Ausarbeitung zum Artikel

Learning to Construct Knowledge Bases from the World Wide Web

von **Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, Seán Slattery**

http://www.biostat.wisc.edu/~craven/papers/aij00.ps
(2000)

Seminar Informationsextraktion WS 2003

Oliver Rohr

INHALTVERZEICHNIS

1. Einführung

Das World Wide Web ist mittlerweile eine der grössten Informationsquellen der Welt. Man ist folglich daran interessiert die im WWW enthaltenen Daten so darzustellen, daß sie von Computerprogrammen ausgewertet werden können, die diese dann in unzähligen Anwendungsgebiete verarbeiten können.

Im Gegensatz zu *Line Eikvils -Information Extraction from the World Wide Web (99)* werden im folgenden nur Informationen aus ‚frei zugänglichen' Hypertextseiten ausgewertet, d.h. es gibt keine Zugriffe auf das Hidden Web (die in Eikvils Artikel ja eine primäre Rolle spielen). Von zentraler Bedeutung ist, daß das vorgestellte System im Gegensatz zu Wrappern (wie bei Eikvils und auch bei den in *Gunter Grieser, Klaus P.Jantke, Steffen Lange and Bernd Thomas "A Unifying Approach to HTML Wrapper Representation and Learning"* vorgestellten Systemen) unterschiedliche Seiten einer Domäne und deren Umgebung zur Informationsgewinnung heranziehen kann. Diese Strategie gleicht den vorherrschenden Mangel an sematischen vordefinierten Strukturen innerhalb des WWWs wie sie u.a. in *Michael Klein, Dieter Fensel, Frank van Harmelen and Ian Horrocks (2000) "The Relation between Ontologies and Schema- languages: Translating OIL- specifications in XML-Schema"* beschrieben werden, zwar nicht aus, jedoch ist durch die Betrachtung unterschiedlicher Informationsquellen eine zufriedenstellende Extraktion möglich.

Die hier vorgstellten Ansätzte sind grössenteils neu entwickelte / modifizierte Methoden der Informationsextraktion. Lediglich in Kapitel 7 wird vorwiegend auf Methoden herkömmlicherer Textextraktion zurückgegriffen, die den in *Ralph Grishman (1997) "Information Extraction: Techniques and Challenges"* beschriebenen Methoden ähneln, wobei allerdings die Berücksichtigung natürlichsprachlicher Zusammenhänge (Grammatik, Wortarten) - wie in mehreren anderen Projekten der Informationsextraktion gezeigt (z.B. Grisham, s.o. oder *Roman Yangarber, Ralph Grishman, Pasi Tapanainen, Silja Huttunen (2000) "Unsupervised Discovery of Scenario-Level Patterns for Information Extraction"*) - ausbleibt.

Im Folgenden wird ein Prototyp behandelt, das WEB→KB System, der von den Autoren entwickelt worden ist und welcher die allgemeine Aufgabe hat, die im WWW enthaltenden Informationen in eine rechnerverständliche Wissensbasis umzuwandeln.

Es werden hierzu schwerpunktmäßig Techniken des maschinellen Lernens angewandt. Im Unterschied zum Ansatz in *Tobias Scheffer, Christian Decomain, Stefan Wrobel "Active Hidden Markov Models for Information Extraction "*, der mit ML Hidden Markov Models lernt, indem auf spärlich gelabelten Seiten trainiert und der Benutzer in das Training einbezogen werden kann, steht hier eine vollständig gelabelte Trainingsmenge zur Verfügung. Das System kann durchgehend selbständig, ohne Interaktion mit dem Benutzer, trainieren.

Wie kann nun die zuvor erwähnte Wissensbasis konstruiert werden? Die vorgestellte These zur Lösung dieses Problems besteht aus 2 Schritten:

1. Es wird Maschinelles Lernen (ML) angewendet, um Erkennungsfunktionen zur Informationextraktion für jeden der gewünschten Wissensarten zu finden.
2. Diese Methoden werden direkt auf den Hypertext angewendet, um symbolische, probabilistische Statements zu extrahieren.

Die obige These wird untersucht und der erwähnte Prototyp wird nun genauer beschrieben. Anschliessend werden verschiedene Lernstrategien vorgestellt und bewertet, die zur Lösung des obigen Problems angewandt werden können.

2. Überblick über das Web→KB System

Das WEB→KB System wird zuerst darauf trainiert die Informationen der gewünschten Klassen zu extrahieren. Anschliessend erlaubt man dem System eine gegebene Wissensbasis durch Information, die aus neuen Webseiten extrahiert wurden, selbstständig zu erweitern.

Das Training erfordert zwei Eingaben: Zum ersten eine Ontologie, die das Vokalbular der Wissensbasis definiert. Diese Ontologie besteht aus den gewünschten Klassen und Relationen zwischen diesen Klassen. Sie stellt so eine initiale Wissensbasis dar. Zum zweiten ist eine Trainingsmenge erforderlich, die einige Instanzen der Klassen und Relationen aus der Ontologie enthält.

Der Prototyp arbeitet mit einer Ontologie, die Informatikfachbereichsseiten beschreibt.
Als Trainingsmenge wurden ca. 8000 Seiten (Instanzen von Klassen) und ca. 1400 Webseiten-Paare (Instanzen von Relationen) angeboten, die gemäß der Ontologie manuell ausgezeichnet wurden. Dem System wurde dann erlaubt neue Instanzen aus weiteren 4127 Informatikfachbereichsseiten und 10945 zugehörigen Hyperlinks zu bilden.

Das System lernt beim Training generelle Erkennungsfunktionen zur Extraktion neuer Instanzen aus dem Netz. Der vorgestellte Prototyp startet mit einer gegebenen URL zu einer Webseite und durchläuft dann die Links auf dieser Seite mittels Breitensuche. Jede so gefundene Seite ist potentieller Kandidat für eine Klasseninstanz. Wird nun festgestellt, das eine Seite tatsächlich einer Klasse zugeordnet werden kann, so wird eine Entität, die diese Seite repräsentiert der Wissensbasis hinzugefügt, anschliessend werden gegebenenfalls Relationen instanziert, was von der Seite und der Sruktur der Umgebung der Seite abhängt (zu den Details später mehr). Kann eine Seite keiner Klasse zugeordnet werden, so wird die Rekursion abgebrochen und die Links auf dieser Seite werden nicht weiterverfolgt.

Und so sieht die Oberfläche des Prototypen in Aktion aus:

