empfehlungen könnten im schlimmsten Fall abgelehnt werden. Andererseits sollte das System mit der Klassifikation als Freund etwas vorsichtiger sein. Hier sollte eine hohe Correctness gefordert werden, da es besser ist, eine Person zu *übersehen*, als jemanden als Freund zu klassifizieren, den man überhaupt nicht leiden mag.

Bei der Bewertung der Klassifikation von `wants_to_meet` sollte also ein hoher Coverage-Wert angestrebt werden, schließlich besteht ja kein Treff-Zwang, es sind lediglich Empfehlungen, die das System gibt.

Die anderen Klassifikationen werden nur benutzt, um damit vielleicht eher ein Treffen vorhersagen zu können. Trotzdem sollte bei dem Prädikat `freund` eher auf einen hohen Correctness-Wert geachtet werden. Bei `well_known` muss nicht ganz so streng verfahren werden, immerhin ist es nicht tragisch, wenn die Regeln zwei Personen für gute Bekannte halten, die es eigentlich gar nicht sind. Zum anderen handelt es sich auch hier um ein »verstecktes« Prädikat genau wie `known`.

5.2.2 Auswertung

Im Laufe der Versuche zeigte sich, dass das Prädikat `friends` von den Testpersonen unterschiedlich bewertet wurde. Die einen möchten einen Freund unbedingt treffen, wenn er an derselben Konferenz teilnimmt. Die andere argumentieren, dass sie gute Freunde auf jeden Fall treffen, und sie diese Personen deshalb nicht in die Kategorie der Personen aufgenommen haben, die sie treffen wollen. Eine gelernte Regel war z. B.:

```
friends(X, Y) --> wants_to_meet(X, Y).
```

Ein Vergleich mit Abbildung 4.1 (Seite 60) zeigt, dass `wants_to_meet` sich in diese Ordnung einbetten würde. Durch die unterschiedlichen Klassifikationen der Relation `friends` durch die Testpersonen, führt diese Regel jedoch zu falschen Vorhersagen. Deshalb wurden unterschiedliche Lernläufe durchgeführt, bei denen zuerst das Prädikat `friends`, dann `well_known` und schließlich `known` weggelassen wurde, um hier einen Vergleich zu haben. Durch das Weglassen einiger Prädikate wurde natürlich auch der Hypothesenraum verändert. Die Hypothesen sind demnach auch immer ein wenig unterschiedlich.

Tabellen 5.4–5.7 enthalten die Ergebnisse verschiedener Lernläufe. Die Abhängigkeit der Vorhersage `wants_to_meet` von den anderen Prädikaten ist unterschiedlich.

Zielprädikat	AK	Anzahl gelernter Regeln	Coverage	Correctness	Accuracy
`known`	AK1	16	100 %	72 %	72 %
	AK2	19	100 %	72 %	72 %
	AK3	19	100 %	72 %	72 %
`well_known`	AK1	18	45 %	39 %	40 %
	AK2	8	32 %	86 %	65 %
	AK3	5	50 %	81 %	71 %
`friends`	AK1	4	25 %	45 %	40 %
	AK2	2	21 %	62 %	87 %
	AK3	3	42 %	62 %	88 %
`wants_to_meet`	AK1	6	12 %	89 %	63 %
	AK2	2	10 %	87 %	62 %
	AK3	7	58 %	75 %	73 %

Tabelle 5.4: Ergebnisse des Lernens von allen Prädikaten.

Die Prädikate `known` und `well_known` haben bei den Akzeptanzkriterien AK1 und AK2 keinen Einfluss auf die für `wants_to_meet` gelernten Regeln.

Die Regeln, die für `known` gelernt wurden, klassifizieren alle Personen, die eine Verknüpfung über Veröffentlichungen oder Projekte haben, in die Kategorie der Bekannten. Die gelernten Regeln bringen hier keinen Vorteil und das Konzept `known` kommt auch nicht in den Rümpfen der Regeln für die anderen Konzepte vor – es kann weggelassen werden.

Zielprädikat	AK	Anzahl gelernter Regeln	Coverage	Correctness	Accuracy
`known`	AK1	15	100 %	72 %	72 %
	AK2	18	100 %	72 %	72 %
	AK3	18	100 %	72 %	72 %
`well_known`	AK1	17	45 %	39 %	40 %
	AK2	9	32 %	86 %	65 %
	AK3	4	50 %	81 %	71 %
`wants_to_meet`	AK1	2	7 %	83 %	61 %
	AK2	1	-	-	58 %
	AK3	6	58 %	75 %	73 %

Tabelle 5.5: Ergebnisse des Lernens ohne `friends`.

Zielprädikat	AK	Anzahl gelernter Regeln	Coverage	Correctness	Accuracy
`known`	AK1	14	100 %	72 %	72 %
	AK2	17	100 %	72 %	72 %
	AK3	17	100 %	72 %	72 %
`wants_to_meet`	AK1	2	7 %	83 %	61 %
	AK2	1	-	-	58 %
	AK3	5	31 %	87 %	69 %

Tabelle 5.6: Ergebnisse des Lernens ohne `friends` und `well_known`.

Das Akzeptanzkriterium AK2 sorgte dafür, dass die gelernten Regeln eine hohe Correctness haben, aber eine schlechte Coverage. Im Falle des Lernlaufes ohne den »versteckten« Prädikaten wurden keine Regeln gelernt (Tabelle 5.7). Auch die Ergebnisse in Tabellen 5.6 und 5.5 waren mit AK2 schlecht. Es wurden keine Personen vorhergesagt, die sich Treffen möchten – so konnte ein Wert für die Accuracy berechnet werden, aber keiner für Coverage und Correctness (da $A = 0$!).

Mit dem Akzeptanzkriterium AK3 wurden die besten Ergebnisse erzeilt. AK3 erlaubte im *confirmation*-Kriterium eine höhere Anzahl abgedecker negativer Instanzen, mit dem Hintergrund, dass empfohlene Treffen nicht unbedingt wahrgenommen werden müssen. Ein Vergleich der Tabellen zeigt, dass das Konzept `friends` keinen Einfluss auf die Güte des Lernergebnisses für `wants_to_meet` hat, wohl aber das Konzept `well_known`.

Zielprädikat	AK	Anzahl gelernter Regeln	Coverage	Correctness	Accuracy
`wants_to_meet`	AK1	2	7 %	83 %	61 %
	AK2	-	-	-	-
	AK3	5	31 %	87 %	69 %

Tabelle 5.7: Ergebnisse des Lernens von `wants_to_meet`.

5.2.3 Die Systemregeln

Da das AK3 die besten Ergebnisse geliefert hat, wurden einige dieser Regeln als Templates für den Information Layer bestimmt (Tabelle 5.8).

`wants_to_meet(X, Y)`	`many_publications(X, Y)`
	`editor_editor(X, Y, A, B, C)`
	`freq_frequent(X, Y) & pp(X, O1, Pro, Y, O2, Y1, Y2)`
	`author_author(X, Y, A, B, C) & freq_frequent(X, Y)`
	`author_editor(X, Y, A, B, C) & freq_frequent(X, Y)`

Tabelle 5.8: Diese Regeln werden in den Information Layer integriert.

Hier bleibt es abzuwarten, wie gut diese Regeln auf einer Konferenz die Treffen vorhersagen.

5.3 Ausblick

Die Idee, das Interesse von Personen aus den persönlichen Webseiten zu extrahieren wurde nach einigen Stichproben aufgegeben. Mit diesem Interesse könnten Treffen empfohlen werden, die auf einem gemeinsamen Interesse basieren. [Basu et al., 1999] betrachten den kompletten Inhalt einer Homepage als *abtract* eines Interesses. Dieses *abstrakte* Interesse wird dazu benutzt, Artikel, die für eine Konferenz eingereicht werden, den geeignetsten Gutachtern zukommen zu lassen. Ihr bestes Ergebnis hatte eine *Precision* von 37,4 %.

