ILP-basierte Klassifikation von Web-Dokumenten

Jens Hartmann
Institut AIFB, Universität Karlsruhe
76128 Karlsruhe, Germany
hartmann@aifb.uni-karlsruhe.de

Abstract

In dieser Arbeit wird ein semi-automatisches Verfahren zur Erkennung und Extraktion relationaler Regularitäten in Web-Dokumenten präsentiert und für die Klassifikation im *wissensbasierten Content-Management*, basierend auf Spectacle, eingesetzt. Exemplarisch wurde das Verfahren auf *Umweltinformationssysteme* angewendet und die Ergebnisse evaluiert.

1 Einleitung

Die Daten- und Informationsverarbeitung hat in der Gesellschaft eine weite Verbreitung und einen hohen Stellenwert erreicht. Ein Beispiel einer globalen Anwendung ist das World Wide Web (WWW), durch das auf Millionen von Seiten Informationen bereitgestellt werden. Die hohe Informationsvielfalt und die heterogene Struktur erschweren jedoch eine *effiziente Suche* nach geeigneten und nützlichen Informationen [Chakrabarti, 2000; Leighton and Srivastava, 1997].

Insbesondere ist die *Extraktion Kontext relevanter Informationen*, basierend auf der Analyse semi-strukturierter Dokumente[1], ein auf *spezielles Domänenwissen* basierender, *zeitintensiver* und manuell nur bedingt durchführbarer Prozess. Dies ist durch eingeschränkte *Inferenz-Mechanismen*, insbesondere durch das Fehlen von *semantischem Markup*, begründet. Der Ansatz Wissen für Menschen und Maschinen interpretierbar zu repräsentieren wird als *Semantic Web* [Berners-Lee and Fischetti, 1999] bezeichnet, welches zu einer Vielzahl von neuen Anwendungsmöglichkeiten führt [Decker *et al.*, 1999; Farquhar *et al.*, 1996; Heflin *et al.*, 1999; Benjamins, 1998]. Ontologien [Guarino, 1998] werden dabei zur Repräsentation von Informationen und Wissen verwendet und stellen eine Grundlage des *Semantic Web* dar [Studer *et al.*, 1998; Benjamins and Fensel, 1998; Benjamins *et al.*, 1998].

Ein Ansatz zur Wissensrepräsentation durch Ontologien wird in dem *Spectacle* Verfahren [van Harmelen and van der Meer, 1999] beschrieben und lässt sich bereits auf die bestehende Struktur des WWW und dessen Dokumente anwenden [Harmelen, 2000]. Eine Ontologie repräsentiert hierbei ein Modell mehrerer Web Dokumente, welches aus einer Menge von einzelnen Kategoriebeschreibungen gebildet wird. Die Beschreibung der jeweiligen Kategorien erfolgt manuell durch eine Spezifikation von Klassifikationsregeln.

Die manuelle Deklaration der Kategoriebeschreibungen in Spectacle erfordert *spezielles Domänenwissen* des Anwenders und bedeudet bei steigender Anzahl beziehungsweise Komplexität der Dokumente einen entsprechend steigenden Arbeitsaufwand. Die Notwendigkeit der Existenz von Hintergrundwissen über die zu untersuchenden Dokumente erschwert dabei die Verarbeitung unbekannter Dokumente.

Ein möglicher Ansatz zur semi-automatischen Generierung von Kategoriebeschreibungen ist die sogenannte *Web Seiten Kategorisierung* [Pierre, 2001; Moore *et al.*, 1997], bei der zum einen Kategorien von Dokumenten identifiziert werden und zum anderen *ungesehene* Dokumente deklarierten Kategorien zugeordnet werden. Für eine semi- automatische Klassifikation wurden bisher verschiedenste Ansätze verfolgt [Craven *et al.*, 2000; Yang *et al.*, 2002; Ghani *et al.*, 2001b; Ghani *et al.*, 2001a]. Web Dokumente bieten unterschiedliche *Informationsarten*, die zur Klassifikation genutzt werden können. Die Repräsentation der Dokumente im WWW basiert auf einer *Markup Language (ML)*. Die Anwendung klassischer Textklassifizierungsmethoden ist auf Web Dokumente nur beschränkt möglich, da die besonderen Eigenschaften und Strukturen eines Web Dokumentes im Allgemeinen nicht in einem Textklassifizierer repräsentiert werden können.

Im Rahmen dieser Arbeit wird ein Verfahren zur semi-automatischen Generierung von Klassifikatoren für Webdokumente vorgestellt. Die Klassifikatoren basieren dabei auf relationalen Struktureigenschaften der Dokumente und benötigen kein zusätzliches Hintergrundwissen, wodurch der Ansatz auf unbekannte Dokumente anwendbar ist. Das entwickelte Verfahren führt dazu eine Analyse verschiedener *syntaktischer Eigenschaften* in den Dokumenten durch. Hierzu werden Informationen über die *syntaktische Struktur der Web-Seiten*, *Metadaten*, *Dokumentenadresse (URL)* und *relationale Eigenschaften* zur Generierung der Klassifikatoren berücksichtigt und in einer *formalen Repräsentation* dargestellt. Die Arbeit basiert auf vorangegangenen Arbeiten [Hartmann, 2002; Stuckenschmidt *et al.*, 2002]. In den durchgeführten Experimenten konnten durchschnittliche Erkennungsraten von 98,41 Prozent erreicht werden [Hartmann and Stuckenschmidt, 2002].