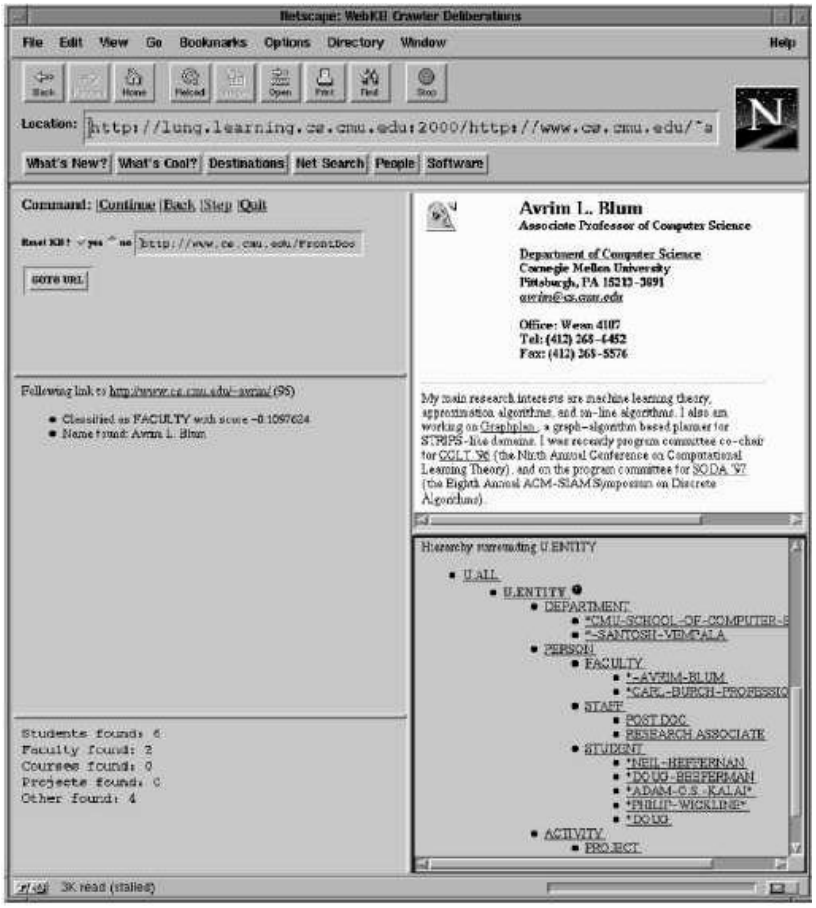

Abb.1:
Das Webinterface des vorgestellten Prototyps. Die fünf Frames haben folgende Bedeutungen:
oben rechts: die aktuell untersuchte Seite
unten rechts: eine hierarchieche, grafische Darstellung der aktuellen Wissensbasis
oben links: Interaktion mit dem System
mitte links: Instanzen, die in der aktuellen Seite gefunden wurden.
unten links: Übersicht über Instanzen, die in der aktuellen Session gefunden wurden

3. Problemspezifikation

Zusammenfassend sind wir nun formal an folgendem Problem interessiert:

Gegeben:

- eine Wissensbasis, die aus einer Ontologie der Klassen und Relationen von Interesse und optional aus Instanzen einiger dieser Klassen und Relationen besteht
- Trainingsbeispiele aus dem Web, die Instanzen dieser Klassen und Relationen beschreiben

Gesucht:

- allgemeine Erkennungsfunktionen, die fähig sind weitere Instanzen dieser Klassen und Relationen durch Traversierung des restlichen Webs zu extrahieren

Es sind einige Annahmen über die Wissensarten sowie über die Art und Weise wie Informationen im Web dargestellt werden nötig:

- Instanzen von (Ontologie-)Klassen werden durch Hypertextsegmente dargestellt. Ein (Hypertext-)Segment besteht entweder aus einer Webseite, aus einer fortlaufenden Zeichenkette in einer Webseite oder aus einem gerichtetem Graphen mehrerer über Links zusammenhängender Webseiten.
- Instanzen von Relationen werden wie folgt dargestellt: Sei R(A,B) eine Instanz einer Relation R mit Segmenten A und B. R(A,B) kann nun auf drei Arten dargestellt werden:

- als ungerichteter Weg aus Hyperlinks und Seiten, die Segment A mit Segment B verbinden
- Segment A ist ein Textabschnitt, der Segment B enthält
- Segment A wird durch gelernte Methoden zu Segment B in Relation gesetzt

In den behandelten Artikel wird weiterhin vorrausgesetzt, daß

- Instanzen von Klassen durch eine einzelne Webseite dargestellt werden (z.B. Instanz von person durch eine Homepage). Beschreiben mehrere Seiten eine Instanz, so wird nur die primäre (bedeutentste) Seite berücksichtigt.
- jede Instanz einer Klasse durch genau ein Hypertextsegment beschrieben wird
- jede Relation binär ist

Aus dieser Problemspezifikation ergeben sich nun unmittelbar die noch offenen Aufgaben:

a) Das Erkennen von Klasseninstanzen durch Klassifizierung von Hypertextkörpern (Kapitel 5)
b) Das Erkennen von Relationsinstanzen durch Klassifizierung von Hyperlinkketten (Kapitel 6)
c) Das Erkennen von Klassen- und Relationsinstanzen durch Extraktion kleiner Textsegmente aus Webseiten (Kapitel 7)

4. Experimentelle Testumgebung

Wie in Abb.1 gezeigt wird eine Ontologie verwendet, die Inhalte von Informatikfachbereichsseiten beschreibt. Diese beinhaltet Klassen wie **department**, **faculty**, **staff**, **student**, **research_project** und **course**. Seiten, die diesen Klassen nicht zugeordnet werden können werden der Klasse **other** zugeordnet.

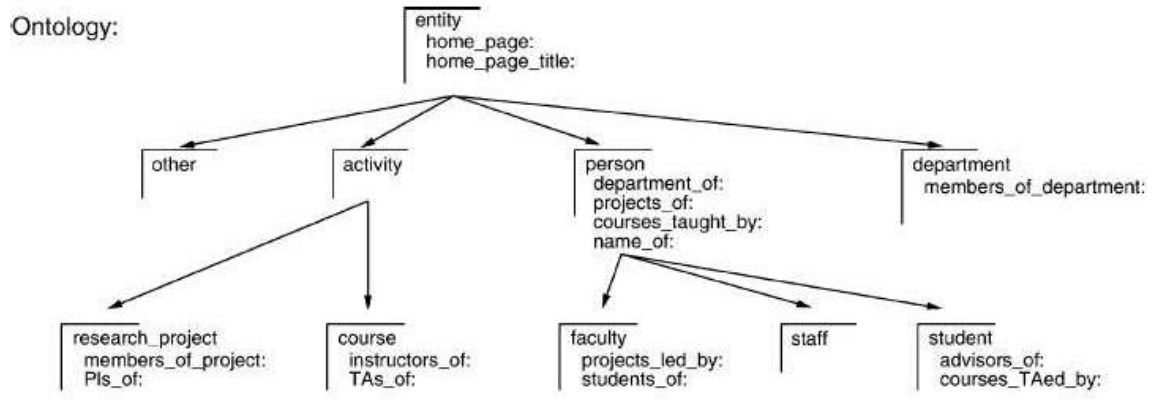

Abb.1: Die vom Prototyp verwendete Ontologie

5. Erkennen von Klasseninstanzen

In diesem Abschnitt werden Methoden vorgstellt und bewertet, die Instanzen von Klassen, die durch einzelne Webseiten repräsentiert werden, extrahieren können.

5.1. Statistische Text Klassifikation

Diese Methode ähnelt anderen Methoden der Textklassifikation, die sogennante *bag of words* oder *unigram* Darstellungen verwenden. Jedoch wird diese Methode so angepasst, daß sie vorhandene Redundanzen in Hypertextdokumente ausnutzt.
Speziell werden hier drei unabhängige Classifier zur Seitenklassifikation verwendet:

- *Full-Text*: Wörter, die irgendwo auf der Seite stehen
- *Title / Heading*: Wörter, die im html-title oder head stehen
- *Hyperlink*: Wörter, die in Hyperlinks stehen

5.1.1. Ansatz

Der Ansatz der Autoren beinhaltet die Konstruktion eines probabilitischen Models und die Auswahl derjeningen Seite, die anhand dieses Models mit größter Wascheinlichkeit der betrachteten Klassen zugeordnet wird.