Ein nach wie vor schwieriger Punkt ist die automatische Informations-Extraktion aus Webseiten. Im Laufe dieser Untersuchung hat sich herausgestellt, dass HTML-Dokumente zwar durch die Tags stukturiert werden können, diese Strukturen jedoch nicht ausreichen, um Rückschlüsse auf die enthaltenen Daten zu gewinnen [Freitag, 1998]. Der einzige Nutzen im Hinblick auf Agenten besteht in den Hyperlinks. Da inzwischen immer mehr Firmen dazu übergehen Bilder an Stelle von Text zu benutzen, wird es höchste Zeit für standardisierte Inhaltsbeschreibungen.

Die Bedeutung des Internets bzw. des WWW als Informationssystem wird in Zukunft noch zunehmen. Es wäre deshalb wünschenswert, wenn es mehr Informationen über Inhalte gäbe und es so zu eine besseren Strukrurierung käme. Die Einführung von XML als neuen Web-Standard ist sicherlich ein Gewinn, wenn diese Sprache so genutzt wird, wie beabsichtigt. Auch HTML bietet Strukturierungsmöglichkeiten, die jedoch leider sehr begrenzt sind und meistens nicht genutzt werden.

Durch die wachsende Vernetzung der Computer und die wachsende Zahl von Datenbank- und Informationsanbietern, entsteht ein riesiges verteiltes Informationssystem. Sicherlich ist es für die Bezeichnung als *Wissensbasis* noch ein wenig früh, doch wir befinden uns auf dem Weg dorthin.

Anhang A

Klassenbeschreibung des Agentensystems

It's a jungle *out there,*
So drink your JAVA

T-SHIRT VON *Printer's Inc Cafe*, PALO ALTO, CALIFORNIA

Im nächsten Abschnitt werden die einzelnen Klassen beschrieben und erläutert. Abschnitt A.2 versteht sich als kurze Betriebsanleitung. Außerdem ist eine mit `javadoc` erzeugte Online-Klassenbeschreibung verfügbar.

A.1 Klassenbeschreibung

Gesteuert werden diese Agenten vom `CollectorSupervisor`, der den Information Layer beobachtet und Neuzugänge unter den Personen zum Anlass nimmt, die Agenten auf die Suche zu schicken. Abbildung A.1 zeigt das Klassendiagramm des Agentensystems.

Jede `Collector`-Instanz hat eine Assoziation zu einer Instanz der Klasse `Person`, in der die Ergebnisse des Suchlaufs verwaltet werden, bevor die Daten mit dem Information Layer synchronisiert werden. Die Klasse `Person` greift ihrerseits auf die Klassen `Project` und `Publication` zu. Für jede gefundene Publikation und jedes gefundene Projekt werden Instanzen dieser Klassen angelegt. Die Klasse `Publication` kennt folgende Variablen:

- Titel,
- Erscheinungsjahr,
- einen Vektor mit den Namen der Autoren,

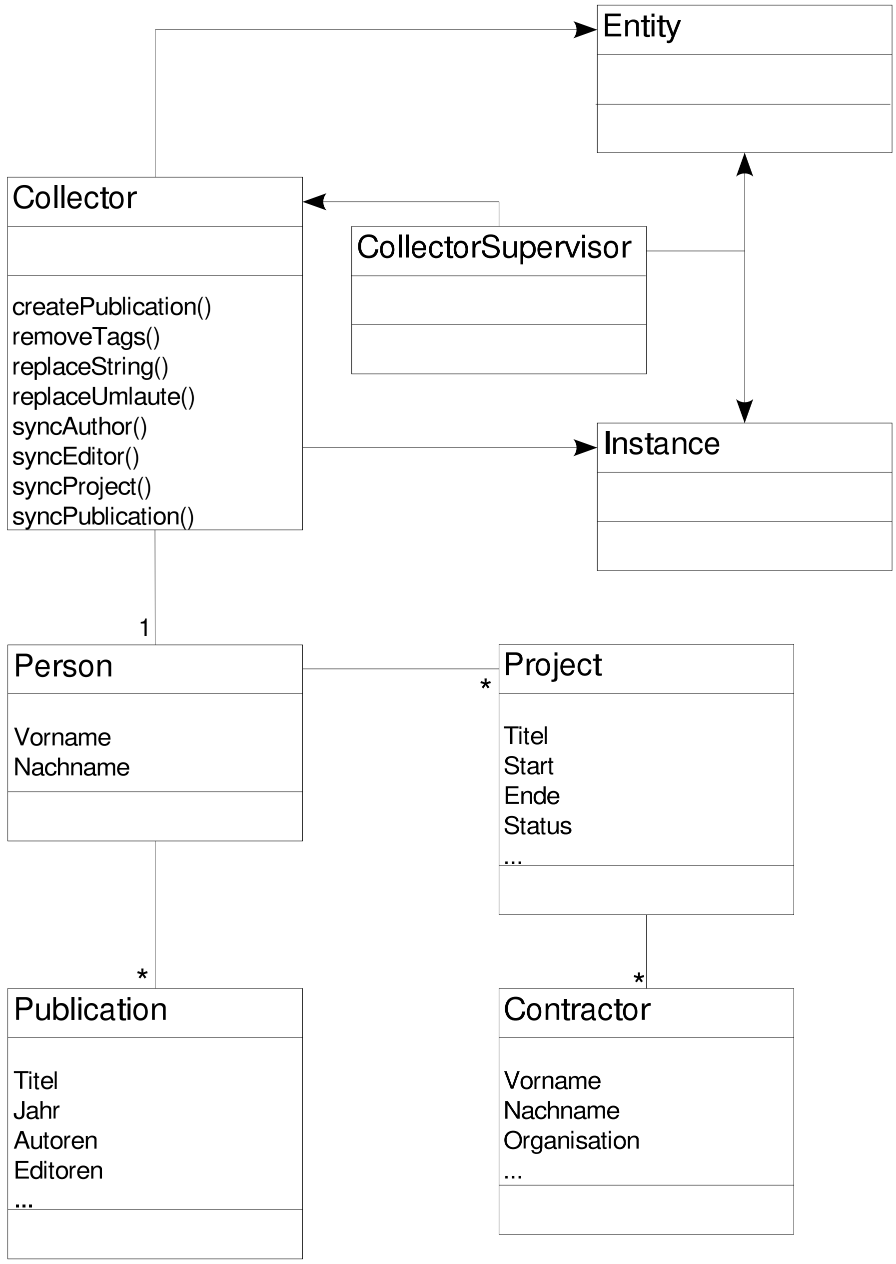


Abbildung A.1: Klassendiagramm.

- einen Vektor mit den Namen der Editoren, evtl.
- die Dokumentenklasse (je nach Agent).

Die Klasse `Project` besteht aus allgemeinen Projektdaten und Daten über die Projektpartner, die in einer eigenen Klasse (`Contractor`) verwaltet werden. Im einzelnen sind das:

- Projekttitel,
- Startdatum, Enddatum,
- ein Hauptvertragspartner (1:1-Relation) und
- einer oder mehrere Projektpartner (1:n-Relation).

Die Klasse `Contractor` kennt Vornamen, Nachnamen und Organisation des entsprechenden Partners.

Außerdem gibt es einige Hilfsklassen, die im Testbetrieb und zum Erzeugen der MOBAL-Syntax dienen.

`AgentsystemGUI` ist für den Testbetrieb entworfen und bietet eine graphische Oberfläche (Abbildung A.2). Hier kann eine einzelne Person eingegeben werden, für die dann nach Veröffentlichungen und Projekten gesucht wird. In einer Statusanzeige kann abgelesen werden, womit die Agenten gerade beschäfftigt sind. Mögliche Anzeigen sind:

1. »`connecting`«
2. »`collecting`«
3. »`finished`« und
4. »`Fehler`«.

Für die Überwachung des Status wurde die Klasse `Postman` eingeführt.

`Postman` ist eine JAVA-Interface-Klasse, über die die Agenten ihren Status an das GUI[1] versenden. Werden die Agenten ohne GUI gestartet, werden die Nachrichten auf dem Standard Output (in der Regel der Bildschirm) ausgegeben.

`MobalOut` transformiert die für das Lernen mit MOBAL relevanten Daten aus dem Information Layer in die entsprechende Syntax.

`HumanOut` ist die »lesbare« Variante von `MobalOut`. Dient zur Kontrolle.

`Exchange` regelte vor der Einführung des Client-Server-Betriebs den Austausch zwischen den Agenten und dem Information Layer. Die Klasse ist inzwischen außer Betrieb.

[1] *Graphical User Interface.*

Abbildung A.2: Die Oberfläche des `AgentsystemGUI`. Links oben befinden sich zwei Textfelder zur Namenseingabe, rechts von der Mitte Funktionsbuttons und auf der rechten Seite die Statusfelder der Agenten, in denen die beschriebenen Meldungen angezeigt werden. Die unteren beiden Textflächen dienten während der Testphase als Protokollanzeigen.

A.2 Betrieb des Agentensystems

Das Agentensystem ist darauf ausgerichtet, Informationen über Konferenzteilnehmer, die im Information Layer gespeichert sind zu suchen und gegebenenfalls entsprechend zu ergänzen. Diese Informationen beziehen sich auf Publikationen und Projektbeteiligungen einer Person. Im Normalfall werden die Agenten nicht explizit gestartet, da sich der `CollectorSupervisor` im laufenden Betrieb darum kümmert. Er startet die Agenten immer dann, wenn er einen neuen Konferenzteilnehmer im IL wahrnimmt, bzw. zu regelmäßigen Updates. Vorausgesetzt der IL-Service läuft, lautet der Aufruf:

```
java CollectorSupervisor -source socket <host> <port>
```

Zu Testzwecken wird jedoch ein Programm bereitgestellt, mit dem entweder eine Person durch Angabe des Vor- und Nachnamens frei gwählt werden kann – und die so in den IL aufgenommen wird –, oder das einfach auf alle Personen, die bereits im IL gespeichert und gleichzeitig Konferenzteilnehmer sind, eine Suche startet kann. Der Aufruf lautet:

```
java AgentsystemGUI
```

Ein einzelner Agent wird durch einen Aufruf gestartet, der den Vor- und den Nachnamen der zu suchenden Person als Argumente enthält. Es macht im Allgemeinen keinen Sinn macht, eine Suche auf einen Vornamen zu starten. Um eine Person eindeutig zu identifizieren ist neben dem vollständigen Namen sicherlich noch einiges mehr notwendig, in diesem Kontext kann man jedoch davon ausgehen, dass eine Person mit ihrem Vor- und Nachnamen eindeutig im Information Layer zu identifizieren ist.

Entsprechend der gewünschten Datenquelle können folgende Agenten aufrufen werden:

- DBLP-Bibliographie Trier:

```
java DBLPCollector "<Vorname>" "<Nachname>"
```

- UKA-Bibliographie-Sammlung Karlsruhe:

```
java UKACollector "<Vorname>" "<Nachname>"
```

- CORDIS Projekt-DB Luxemburg:

```
java CordisCollector "<Vorname>" "<Nachname>"
```

Allen Aufrufen gemein sind auch die Auswirkungen auf den Information Layer. Die gefundenen Daten werden mit dem IL synchronisiert, d. h. bestehende Daten werden ergänzt, nicht existierende neu angelegt.

Hierbei ist Folgendes zu beachten. Die Konferenzteilnehmer des IL besitzen eine Kennzeichnung, so dass die Agenten bei einem Update nur für diesen Personenkreis aktiv werden. Das hat den Vorteil, dass die Agenten wirklich nur nach den Personen suchen, die auch an der Konferenz teilnehmen.

Ruft man die *Stand-alone*-Agenten für eine Person auf, so wird diese Person ebenfalls in den IL aufgenommen, jedoch mit der Kennzeichnung, dass es sich nicht um einen Konferenzteilnehmer handelt. Diese Aufrufe sind dann interessant, wenn Querbezüge zwischen Teilnehmern über Dritte untersucht oder aufgedeckt werden sollen.

Die Lauf- bzw. Suchzeiten der Agenten sind natürlich abhängig vom allgemeinen Netzverkehr (Ergebnisse siehe Kapitel 5, Abschnitt 5.1 Seite 61). Die Experimente haben darüberhinaus jedoch gezeigt, dass die Verbindung vom LS VIII der Universität Dortmund zum CORDIS-Server in Luxemburg um einiges langsamer ist, als die Verbindung zur Bibliographiesammlung in Karlsruhe bzw. zur DBLP-Bibliographie in Trier. Außerdem finden die Agenten die Informationen in den Bibliographien auf der ersten Ebene, während die Suche nach Projekten einer Person das Absteigen in eine zweite Ebene erfordert und das zu mehr Traffic führt.

Anhang B

Datenquellen

In diesem Anhang werden die von den Agenten benutzten Informationsquellen detailliert beschrieben. Im Rahmen dieser Diplomarbeit wurden folgende Datenquellen exemplarisch nach Informationen durchsucht (Tabelle B.1 zeigt die URLs):

1. Eine Sammlung von Bibliographien der Universität Karlsruhe (*The Collection of Computer Science Bibliographies*).
2. Die Informatik-Bibliographie der Universität Trier (*ACM Sigmoid DBLP*).
3. Die CORDIS-Projektdatenbank der Europäischen Gemeinschaft in Luxemburg.

QUELLE	URL
DBLP	http://www.informatik.uni-tier.de/~ley/db/indicies/query.html
UKA	http://liinwww.ira.uka.de/bibliography/index.html
CORDIS	http://dbs.cordis.lu/EN_PROJl_search.html

Tabelle B.1: URLs der Datenquellen.

In der Informatik wird ein wesentlicher Teil der Fachinformationen über Medien wie Zeitschriften und Tagungsbände verbreitet. An Hochschulen oder großen Forschungseinrichtungen stellen Bibliotheken diese Publikationen teilweise lokal zur Verfügung. Auf der Ebene von Bänden bzw. Zeitschriftentiteln weisen die Bibliotheken ihre Bestände in Katalogen (OPACs) nach. In Zeitschriften, Tagungsbänden und Sammelbänden erschienene Einzelarbeiten werden in OPACs nicht aufgeführt. Bei der Suche nach Artikeln muss daher auf externe Bibliographien zurückgegriffen werden. In der Praxis wird von diesen Diensten vor allem aus Kostengründen selten Gebrauch gemacht [Ley, 1997].

B.1 Bibliographiesammlung in Karlsruhe

Die »Collection of Computer Science Bibliographies« an der Universität Karlsruhe ist eine Sammlung von ca. 1200 Bibliographien von unterschiedlichen Quellen für den Bereich Informatik. Jeden Monat werden diese Bibliographien aktualisiert, so dass sie immer den Stand der Quellen entsprechen. Die Sammlung beinhaltet ca. 940.000 Einträge (600MByte) im BibTeX-Format[1] – meistens Journalartikel, Konferenzproceedings und technische Reporte [Archilles, 1997].

Folgende Themenbereiche werden von den Bibliographien abgedeckt:

- Artificial Intelligence
- Compiler Technology, Programming Languages and Type Theory
- Database Research
- Distributed Systems, Networking and Telecommunication
- Computer Graphics and Vision
- Logic Programming
- (Computational) Mathematics
- Neural Networks
- Object-Oriented Programming and Systems
- Operating Systems
- Parallel Processing
- Software Engineering and Formal Methods
- Theory/Foundations of Computer Science
- Typesetting

Die einzelnen Bibliographien wurden unabhängig von verschiedenen Autoren zusammengestellt. Zudem ist BibTeX ein sehr weitgefasstes Bibliographie-Format ohne strenge Regeln. Das führt dazu, dass die Bibliographien mitunter unterschiedliche Formate haben – die einen enthalten kurze Zusammenfassungen oder Kommentare, andere enthalten Schlüsselwörter.

Das Hauptaugenmerk dieser Sammlung liegt auf Veröffentlichungen in Jornalen, technischen Reporten und Konferenzpapieren und nicht auf veröffentlichen Büchern. Die meisten Bibliographien in dieser Sammlung erheben keinen Anspruch auf Vollständigkeit und sind es auch nicht[2]. Durch

[1] Beispiel in Tabelle 4.5, Seite 54.
[2] Zur Zeit dieser Diplomarbeit waren nur Publikationen bis zum Jahr 1998 verfügbar.

die Zusammenfassung so vieler Bibliographien finden sich jedoch viele Veröffentlichungen aus dem Bereich Informatik – einige von ihnen doppelt oder öfter.