Die vorgestellte Mehode ist eine Variante des naiven Bayes Algorithmus. Zu einem gegebenen Dokument d berechnet sich der Score bezüglich einer Klasse c wie folgt:

$$Score_c(d) = \frac{\log \Pr(c)}{n} + \sum_{i=1}^{T} \Pr(w_i|d) \log\left(\frac{\Pr(w_i|c)}{\Pr(w_i|d)}\right)$$

wobei n die Anzahl der Wörter in d, T die Grösse des Vokabulars und w mit Index i das i-te Wort im Vokabular ist. Pr(wi|c) stellt so die Wahrscheinlichkeit dar, daß ein zufällig gewähltes Wort wi aus einem zufällig gewählten Dokument d in Klasse c vorkommt. Pr(wi|d) ist abhängig von der Häufigkeit, mit der wi in d auftaucht.
Die Klasse, die nun ausgewählt wird ist nun die mit dem grösstem Score. Die verwendete Scorefunktion ist abgeleitet aus der Bayes-Bewertungsfunktion.

Naiver Bayes

Bei naiven Bayes wird angenommen, daß die Reihenfolge, in der die Wörter auftreten unerheblich ist und das Wörter unabhägig voneinander auftreten.

Wir haben nun eine Mange C von Klassen c1...cN und ein Dokument mit n Wörtern w1...wn. Das Dokument wird nun derjenigen Klasse c* zugeordnet, die am wahrscheinlichsten ist, wenn man die Wörter in dem Dokument betrachtet, d.h. c* ist das Maximum der Wahrscheinlichkeiten, daß die Wörter w1...wn zur Klasse c gehören. Solch eine Wahscheinlichkeit Pr(c|w1,...,wn) läßt sich auch wie folgt bschreiben:

$$\Pr(c|w_1, \ldots, w_n) \simeq \Pr(c) \prod_{i=1}^{n} \Pr(w_i|c)$$

Abschätzung der Wort Wahrscheinlichkeiten

Man sieht, daß noch die Wahrscheinlichkeit, daß ein Wort zu einer bestimmten Klasse gehört, bestimmt werden muss. Mit einer speziellen Glättungsmethode ergibt sich:

$$\Pr(w_i|c) = \begin{cases} \dfrac{N(w_i, c)}{T_c + \sum_j N(w_j, c)}, & \text{if } N(w_i, c) \neq 0, \\ \dfrac{T}{T_c + \sum_j N(w_j, c)} \dfrac{1}{T - T_c}, & \text{if } N(w_i, c) = 0, \end{cases}$$

wobei N(wi,c) angibt wie oft wi in der Trainingsmenge für c vorkam, Tc die absolute Anzahl einzigartiger Wörter in c und T die absolute Anzahl einzigartiger Wörter in allen Klassen ist. Mit dieser speziellen Glättung wir verhindert, daß eine Wortwahrscheinlichkeit 0 und somit das Produkt (s.o.) auch 0 wird, was zu extremen Abstufungen in den Abschätzungen führen kann.

Vokabular

Die Autoren sind empirisch zu dem Schluss gekommen das verwendete Vokabular auf 2000 Wörter zu begrenzen.
Das Vokabular wird gewählt, indem die Wörter entsprechend ihres durchschnittlichen Informationgehaltes bezüglich der Klassenauszeichnungen in ein Ranking gebracht werden.

5.1.2. Experimentelle Bewertung

Läßt man nun diese Methode auf der gewählten Testumgebung laufen, ergeben ich folgende Einteilungen (es wurde nur der full-text Classifier verwendet):

	Actual							
Predicted	course	student	faculty	staff	research_project	department	other	**Accuracy**
course	**202**	17	0	0	1	0	552	26.2
student	0	**421**	14	17	2	0	519	43.3
faculty	5	56	**118**	16	3	0	264	17.9
staff	0	15	1	**4**	0	0	45	6.2
research_project	8	9	10	5	**62**	0	384	13.0
department	10	8	3	1	5	**4**	209	1.7
other	19	32	7	3	12	0	**1064**	93.6
Coverage	82.8	75.4	77.1	8.7	72.9	100.0	35.0	

Abb.2 Konfusionsmatrix des full-text Classifiers

Diese Matrix läßt sich so lesen: In den Spalten (Actual) stehen die Klassen und wie nun die Instanzen dieser Klassen vom System zugeordnet wurden. Die fettgedruckten Zahlen stehen für die korrekten Zuordnungen. In den Zeilen (Predicted) steht wie das System die Instanzen den Klassen zugeordnet hat. Coverage gibt an wieviele der zur Klasse gehörigen Instanzen der Klasse zugeordnet wurden. Accuray gibt an wie viele Instanzen insgesamt der Klasse richtig zugeordnet wurden.

Es fällt auf, daß die Coverage bei fast allen Klassen relativ hoch ist.
Um einen Eindruck zu bekommen wie das System intern arbeitet, kann man sich die gelernten Classifier nach dem Training anschauen. Es werden die Wörter betrachtet, die bei der Bildung der Scorefunktion der entsprechenden Klasse den grössten Einfluss hatten.

student		faculty		course	
my	0.0247	*DDDD*	0.0138	course	0.0151
page	0.0109	of	0.0113	*DD* : *DD*	0.0130
home	0.0104	and	0.0109	homework	0.0106
am	0.0085	professor	0.0088	will	0.0088
university	0.0061	computer	0.0073	*D*	0.0080
computer	0.0060	research	0.0060	assignments	0.0079
science	0.0059	science	0.0057	class	0.0073
me	0.0058	university	0.0049	hours	0.0059
at	0.0049	*DDD*	0.0042	assignment	0.0058
here	0.0046	systems	0.0042	due	0.0058
researc_project		department		other	
group	0.0060	department	0.0179	*D*	0.0374
project	0.0049	science	0.0153	*DD*	0.0246
research	0.0049	computer	0.0111	the	0.0153
of	0.0030	faculty	0.0070	eros	0.0010
laboratory	0.0029	information	0.0069	hplay *D*	0.0097
systems	0.0028	undergraduate	0.0058	u *DD* b	0.0067
and	0.0027	graduate	0.0047	to	0.0064
our	0.0026	staff	0.0045	bluto	0.0052
system	0.0024	server	0.0042	gt	0.0050
projects	0.0020	courses	0.0042	that	0.0043

Abb.3

Diese Top-Ten Listen zeigen also die Wörter, die das System als höchstwertigste (full-text)Classifier für eine bestimmte Klasse gelernt hat.

Gibt es nun Möglichkeiten die Accuracy zu erhöhen?
Ein Ansatz besteht darin die Werte der Scorefunktion nur dann zu berücksichtigen, wenn diese einen gewählten Schwellenwert überschreiten. Man erhält so eine höhere Accuracy auf Kosten der Coverage. Die folgende Abblidung stellt diesen Tradeoff bei den *full-text* Classifiern dar:

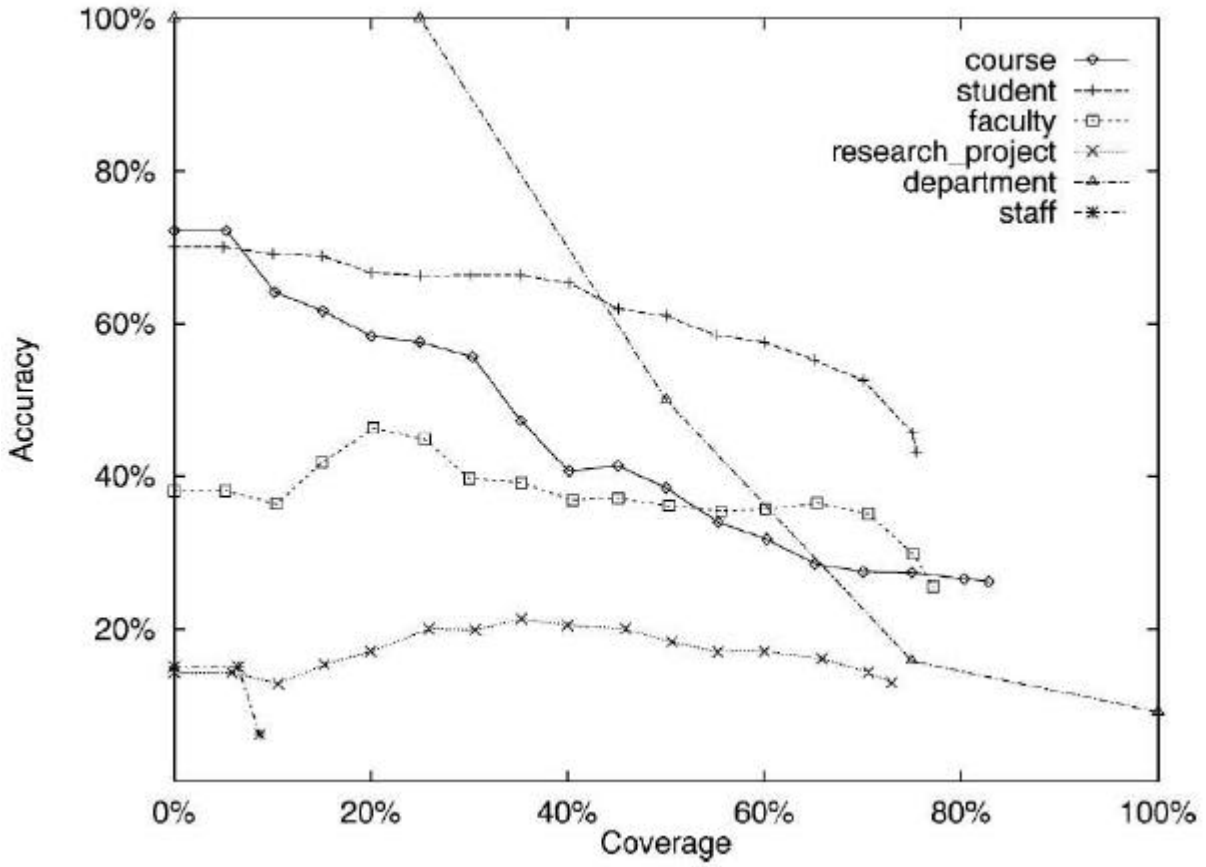

Abb.4 Accuracy / Coverage Tradeoff (full-text)

In ähnlicher Weise läßt sich die Accuracy bei den *hyperlink* und *title/heading* Classifiern anheben. Allerdings bleibt die Accuracy bei den einzelnen Classifiern noch zu niedrig.

5.2. Logische Klassifikation

Das Ziel dieser Methode ist es Instanzen von Klassen zu finden, indem die Umgebung einer Webseite genauer untersucht wird. Insbesondere werden Wörter in benachbarten Seiten und die Struktur von Hyperlinks um die Seite herum berücksichtigt. Der folgende Ansatz induziert logische Regeln 1.Grades.

5.2.1. Ansatz

Der verwendete Lernalgorithmus heißt FOIL Algorithmus. Dieser arbietet nach der greedy-Methode und lernt funktionsfreie Hornklauseln. Der Algorithmus startet mit einer ‚leeren' Klausel für einer Klasse und fügt solange per hill-climbing Suche Argumente hinzu bis (fast) nur positive Instanzen von ihr abgedeckt werden.

Es werden folgende Hintergundrelationen zur Darstellung verwendet:

- has_*word*(Page): Diese Menge von booleschen Relationen gibt an, ob das entsprechende Wort auf der Seite im Parameter vorkommt. Das bezieht sich auch auf die von der Grundform abgeleitet Wörter.

- link_to(Page,Page): Die Relation repräsentiert einen Hyperlink zwischen den Seiten in den Parametern.

Um die Vorhersagen besser beurteilen zu können wird ein Accuray-Maß eingeführt. Als erstes wird die Accuracy einer Klausel mit der Berechnung eines sogenanten m-estimate bestimmt:

$$m\text{-estimate accuracy} = \frac{n_c + mp}{n + m},$$

wobei nc die Anzahl von Instanzen ist, die richtig klassifiziert wurden, n die Gesamtanzahl der klassifizierten Instanzen, p die vorherige Abschätzung der Regel und m eine Konstante ist, die angibt wie stark p im Vergleich zu den untersuchten Daten gewichet wird.

Nun werden die Klauseln gemäß dieser Bewertung in eine gemäß der Accuracy absteigende Reihenfolge gebracht. Danach werden die Regeln in Abhängigkeit von den Classifiern bewertet:

- Falls keine Classifier eine Regel hat die zur gegebenen Seite passt, so wird other mit Zuverlässigkeit 1.0 vorhergesagt.
- Falls nur ein Classifier eine passende Regel hat, so sagen wir die zum Classifier gehörende Klasse mit der Zuverlässigkeit des m-estimate dieses Classifiers vorher und die Klasse other mit 1 minus diesem Wert.
- Falls es mehrere Classifier mit passenden Regeln gibt, so sagen wir jede Klasse mit der Zuverlässigkeit des für die Klasse besten Classifiers geteilt durch die Gesamtzahl der Classifier für die es passende Regeln gibt. Die Klasse other wird mit dem Rest bewertet, der der Summe dieser Bewertungen zu 1 fehlt.

5.2.2. Experimentelle Bewertung

Nach der Bewertung ergaben sich folgende Accuray / Coverage Tradeoff Kurven:

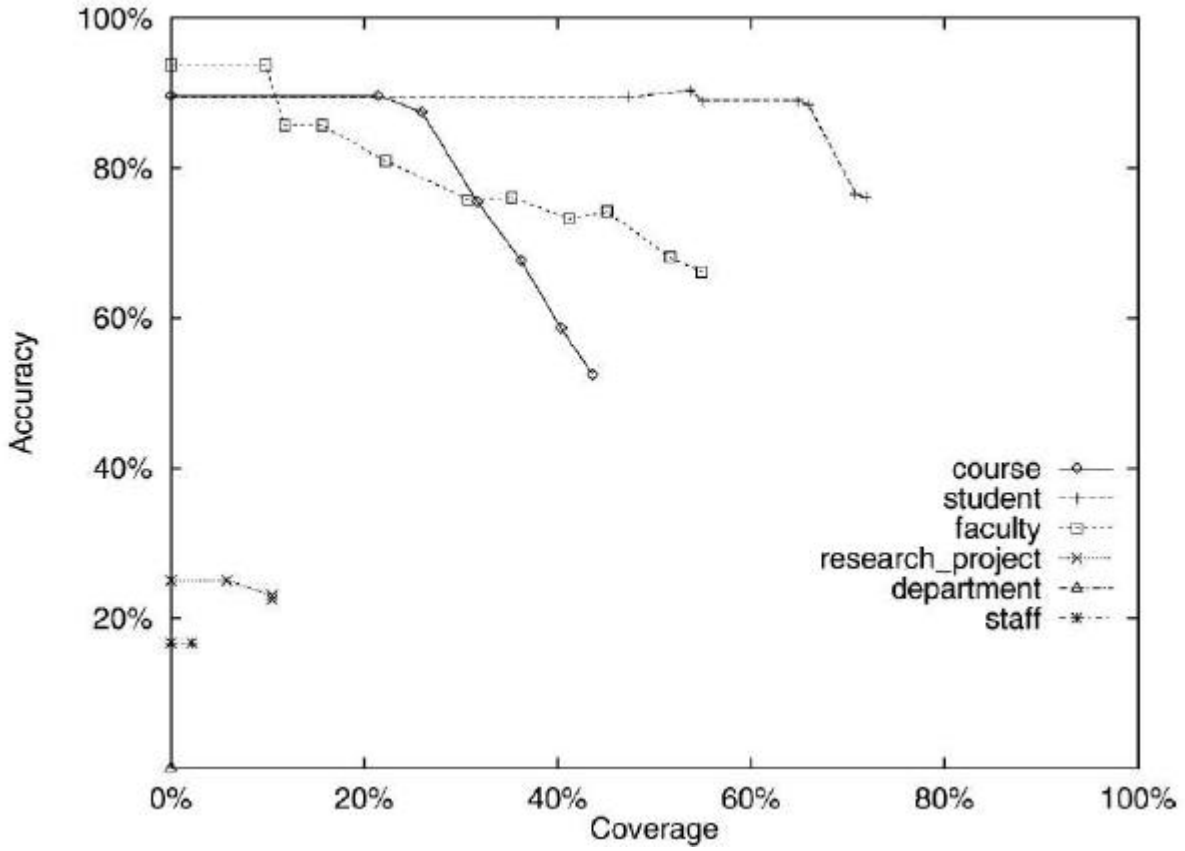

Abb.5 Accuracy / Coverage Tradeoff (logische Regeln)

Wenn man dieses Ergebnis mit dem der statistischen Methode vergleciht (Abb.4), so erkennt man, daß die logischen Regeln, obwohl sie generell eine niedrigere Coverage bieten als die statistischen Classifier, eine höhere Accuracy für einige Klassen besitzen.

Es folgen einige der nach FOIL gelernten Regeln als Beispiel:

---

student(A) :- not(has_*data*(A)), not(has_*comment*(A)), link_to(B,A), has_*jame*(B),
has_*paul*(B), not(has_*mail*(B)).
`Training Set: 147 Pos, 0 Neg; Test Set: 126 Pos, 5 Neg`

faculty(A) :- has_*professor*(A), has_*ph*(A), link_to(B,A), has_*faculti*(B).
`Training Set: 47 Pos, 0 Neg; Test Set: 18 Pos, 3 Neg`

course(A) :- has_*instructor*(A), not(has_*good*(A)), link_to(A,B), not(link_to(B,_1)),
has_*assign*(B).
`Training Set: 31 Pos, 0 Neg; Test Set: 31 Pos, 3 Neg`

Die course(A)- Regel hat z.B. die folgende Bedeutung:
Eine Seite wird als Homepage eines Kurses bestimmt, wenn folgende Kriterien erfüllt sind:

(1) Die Seite enthält das Wort *instructor*, aber nicht das Wort *good*.
(2) Die Seite enthält einen Hyperlink zu einer Seite, die keine Hyperlinks zu anderen Seiten enthält.
(3) Diese durch den Link erreichbare Seite enthält das Wort *assign*.

Alle diese Regeln zeigen, daß Webseiten-Klassifikation verschieden von gewöhnlicher Textklassikation ist. Lernstrategien in dieser Domäne, die solche logischen Regeln benutzen, sollten also effektiver sein als Standardtechniken.

5.3. Kombinierte Lernstrategien

Die vorherigen Untersuchungen zeigen, daß die beste Darstellung einer Seitenklassifikation von der Klasse abhängig ist. Dies legt den Ansatz nahe die Vorhersagen aller 4 Klassifier (*full-text*, *title/heading*, *hyperlink*, *logische Regeln*) zu kombinieren.

5.3.1. Ansatz

Um die Vorhersagen zu kombinieren wird einfach die Seite gewählt, die von der Mehrheit der einzelnen Classifier Stimmen bekommen hat. Gibt es keine Mehrheit, so wird die Klasse des Classifiers genommen, der die zuverlässigste Vorhersage gemacht hat.

5.3.2. Experimentelle Bewertung

Accuracy / Coverage Tradeoff der kombinierten Classifier bei 2000 Vokabeln:

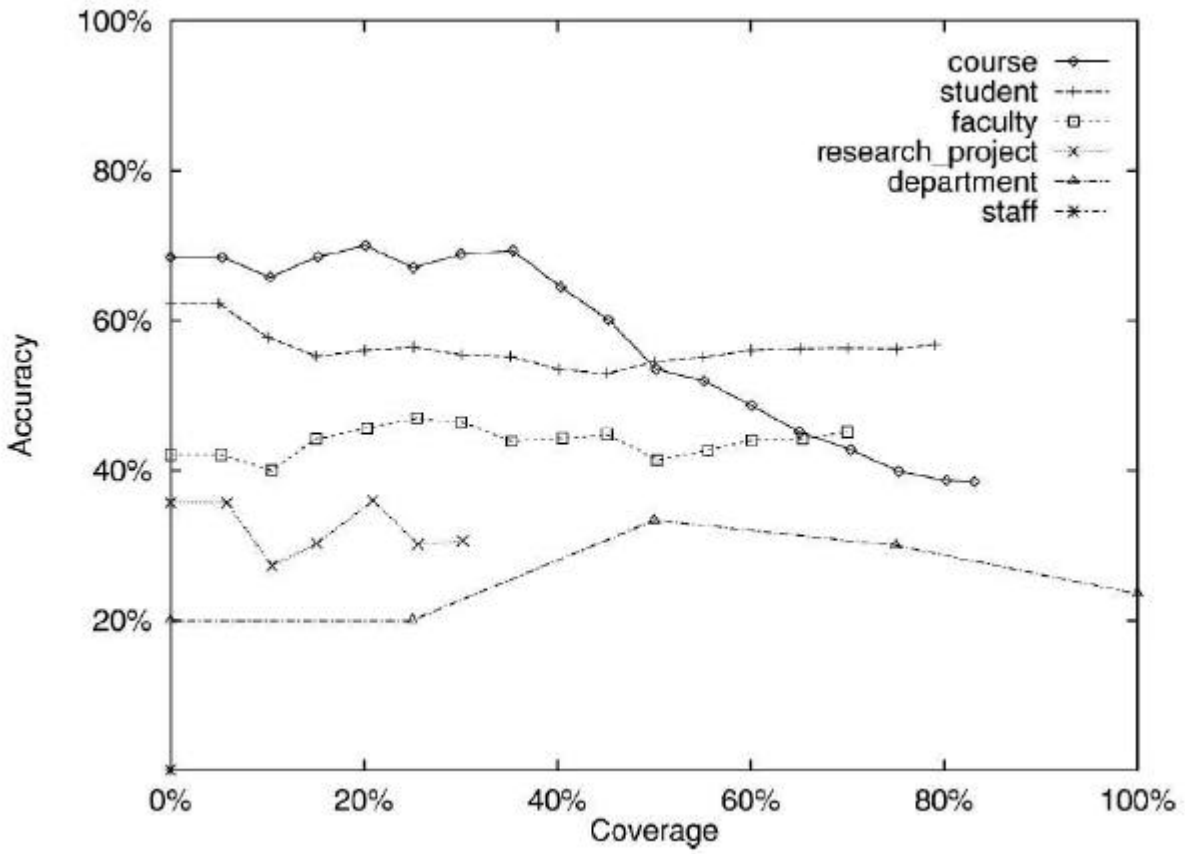

Abb.6 Accuracy / Coverage Tradeoff (kombinierte Classifier)

Vergleicht man diesen Plot wieder mit Abbildung 4 so sieht man, daß generell akkuratere Vorhersagen möglich sind, wenn man auch andere als den *full-text* Classifier berücksichtigt.

Dennoch ist dieses Ergebnis etwas entäuschend. Die Kurven für die kombinierten Vorhersagen sind nicht überall besser als die Kurven der zugrundeliegenden Classifier. Wünschenswert wäre es, wenn die Kurven mindestens so gut sind wie der beste der zugrundeligenden Classifier.