Die wachsende Zahl von Publikationen (Abbildung B.1) und die Integration von immer mehr Bibliographien läßt die Anzahl der Duplikate rapide zunehmen.

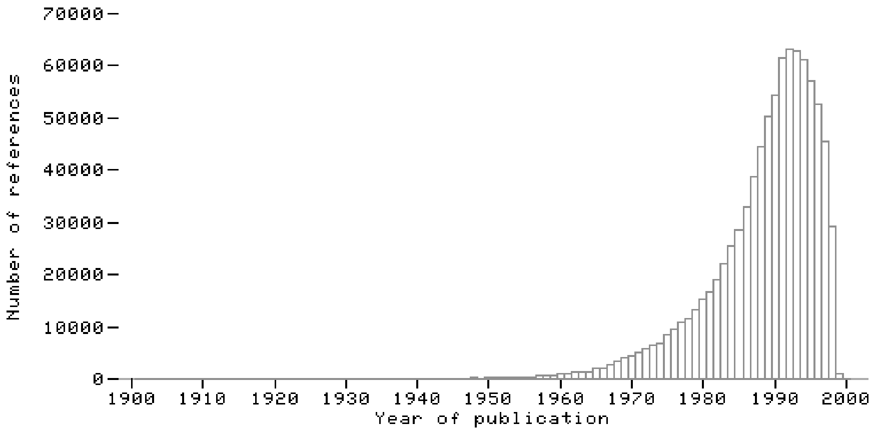


Abbildung B.1: Verteilung der Publikationsjahre ([Archilles, 1997]).

30 % aller Einträge wurden 1997 automatisch als Dublikate erkannt und entfernt . Hier könnten Methoden des Information Retrieval eingesetzt werden, um geringfügige Abweichungen im Titel zu entdecken und um so die verbleibenden Dublikate zu erwischen.

Die Bibliographiesammlung in Karlsruhe bietet zwei Abfragemöglichkeiten, eine Standardsuche (Abbildung B.2) und eine erweiterte Suche. In der erweiterten Suche kann gezielt auf bestimmten Feldern gesucht werden. Diese Felder sind weitesgehend identisch mit den bekannten BIBTEX-Feldern. So kann man z. B. gezielt nach Autoren, Titeln, Erscheinungsdaten, Journalen suchen.

Allerdings lässt sich das Feld Editor nicht abfragen, was diese Schnittstelle für den Agenten unbrauchbar macht. Grundsätzlich sind die Abfragemöglichkeiten hier jedoch viel komfortabler als in der Standardabfrage. Um alle Veröffentlichungen einer Person zu finden, bei der sie sowohl Autor als auch Editor (oder beides) ist, muss diese Person auf allen Feldern gesucht und das Ergebnis entsprechend gefiltert werden. Kommt die entsprechende Person wirklich als Autor oder Editor vor? Wenn der Personenname zufällig im Titel einer Veröffentlichung steht, ist es keine Veröffentlichung dieser Person[3].

Ein besonderer Vorteil dieser Datenquelle in Bezug auf Agenten ist die Möglichkeit, das Ergebnis der Anfrage im BIBTEX-Format geliefert zu bekommen. Innerhalb einer HTML-Seite befinden sich die BIBTEX-Angaben dann zwi-

[3]Es könnte sich dann vielleicht um eine Veröffentlichung *über* eine Person handeln, was natürlich unter gewissen Bedingungen auch interessant ist. In dieser Arbeit wird dieser Aspekt jedoch nicht berücksichtigt.

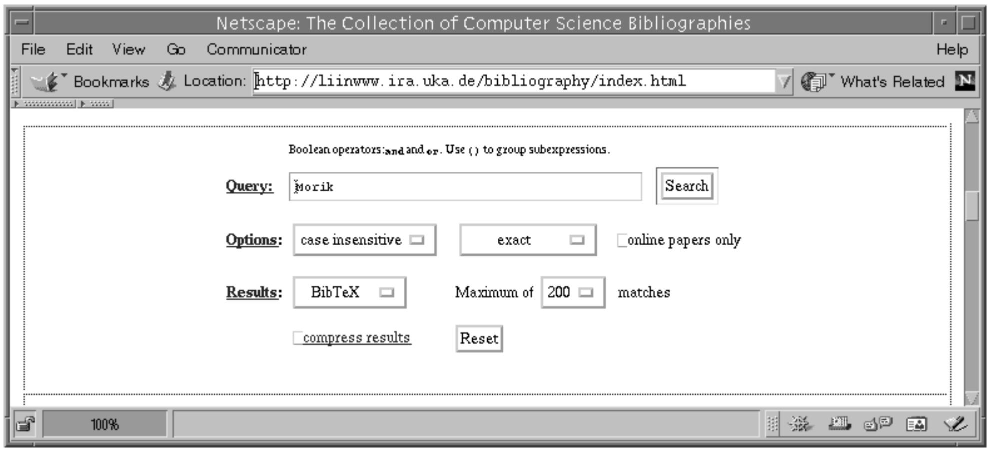


Abbildung B.2: Die Standardabfrage der »Collection of Computer Science Bibliographies« der Universität Karlsruhe.

schen `pre`-Tags, was natürlich die automatische Extraktion dieser Informationen aus der HTML-Seite erheblich erleichtert. Abbildung B.3 zeigt einen Auszug aus den Resultaten einer Anfrage.

B.2 Die Informatik-Bibliographie DBLP

An der Universität Trier wird sei Anfang 1994 ein Web-Server für Informatik-Fachinformationen betrieben. Bis Mitte Mai 1997 waren in der Bibliographie über 60000 Artikel aufgeführt, die aus den in Tabelle B.2 aufgeführten Zeitschriften stammen [Ley, 1997].

ACM Comp. Surveys	ACM TOCS	ACM TODS
ACM TOIS	ACM TOMACS	ACM TOPLAS
Acta Informatica	Algorithmica	CACM
Computer Languages	Data & Knowledge Eng.	Data Eng. Bulletin
Distr. Computing	ECCC	IEEE TKDE
IEEE TSE	Informatik Forsch. Ent.	Informatik Spektrum
Inf. Proc. Letters	Information Systems	J. Algorithms
JCSS	JIIS	J. Logic Programming
SIAM J. Computing	SIGMOD Record	Software Prac. & Exp.
TCS	Theory Comp. Sys. (MST)	VLDB Journal

Tabelle B.2: Einige von DBLP erschlossene Zeitschriften.

Abbildung B.4 zeigt die Abfrage-HTML-Seite, hinter der die Suchmaschine freeWAIS-sf [Pfeifer et al., 1995] steckt. Es lassen sich hier Autoren, Konferenzpapiere und Journale abfragen. An dieser Stelle sei Dr. Michael Ley ein Dank ausgesprochen, da er mir die offiziell noch nicht freigegebene Anfrage-