Die Autoren vermuten, daß die simple Kombination der Classifier ein Grund für diese Ergebnisse ist, insbesondere das Mapping von Klassifikation scores auf die geschätzte Accuracy sei hierfür verantwortlich. Weiterhin wurde beobachtet, daß sich die Ergebnisse durch Einschränkung des Vokabulars für die statistischen Classifier auf 200 Vokabeln wesentlich verbessert. Bei dem ensprechendem Plot sind die Kurven stets besser als die des *full-text* Classifiers.

Die Autoren planen in Zunkunft hier verbesserte Kombinationsmethoden zu untersuchen.

5.4. Identifikation von Multi-Page Segmenten

Da jede Instanz einer Klasse wie in Abschnitt 3 beschrieben durch ein Segment dargestellt wird, ist es möglich das eine Instanz durch mehrere verlinkte Seiten beschrieben wird.

Es besteht aber weiterhin die Annahme, daß jede Instanz durch eine einzelne Seite beschrieben wird, dürfen wir nur eine Seite in die Knowdledge-Base stecken. Das hat zur Folge, daß es z.B. Homepages von Studenten gibt, bei denen mehrere Seiten diesselbe Person beschreiben, aber nur die primäre Seite der Klasse student und die anderen Seiten der Klasse other zugeordnet werden.

Im folgenden wird nun eine Methode zum Identifizieren solcher Mult-Page Segmente vorgestellt, die oben erwähnten Misstand beseitigen soll.

5.4.1. Ansatz

Wir teilen zunächst das Problem in 2 Teilprobleme:

(1) Gruppiere verbundene Seiten zusammen.
(2) Finde die repräsentativste Seite aus dieser Gruppierung.

Das Gruppierungsproblem wurde von Hand gelöst. Es wurden Regeln vorgegeben, die Seiten als verbunden erkennen sollen. So werden z.B. gleiche Verzeichnispräfixe in den URLs als ein Verbundenheitskriterium gewertet, wobei der längste dieser Präfixe mehrerer URLs als Gruppierungsindikator dient.

Diese Gruppierungen werden jetzt als die Beschreibung einer Instanz angesehen. Nun gilt es (2) zu lösen. Da die repräsentativste Seite einer Gruppierung gewöhnlich deren Hauptseite ist, wird z.B. eine Seite mit dem charakteristischem Namen „index.html“ zur Primärseite (repräsentative Klasse) dieser Gruppierung erklärt. Jede Seite, die mit diesem oder ähnlichen Mechanismen nicht erkannt wird geht in die Klasse **other**. Falls keine Seite mit diesen Heuristiken zur Primärseite gemacht wurde, wird die Seite mit der höchsten Klassifizierungszuverlässigkeit Primärseite.

5.4.2. Experimentelle Bewertung

Im Accuracy / Coverage – Tradeoff Plot ist nach der Anwendung der URL-Heurstiken für den *full-text* Classifier ein erheblicher Anstieg der Accuracy bei einem leichtem Rückgang der Coverage zu beobachten:

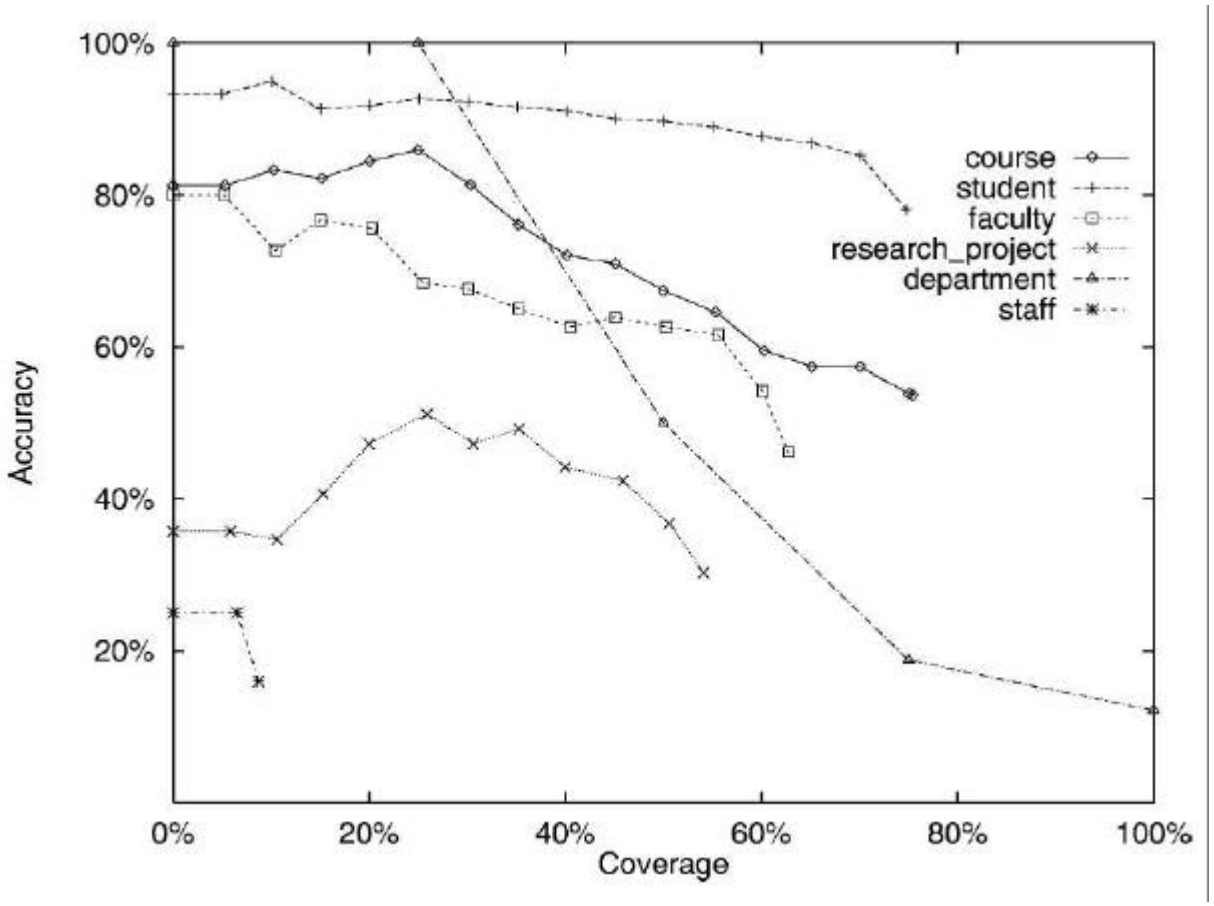

Abb.7 Accuracy / Coverage Tradeoff (full-text + URL-Heuristiken)

5.5. Zusammenfassung

Es wurde gezeigt, daß – weil Hypertext viele redundante Informationen enthält – Webseiten klassifiziert werden können, indem verschiedene Informationsquellen herangezogen werden: full text, Text im Titel/Heading, Text assoziert mit Hyperlinks, Text auf benachbarten Seiten und die Dateiorganisation, die in URLs dargestellt wird.

Keine dieser Informationen allein kann jedoch eine Instanz einer Klasse mit genügender Genauigkeit erkennen. Ein erfolgversprechender Ansatz ist es mehrere Classifier mir unteschiedlicher Informationsbasis zu kombinieren.

6. Erkennen von Relationsinstanzen

In diesem Abschnitt geht es darum eine Methode zu finden Instanzen von Relationen zwischen den extrahierten Klasseninstanzen zu finden. Wir gehen davon aus, daß Relationsinstanzen oft durch Hyperlinkpfade dargestellt werden. So z.B. kann eine Instanz der Relation **instructor_of** durch folgende Regel dargestellt werden:

instructor_of(A,B) :- course(A), person(B), link_to(A,B)

Die Regel sagt, daß eine Instanz von **instructor_of** gefunden wurde, wenn es eine Kursseite gibt, die einen Link zu einer Seite der Klasse person hat.