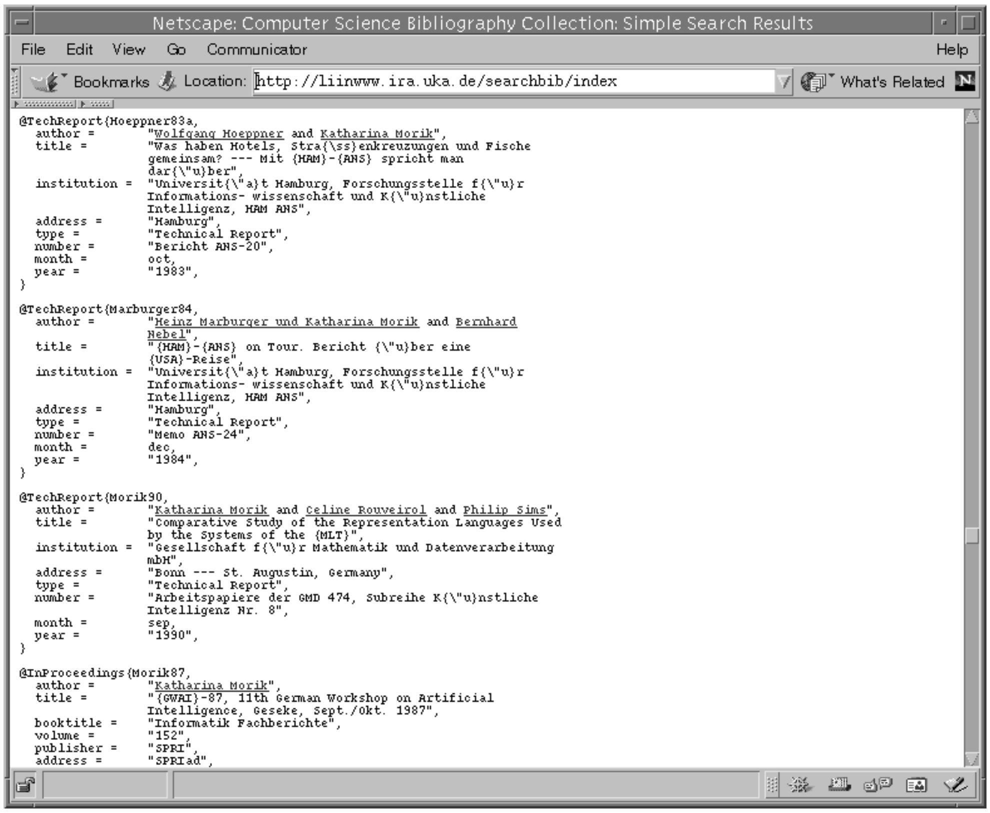


Abbildung B.3: Beispiel einer HTML-Seite, die die Ergebnisse einer Anfrage im BIBTEX-Format enthält (Auszug).

Seite mitgeteilt und zur Benutzung für meine Diplomarbeit freigegeben hat.

Der Kern der in DBLP bereitgestellten Informationen sind bibliographische Sätze, die intern in einem BIBTEX-ähnlichen Format gespeichert sind. Die Web-Seiten sind materialisierte Datenbank-Sichten, die um Zusatzinformationen ergänzt, geeignet formatiert und durch Hyperlinks – in der Regel zu den Seiten der aufgeführten Koautoren – miteinander verknüpft sind. Diese Seiten werden als Treffer zurückgeliefert. Sie sind nach Nachnamen in entsprechenden Verzeichnissen gespeichert. Im Beispiel (Abbildung B.5) heißt die Seite `Morik:Katharina.html` und liegt im Verzeichnis `m`. Die Informationen auf der Seite sind innerhalb einer Tabelle plaziert, was für den Agenten von Vorteil ist, da er die Tabellenstruktur ausnutzen kann.

Wird nur mit dem Nachnamen einer Person gesucht, und gibt es mehrer Personen mit diesem Nachnamen, bekommt man eine Liste von Personen, die der Datenquelle bekannt sind und muss sich dann für die entsprechende Person entscheiden.

Die DBLP liefert zu den meisten Publikationen nur das absolute Minimum an Information: die Namen der Autoren, den Titel und den Publikationskontext. Es ist nicht möglich, nach Editoren zu suchen.

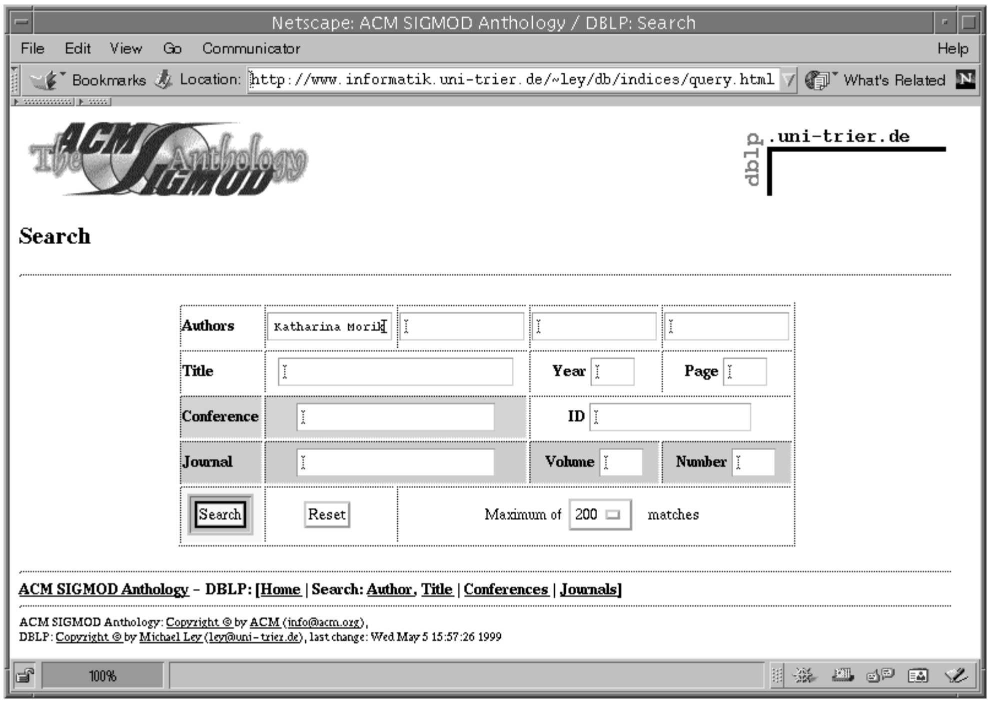


Abbildung B.4: Die Abfragemaske der DBLP in Trier.

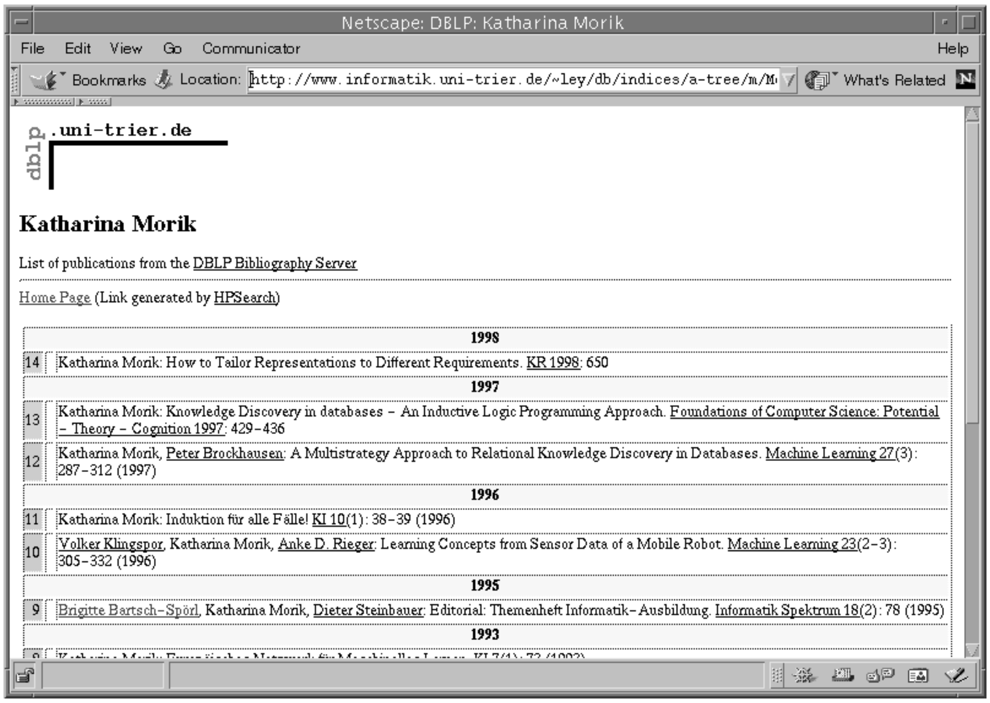


Abbildung B.5: Die zurückgelieferte HTML-Seite. Diese Seiten werden nicht dynamisch erzeugt, sondern liegen bereits vorbereitet auf dem Server.

B.3 Die CORDIS-Projektdatenbank

Die CORDIS-Projektdatenbank ist im WWW bisher einzigartig. Nicht das sie außergewöhnliche Dienstleistungen bietet, aber es ist die einzige Datenquelle für Projekte[4]. Nicht einmal die Deutsche Forschungsgesellschaft (DFG) bietet Projektdatenbanken an.

CORDIS ist ein europäischer Informationsdienst der im Bereich Forschung und Entwicklung (FuE) der Europäischen Gemeinschaft in Luxemburg lokalisiert ist. Dieser Dienst bietet den Nutzern in dieser Eigenschaft Informationen zu einem breiten Spektrum von Aktivitäten, die auf europäischer Ebene in diesem Bereich unternommen werden. Dem Nutzer werden eine Vielzahl unterschiedlicher Dienste zur Verfügung gestellt, die ihn über Aktivitäten in der Europäischen Union stets auf dem Laufenden halten. Unter anderen Diensten gehören die recherchierbaren Datenbanken zu dem, was CORDIS anbietet – darunter:

- Nachrichten: Täglich neue Informationen zu wichtigen Aspekten von Forschung und Entwicklung, wie z.B. Aufrufe zu Vorschlägen, Ausschreibungen und Veranstaltungen.
- Kontakte: Auflistung offizieller Kontaktstellen, die bei Fragen EU-geförderter Forschung und Entwicklung informieren, beraten und unterstützen.
- Programme: Detaillierte Informationen zu allen EU-geförderten Forschungsprogrammen und FuE-bezogenen EU-Programmen.
- Veröffentlichungen: Studien, Berichte und wissenschaftliche Beiträge zur EU-Forschung.
- Projekte: Details zu einzelnen, im Rahmen der Programme ausgeführten Forschungsprojekten.
- Akronyme: Kompaktes Verzeichnis von Begriffen der Forschung und Entwicklung in der EU.
- Partner: Unterstützung bei der Suche nach geeigneten Partnern für die Zusammenarbeit bzw. die Teilnahme an EU-Projekten.
- Ergebnisse: Unterstützung beim Ausfindigmachen verwertbarer Forschungsergebnisse und Prototypen für Innovation sowie Förderung ihrer Nutzung.
- Dokumentenbibliothek: Offizielle Dokumente, wie z.B. Aufrufe zu Vorschlägen, Arbeitsprogramme, Informationspakete, zu erbringende Leistungen und Teilnahmeleitlinien.

[4] Allerdings sind hier nur ESPRIT-Projekte (*European Strategic Programme for Research and development in Information Technologies*) zu finden.

CORDIS bietet verschiedene Zugriffsmöglicheiten von denen das WWW von den Agenten genutzt wird. Abbildungen B.6 und B.7 zeigen die Abfrageseite und die Liste der gefundenen Projekte.

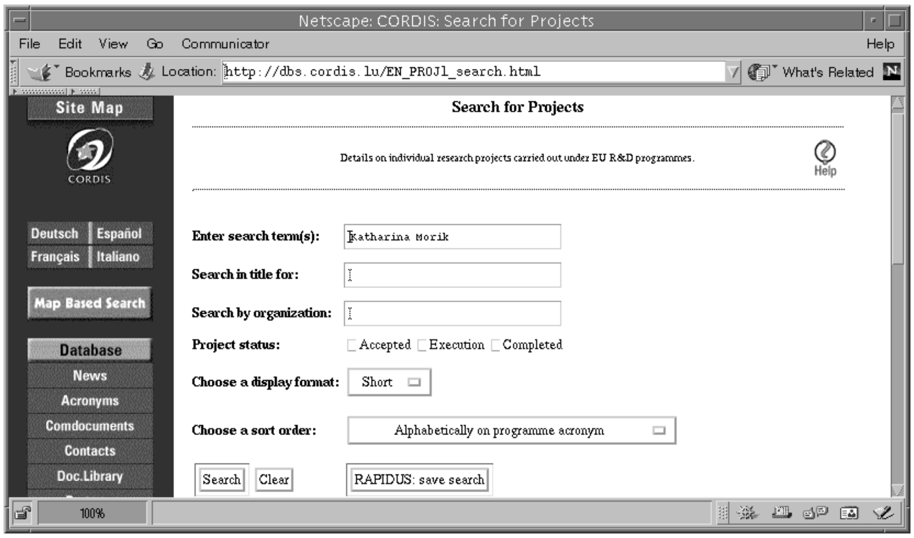


Abbildung B.6: Abfrage-Seite der Projekt-Datenbank von CORDIS.

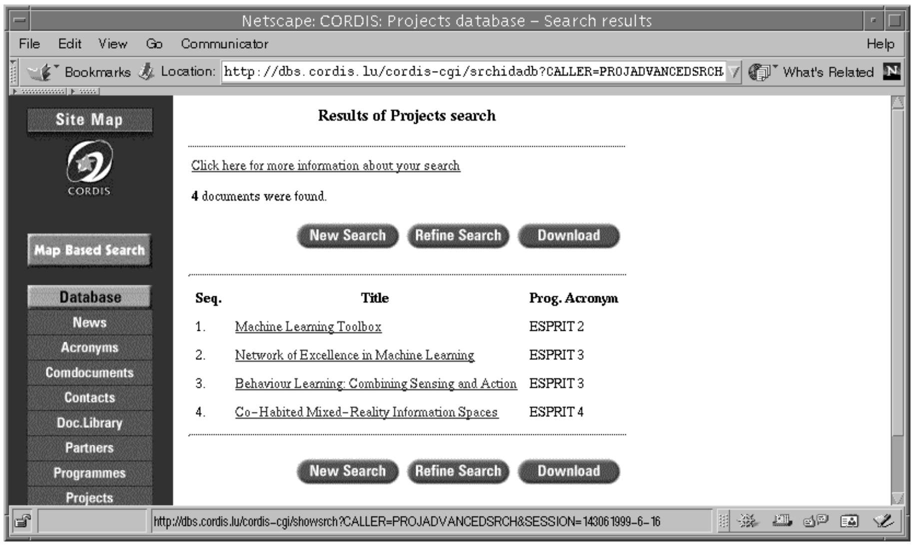


Abbildung B.7: Gibt es Projekte werden diese auf einer HTML-Seite aufgelistet.

Folgt man den Links, erhält man detaillierte Angaben über das entsprechende Projekt (Abbildungen B.8 - B.10).

Abbildung B.10 verdeutlicht auch direkt ein Problem – offensichtlich sind die Daten nicht korrekt! Der Lehrstuhl in Dortmund hat nicht die Nummer fünf, sondern acht!

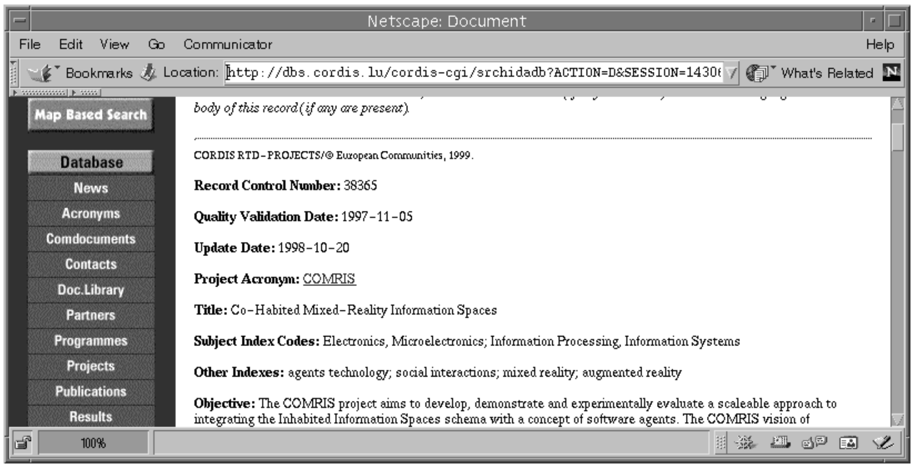


Abbildung B.8: Einzelheiten über das Projekt (Auszug).