6.1. Problemrepräsentation

Zu beachten ist, daß die Hyperlinkpfade variable Länge haben können. Aus diesem Grund wird wieder eine logische Darstellung der gelernten Regeln verwendet.

Hier einige wichtige Hintergrundrelationen:

- *class*(Page): Gibt an, daß die Seite eine Instanz von *class* ist.
- link_to(Hyperlink, Page, Page): Eine Hyperlinkrelation, der 1.Paramter ist ein Identifier, der 2. gibt die Startseite, der 3. die Zielseite an.
- has_*word*(Hyperlink): Gibt an ob *word* im Ankertext des Hyperlinks enthalten ist.

6.2. Lernstrategien

Der Algorithmus, der zum Lernen der Hornklauseln verwendet wird, ist ähnlich dem von FOIL. Allerdings gib es einige Variationen.

Die Autoren fanden heraus, daß es mit hill-climbing Suche nicht möglich ist einen Hyperlinkpfad mit mehr als einen Hyperlink zu lernen. So entschied man sich den ‚Pfadanteil' der Klauseln anders zu lernen und später mit hill-climbing weitere Literale hinzuzufügen. So entstand ein relationaler Pfadfindungsalgorithmus. Dieser benutzt Backtracking auf dem Graphen, der durch gegebene Konstanten (Webseiten) und Relationen zwischen den Konstanten gebildet wird.

Input: training set of negative and uncovered positive instances

1. for each uncovered positive instance
2. find a path (up to bounded length) using the background relations
3. select the most common path prototype for which clause search hasn't yet failed
4. generalize the path into an initial clause
5. do hill-climbing to refine the clause
6. if hill-climbing fails to find an acceptable clause, backtrack to step 3.

Return: learned clause

Abb.8: Der relationale Pfadfindungsalgorithmus

Der erste Schritt besteht darin, einen kürzesten Pfad für jede nicht durch eine vorherige Regel abgedeckte, positive Instanzen der Zielrelation zu finden. Bildich ist das so zu sehen:

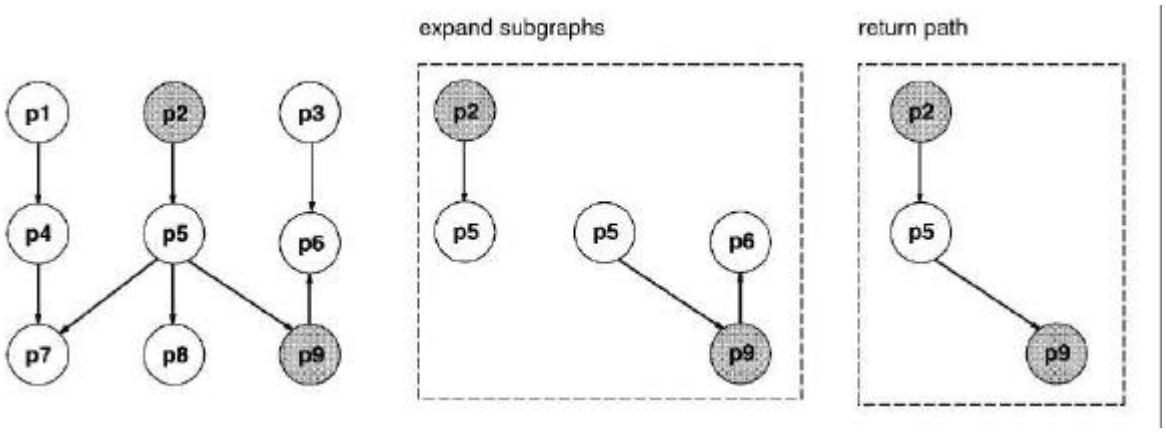

Abb.9: So arbeitet der Pfadfindungsalgorithmus

In Abb.9 ist zu sehen wie die Subgraphen um die Konstanten p2 und p9 erweitert werden. Die geschieht, indem alle Konstanten zu dem Subgraph hinzugenommen werden, die über die gegebenen Hintergrundrelationen (z.B. link_to) erreicht werden können.

Nachdem jetzt ein Pfad für jede nicht abgedeckte positive Instanz gefunden wurde, wird derjenige Pfadprototyp, der am häufigsten auftritt, als initiale Klausel verwendet. Ein Pfadprototyp speichert hierbei die Struktur eines Pfades (wie Seiten über Hyperlinks zusammenhängen, Richtung der Hyperlinks) nicht jedoch irgendeine spezielle Seite oder Hyperlink als Instanz.

Die Klausel wird anschliessend weiter mit der hill-climbing Suche verbessert. Wird keine akzeptable Klausel mit dieser Suche gefunden, so wird der zuletzt gefundene Pfadprototyp gelöscht und der nächsthäufigste Pfadprototyp wird betrachtet.

Als weitere Variation zu FOIL wird eine spezielle Bewertungsfunktion gewählt, die es dem Algorithmus erlaubt, weniger, dafür aber allgemeinere Regeln zu lernen.

6.3. Experimentelle Bewertung

Hier einige der gelernten Regeln:

---

instructors_of(A,B) :- course(A), person(B), link_to(C,B,A).

Test Set: 133 Pos, 5 Neg

department_of(A,B) :- person(A), department(B), link_to(C,D,A), link_to(E,F,D), link_to(G,B,F), has_neighborhood_word_*graduate*(E).

Test Set: 371 Pos, 4 Neg

members_of_project(A,B) :- research_project(A), person(B), link_to(C,A,D), link_to(E,D,B), has_neighborhood_word_*people*(C).

Test Set: 18 Pos, 0 Neg

Man sieht, daß diese Regeln (2 und 3) auch problemlos komplexere Relationen mit mehreren Hyperlinks beschreiben können.