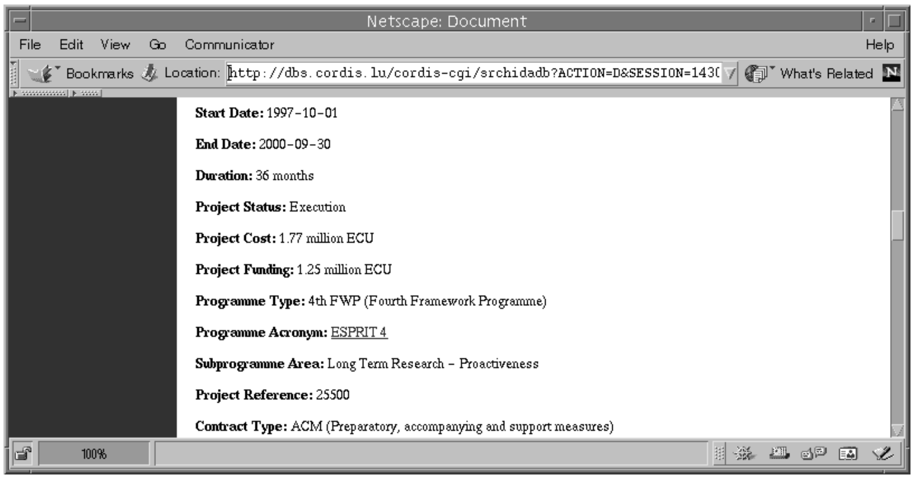


Abbildung B.9: Weitere Projektdaten (Auszug).

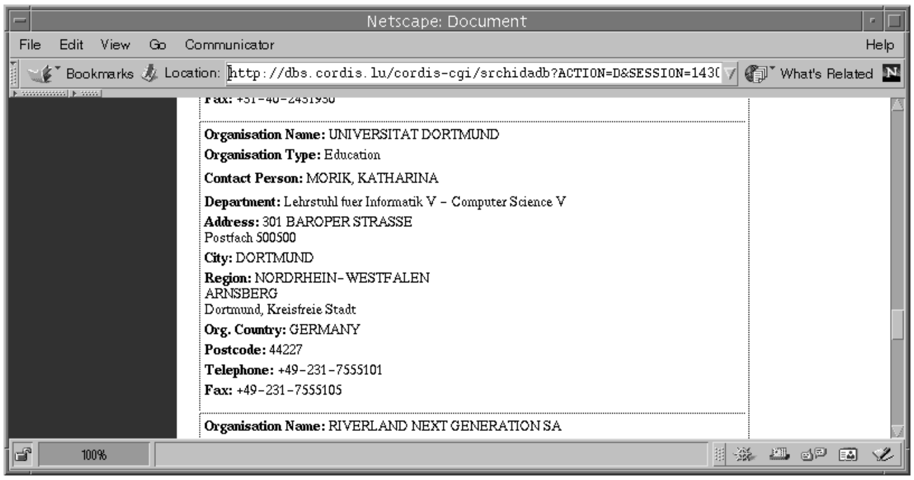


Abbildung B.10: Die Daten über die Projektpartner (Auszug), hier: Fehlerhafte Daten eines Vertragspartners.

Anhang C

MOBAL-Parameter

C.1 RDT-Parameter

C.1.1 Systemwerte von RDT

MOBAL berechnet fünf Systemwerte, die zum Bestimmen des Akzeptanzkriteriums benutzt werden können:

`pos` Die Anzahl positiver Instanzen, die eine Regel abdeckt.

`neg` Anzahl negativer Instanzen.

`pred` Anzahl der vorhergesagten Instanzen einer Regel.

`total` Alle Instanzen, die von einer Regel abgedeckt werden (`pos + neg + pred`).

`unc` Von *uncovered* – Instanzen (`pos` oder `neg`), die von der Regel nicht abgedeckt werden.

C.1.2 RDT Parameter

Abhängig von nachstehenden Parametern wird das Akzeptanzkriterium von RDT gebildet. Dies ist eine bequeme Art, da man die Ausdrücke für das *confirmation*- und das *pruning*-Kriterium nicht selbst bilden muss.

`min_no_of_pos_examples` Dieser Wert legt die Mindestmenge von positiven Beispielen (`pos`) fest, die von einer Regeln abgedeckt werden müssen, bevor RDT diese Regel akzeptiert.

`inductive_leap_percentage` Gibt die Größe des *induktiven Sprungs* in Prozent an. Je größer diese Zahl ist, umso mehr Regeln, deren Anzahl positiver Instanzen im Verglich zu allen möglichen Instanzen klein ist, werden von RDT akzeptiert.

`closed_world_assumption` Wird der Wert dieses Parameters auf `yes` gesetzt, ist alles, was nicht wahr ist, falsch. D. h. außer den negativen Beispielen (`neg`) werden auch alle Fakten, die eine Regel ableitet (`pred`) und für die der Wahrheitswert nicht bekannt ist, als falsch betrachtet.

`max_no_of_exceptions` Die maximale Anzahl von Ausnahmen (`neg`) die erlaubt sind, damit eine Regel noch akzeptiert wird.

`max_percentage_of_exceptions` Der maximale Anteil (in %) negativer Beispiele (`neg`) im Verhältnis zu den positiven (`pos`) und den vorhergesagten (`pred`).

`other_criteria` Hier können Ausdrücke eingegeben werden, die das Lernergebnis beeinflussen. Erlaubt sind Ausdrücke die Mengenbezeichnungen für `pos`, `neg`, `unc`, `pred`, und `total` enthalten.

C.2 STT-Parameter

Mit diesen Parametern kann man die Sortenbildung von MOBAL beeinflussen.

`sort_processing_mode` Dieser Parameter bestimmt, ob und wann die Sorten berechnet werden. Es gibt drei Modi:

1. `off`: Es werden keine Sorten berechnet.
2. `update`: Die Sorten werden kontinuierlich auf dem neusten Stand gehalten, jede Änderung wird direkt berechnet (das kann lange dauern!).
3. `delay`: Die Sorten werden per explizitem Befehl neu berechnet.

`generate_intersection_sorts` Mit diesem Parameter kann man die Schnittmengenbildung an- und ausschalten. Bei großen Wissensbasen kann der Prozess der Schnittmengenbildung sehr zeitintensiv sein.

`process_argument_sorts` Auch hier gibt es drei Möglichkeiten:

1. `always`: Alle Argumente werden als Sorten betrachtet und in die Taxonomie einbezogen.
2. `no_user`: Nur die Sorten, die nicht vom Anwender bestimmt wurden werden einbezogen.
3. `off`: Keine Sorten werden beachtet.

`process_user_sorts` Auch die vom Anwender bestimmten Sorten werden in die Berechnung einbezogen.

Anhang D

Abkürzungsverzeichnis

AK	Akzeptanzkriterien
B	Hintergrundwissen
$L_1, \ldots, L_n$	Literale
e	Beispiel aus E
E	Beispielmenge
E^+	Positive Beispielmenge
E^-	Negative Beispielmenge
h	Hypothese aus H
H	Hypothesenraum
I	Interpretation
LB	Sprache des Hintergrundwissens
LE	Beispielsprache
LH	Hypothesensprache
$\mathcal{M}(X)$	Modell von X
$\mathcal{M}^+(X)$	Minimales Modell von X
$p(t_1, \ldots, t_n)$	Allgemeine Syntax eines Faktums in MOBAL
$t_1, \ldots, t_n$	Terme eines Faktums
T	Theorie

Literaturverzeichnis

[Archilles, 1997] Archilles, A.-C. (1997). The Collection of Computer Science Bibliographies. URL: http://liinwww.ira.uka.de/bibliography/index.html.

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

[Bandler und Grinder, 1996] Bandler, R. und Grinder, J. (1996). *Patterns: Muster der hypnotischen Techniken Milton H. Ericksons*. Junfermann, Paderborn, Germany.

[Basu et al., 1999] Basu, C., Hirsh, H., Cohen, W. W., und Nevill-Manning, C. (1999). Recommending Papers by Mining the Web.