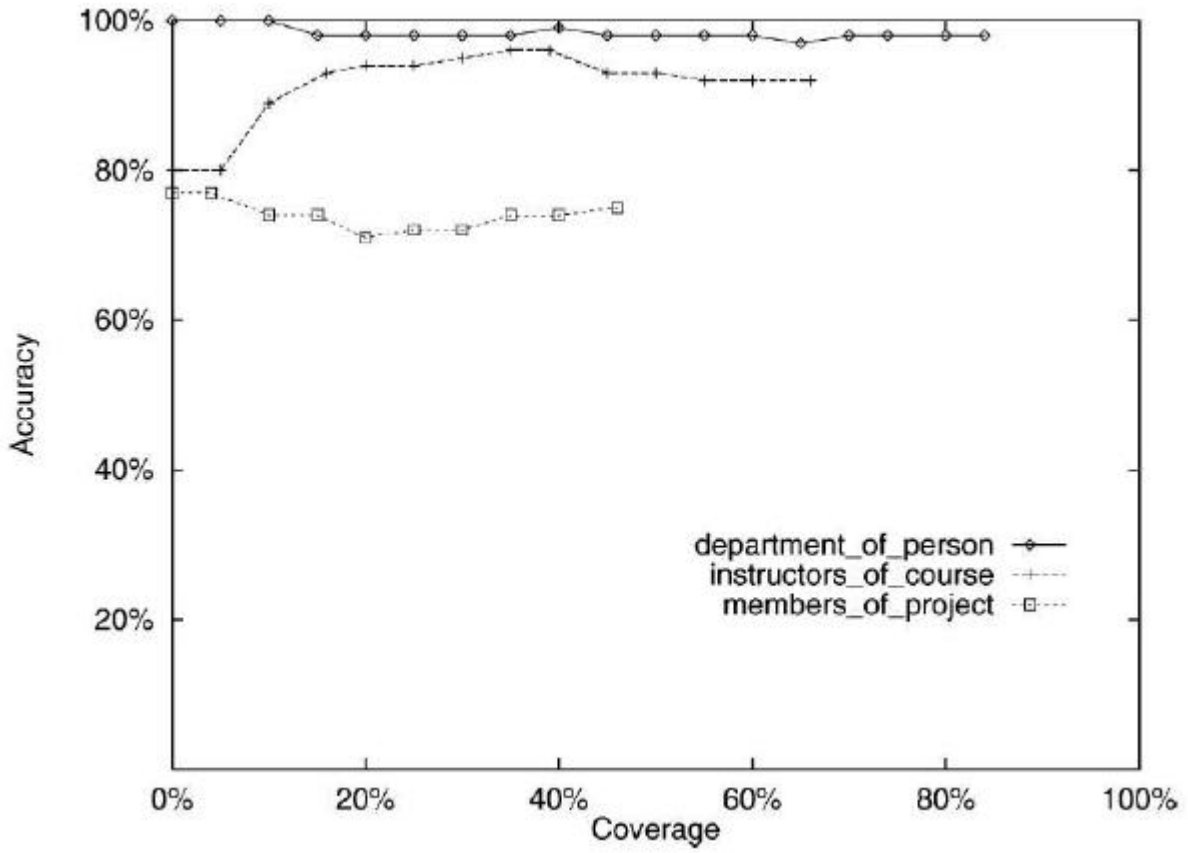

Abb. 10 Accuracy / Coverage Tradeoff (logische Regeln für Relationen)

Bei diesem Tradeoff-Plot fällt die hohe Accuracy bei einigen Relationen auf. Allerdings ist die Coverage bei manchen Relation (members_of_project: 46%) eher niedrig. Das ist bedingt durch die niedrige Coverage bei den Seiten-Classifiern, von denen ja die Coverage-Werte bie den Relationen abhängen.

7. Informationsextraktion aus Textsegmenten

Gelegentlich sind die zu extrahierenden Informationen in kleineren Textfragmenten enthalten (z.B. der Name einer Person auf seiner Homepage). Dieses Problem ist eine Aufgabe der Informationsextraktion im engerem Sinne.

7.1. Ansatz

Der vorgstellte Lernalgorithmus ist wieder ein logischer Lerner im Sinne FOILs. Die Autoren nennen ihn SRV (Sequence Rules with Validation). SRV erhält als Eingabe eine Menge der bezüglich der zu extrahierenden Textfelder gelabelter Seiten (also eine Trainingsmenge) und eine Menge von Features, die zur Regelbildung verwendet werden können. Ausgabe sind dann Regeln zur Informationsextraktion.

Wie bei FOIL wird hill-climbing verwendet, um Regeln zu verbessern. Bei jedem Schritt wird ein Literal, das die meisten noch nicht abgedeckten positiven Instanzen überdeckt, während es eine grosse Anzahl negativer Instanzen ausschliesst, zur Regel hinzugefügt. Wenn eine Regel als gut genug betrachte wird (Das ist der Fall, wenn alle positiven Instanzen überdeckt wurden oder eine weitere Bearbeitung der Regel als unproduktiv angesehen wird) so werden alle von der Regel überdeckten positiven Instanzen aus der Trainingsmenge entfernt und der Prozess wird wiederholt. In dieser speziellen Domäne ist

eine positive Instanz ein gelabeltes Textfragment und eine negative Instanz ein ungelabeltes Textfragment, das diesselbe Grösse wie eine positives Textfragment besitzt.

Die eingebenen Features helfen bei der Analyse eines Textfragmentes. Es werden Aussagen über ein Token in einem Textfragment (einfaches Feature) oder über Beziehungen zwischen den Tokens innerhalb eines Textfragmentes (relationales Feature) gemacht.

So kann z.B. beschrieb weden, ob ein Wort mit einem Grossbuchstaben beginnt (einfaches Features) oder ob ein Wort auf ein anderes folgt (relationales Features).

Literale werden aus Templates generiert, wobei diese auch Aussagen über das gesamte Fragment machen können. Das sogenannte some-Template bildet eine Quantifizierung, bei der Features getestet werden.

Hier eine Extraktionsregel für den Namen des Besitzers einer Homepage:

```
ownername(Fragment) :- some(B, [ ], in_title, true),
                       length(<, 3),
                       some(B, [prev_token], word, "gmt"),
                       some(A, [ ], longp, true),
                       some(B, [ ], word, unknown),
                       some(B, [ ], quadrupletonp, false)
```

Diese Regel ist so zu lesen: Ein Fragment hat den Namen eines Besitzers, wenn es eine Sequenz aus 2 Tokens gibt, wobei das eine (A) innerhalb eines HTML-title Feldes steht und länger als 4 Buchstaben ist und das andere (B) auf eine Token ‚gmt' folgt, unbekannt aus der Trainingsphase ist und kein 4-Buchstaben-Wort ist.

7.2. Experimentelle Bewertung

Die obige Regel wurde nun mit Seiten die Instanzen der Klasse **person** getestet. Das folgende Fragment wird z.B. von obiger Regel erkannt:

```
Last-Modified: Wednesday, 26-Jun-96 01:37:46 GMT

<title> Bruce Randall  Donald</title>

<h1>
<img src="ftp://ftp.cs.cornell.edu/pub/brd/images/brd.gif">
<p>
Bruce Randall Donald<br>
Associate Professor<br>
```

Zu beachten ist, daß der Name falsch extrahiert wird, da der Nachname hier von der Regel ignoriert wird (es hieß ja: ‚eine Sequenz aus 2 Token‘).
Dennoch erhält man bei der Namensextraktion mit SRV (wenn man Seiten, auf denen der Name des Besitzers nicht vorhanden ist, unberücksichtigt läßt) eine Accuracy von 77.4%.

8. Fazit und Ausblick

Abschliessend betrachtet sind die Autoren der Meinung, daß sich ihre These (s. Einführung) durch die Ergebnisse des Prototyp bestätigt hat. Er arbeitet mit einer Accuracy von über 70% bei einer Coverage von ca. 30%.

Dies wurde u.a. durch die Berücksichtigung der Struktur von Hypertextdokumenten ermöglicht. Es gab eine Sicht nach außen, die die Umgebung von Webseiten betrachtete (logische Regeln), wowie eine Sicht nach innen, die auch Instanzen in einzelnen Textabschnitte suchte (SRV).

Es wird bemerkt, daß die vorgestellten Methoden zur Extraktion wahrscheinlich auch in neuen Domänen problemslos laufen, wenn sie in angepasster Weise kombiniert werden. Jedoch ergibt sich die beste Methode zur Extraktion immer in Abhängigkeit von der Darstellung der Instanzen einer Domäne.

Die hier vorgestellten Methoden sind (noch) relativ neu, demzufolge gibt es ein breites Feld zu untersuchender Verbesserungsmöglichkeiten des beschriebenen Systems.

Ein Auszug der von den Autoren vorgeschlagenen Verbesserungsmöglichkeiten:

- Entwicklung von Lernmethoden, die hierarchische Beziehungen in der Ontologie berücksichtigt
- Ungelablte Seiten in Kombination mit gelabelten Seiten verwenden, um die Accuracy beim Training zu erhöhen
- Mehr linguistische Struktur (Nomen, Verben, ...) berücksichtigen
- Mehrere Strategien zur Extraktion aus Textsegmenten entwickeln
- Die Struktur von HTML genauer auswerten