[Beale und Wood, 1994] Beale, R. und Wood, A. (1994). Agent-Based Interaction. In *People and Computers IX: Proceedings of HCI'94*, Seiten 239–245, Glasgow, UK.

[Belkin und Coft, 1992] Belkin, N. und Coft, B. W. (1992). Information Filtering and Information Retrieval: Two Sides of the Same Coin? *CACM*, 35(12):29–38.

[Bradshaw, 1997] Bradshaw, J. M. (1997). *Software Agents*. AAAI Press, Menlo Park, Calif.

[Bray et al., 1998] Bray, T., Paoli, J., und Sperberg-McQueen, C. M., Hrsg. (1998). *Extensible Markup Language (XML) 1.0*. World Wide Web Consortium. http://www.w3.org/TR/1998/REC-xml-19980210.

[Brockhausen und Morik, 1997] Brockhausen, P. und Morik, K. (1997). Wissensentdeckung in relationalen Datenbanken: Eine Herausforderung für das maschinelle Lernen. In Nakhaeizadeh, G., Hrsg., *Data Mining, theoretische Aspekte und Anwendungen*, Wirtschaftsinformatik, Seiten 193–211. Physica Verlag.

[Davies und Weeks, 1995] Davies, N. J. und Weeks, R. (1995). Jasper: Communicating Information Agents. In *Proceedings of the 4th International Conference on the World Wide Web*, Boston, USA.

[Etzioni und Weld, 1994] Etzioni, O. und Weld, D. (1994). A Softbot-Based Interface to the Internet. *Communications of the ACM (CACM)*, 37(7):72–76.

[Ferber, 1998] Ferber, R. (1998). *Philosophische Grundbegriffe: Eine Einführung*. Beck, München.

[Flanagan, 1998] Flanagan, D. (1998). *Java in a Nutshell, deutsche Ausgabe*. O'Reilly.

[Foner, 1997] Foner, L. N. (1997). Yenta: A Multi-Agent, Referral Based Matchmaking System. Tagungsbeitrag, The First International Conference on Autonomous Agents (Agents 97), Marina del Rey, CA, USA.

[Franklin und Graesser, 1997] Franklin, S. und Graesser, A. (1997). Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In Müller, J. P., Wooldridge, M. J., und Jennings, N. R., Hrsg., *Intelligent Agents III*, Seiten 21 – 36, Berlin, Heidelberg, New York. Springer.

[Freitag, 1998] Freitag, D. (1998). Information Extraction from HTML: Application of a General Machine Learning Approach. In *Proceedings of the Fifteenth Conference on Artificial Intelligence (AAAI)*, Seiten 517–523, Madison, Wisconsin, USA. AAAI Press.

[Haustein, 1999] Haustein, S. (1999). The COMRIS Information Layer. Software Documentation.

[Kietz und Wrobel, 1992] Kietz, J.-U. und Wrobel, S. (1992). Controlling the Complexity of Learning in Logic through Syntactic and Task–Oriented Models. In Muggleton, S., Hrsg., *Inductive Logic Programming.*, Nummer 38 in The A.P.I.C. Series, Kapitel 16, Seiten 335–360. Academic Press, London [u.a.].

[Kinny und Georgeff, 1997] Kinny, D. und Georgeff, M. (1997). Modelling and Design of Multi-Agent Systems. In Müller, J. P., Wooldridge, M. J., und Jennings, N. R., Hrsg., *Intelligent Agents III*, Seiten 1–20, Berlin, Heidelberg, New York. Springer.

[Klingspor, 1998] Klingspor, V. (1998). *Reaktives Planen mit gelernten Begriffen*. Dissertation, Univ. Dortmund.

[Klusch und Benn, 1998] Klusch, M. und Benn, W. (1998). Intelligente Informationsagenten im Internet. *Künstliche Intelligenz*, 3:8–17.

[Koenemann und Thomas, 1998] Koenemann, J. und Thomas, C. G. (1998). Agent-Supported Information Broking. *Künstliche Intelligenz*, 3:30–37.

[Lamport, 1995] Lamport, L. (1995). *Das LaTeX-Handbuch*. Addison-Wesley, Bonn.

[Lavrac et al., 1999] Lavrac, N., Flach, P., und Zupan, B. (1999). Rule Evaluation Measures: A Unifying View. In Dzeroski, S. und Flach, P., Hrsg., *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, Lecture Notes in Artificial Intelligence 1634, Seiten 174–185, Bled, Slovenia. Springer.

[Ley, 1997] Ley, M. (1997). Die Trierer Informatik-Bibliographie DBLP. Als PS-File an der Uni Trier erhältlich.

[Liebermann, 1998] Liebermann, H. (1998). Beyond Information Retrieval: Information Agents at the MIT Media Lab. *Künstliche Intelligenz*, 3:17–23.

[Metz-Göckel, 1997] Metz-Göckel, H. (1997). Allgemeine Psychologie 1: Kognitionspsychologie. Universität Dortmund, FB 14. Skript zur gleichnamigen Vorlesung.

[Michalski, 1986] Michalski, R. (1986). Understanding the Nature of Learning. In Michalski, Carbonell, und Mitchell, Hrsg., *Machine Learning - An Artificial Intelligence Approach*. Morgan Kaufmann, Los Altos, California.

[Minsky, 1985] Minsky, M. (1985). *Society of Mind*. Touchstone Press.

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

[Morik, 1989] Morik, K. (1989). Sloppy Modeling. In Morik, K., Hrsg., *Knowledge Representation and Organization in Machine Learning*, Seiten 107–134. Springer Verlag, Berlin, New York.

[Morik, 1998] Morik, K. (1998). Programmierung I: JAVA. Skript zur gleichnamigen Vorlesung, 1.Auflage.

[Morik, 1999] Morik, K. (1999). Maschinelles Lernen. Skript zur gleichnamigen Vorlesung, 3.Auflage.

[Morik et al., 1993] Morik, K., Wrobel, S., Kietz, J.-U., und Emde, W. (1993). *Knowledge Acquisition and Machine Learning - Theory, Methods, and Applications*. Academic Press, London.

[Moukas, 1996] Moukas, A. G. (1996). Amalthaea: Information Discovery and Filtering using a Multi-Agent Evolving Ecosystem. In *Proceedings of the Conference on Practical Application of Intelligent Agents and Multi-Agent Technology PAAM96*, London.

[Naisbitt, 1985] Naisbitt, J. (1985). *Megatrends*. Heyne, München.

[Nwana, 1996] Nwana, H. S. (1996). Software Agents: An Overview. *Knowledge Engineering Review*, 11(3):205–244. Cambridge University Press.

[Oestereich, 1997] Oestereich, B. (1997). *Objektorientierte Softwareentwicklung mit der Unified Modeling Language*. Oldenbourg.

[Pfeifer et al., 1995] Pfeifer, U., Fuhr, N., und T., H. (1995). Searching Structured Documents with the Enhanced Retrieval Functionality of freeWAIS-sf and SFgate. *Computer Networks and ISDN Systems*, 27(6):1027–1036.

[Redlich, 1996] Redlich, J. P. (1996). *Corba 2.0: Praktische Einführung für C++ und Java*. Addison-Wesley.

[Salton, 1989] Salton, G. (1989). *Automatic Text Processing. The Transformation, Analysis and Retrieval of Information by Computer*. Addison Wesley.

[Scheller et al., 1994] Scheller, M., Boden, K.-P., Geenen, A., und Kamoermann, J. (1994). *Internet: Werkzeuge und Dienste*. Springer.

[Sommer et al., 1996] Sommer, E., Emde, W., Kietz, J.-U., und Wrobel, S. (1996). *Mobal 4.1b9 User Guide*. GMD – German National Research Center for Information Technology, AI Research Division (I3.KI), St. Augustin, Germany.

[Weiß und Sen, 1996] Weiß, G. und Sen, S. (1996). *Adaption and Learning in Multi-Agent Systems*. Springer, Berlin Heidelberg NewYork.

[Wooldridge und Jennings, 1995] Wooldridge, M. J. und Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152.