Die Arbeit gliedert sich wie folgt. Zunächst er-

[1] In dieser Arbeit werden mit dem Begriff *Dokumente* HTML und XML Dokumente im Internet oder Intranets bezeichnet.

folgt eine **Formalisierung der Problemstellung**, welche aus der Definition von logischen Prädikaten zur Repräsentation von Dokumenten und deren Relationen besteht. Darüberhinaus wird das Lernproblem für diesen Ansatz formalisiert. Im nächsten Abschnitt wird die **Vorverarbeitung der Dokumente** beschrieben. Die Darstellung von bekannten und unbekannten Strukturen wird daraufhin detailiert im Abschnitt **Formale Repräsentation der Dokumente** erläutert.Das Verfahren wurde exemplarisch auf web-basierte *Umwelt-Informations-Systeme* angwendet. Die Methodik der Durchführung der Experimente und deren Ergebnisse werden im Abschnitt 5 erläutert. Die Arbeit schliesst mit einer Zusammenfassung und gibt Ausblicke auf weitere Arbeiten.

2 Formalisierung der Problemstellung

In diesem Abschnitt wird die Transformation und formale Repräsentation von Dokumenten in logischen Ausdrücken formalisiert und die Methodik zur Generierung relationaler Regeln mittels ILP beschrieben.

Der Ansatz verwendet eine **formale Repräsentation** der Web-Dokumente, welche in die *Repräsentation der logischen Struktur* und in die *Repräsentation der relationalen Struktur* unterteilt wird. Die formale Repräsentation der Dokumente und ggf. vorhandenes Hintergrundwissen werden zur **(semi-) automatischen Regelerzeugung** eingesetzt, die als Klassifikatoren der gegebenen Dokumente verwendet werden. Zuvor erfolgt eine **Vorverarbeitung** der Daten und eine **syntaktische Transformation** der Dokumente.

2.1 Formalisierung der Repräsentation strukturierter Dokumente

Im folgenden stellen wir die formale Repräsentation strukturierter Dokumente durch formale Aussagen vor, wie sie auch in der *Logischen Programmierung (LP)* verwendet werden. Exemplarisch wird in [Boley, 2000] die Repräsentation von XML Dokumenten näher beschrieben. Weitere mögliche Anwendungen von LP und die Anwendung im Web werden unter anderem in [Boley, 1996; Cabeza and Hermenegildo, 1997; Davison and Loke, 1996] vorgestellt.

Im Rahmen dieser Arbeit wird die formale Repräsentation von HTML und XML Dokumenten betrachtet. Diese Dokumententypen lassen sich als *Document Object Model - DOM* darstellen [Consortium, 2000a]. Ein DOM kann als ein kreisfrei gerichteter Graph betrachtet werden. Dabei stellen die Elemente eines Dokumentes Instanzen einer Strukturklasse dar, welche als Knoten deklariert werden. Die Kanten beschreiben die Beziehungen von Elementen in einem Dokument[2]. Dadurch wird die *formale Repräsentation* der Dokumente auf eine formale Repräsentation eines *Document Object Models* abstahiert[3].

Zur Repräsentation der *logischen Dokumentenstruktur* werden **binäre Prädikate** spezifiziert, welche Eigenschaften von Dokumenten (D) und den enthaltenen Strukturen (S) ausdrücken können. Die Prädikate sind wie folgt definiert.

[2]Nicht zu verwechseln mit der relationalen Struktur zwischen Dokumenten.

[3]Somit können alle Dokumenttypen verarbeitet werden, die sich als Document Object Model darstellen lassen

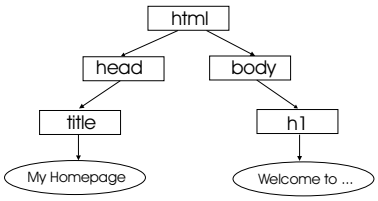

Abbildung 1: Exemplarisches Graphenmodell einer HTML Seite

1. **Existenz von Strukturen** - Das Prädikat $S(O, P)$ in D ist genau dann wahr, wenn eine Struktur O in der Form $< P > Y < /P >$ in D vorkommt, wobei P der Name einer Struktur ist.
2. **Beziehungen von Strukturen** - Das Prädikat $Q(P_x, P_y)$ in D ist genau dann wahr, wenn P_y eine direkte sub-Struktur von P_x in D ist.
3. **Eigenschaften von Strukturen** - Das Prädikat $W(E, Y)$ in D ist genau dann wahr, wenn es für ein Element E eine Zuweisung der Form $E = Y$ in D gibt.

Die formale Repräsentation wird in die Repräsentation der logischen Struktur eines Dokumentes und der relationen Struktur zwischen Dokumenten unterteilt. Für die Repräsentation wird PROLOG Syntax verwendet.

Definition der Prädikatenmenge zur Dokumentrepräsentation

Strukturierte Dokumente bestehen im Allgemeinen aus einem *document header* und einem *document body*. In dem *head* eines Dokumentes können Typdeklarationen oder weitere Verarbeitungsansweisungen definiert werden. Der *body* besteht aus einer Menge von Strukturen, Wertzuweisungen und Relationen. Für die Dokumentendarstellung wird auf Basis der o.g. binären Prädikate eine Grundmenge von speziellen Prädikaten definiert, die Web Dokumente in ihrer logischen Struktur beschreiben können.

- Ein Dokument wird immer durch das Prädikat `document(object)` repräsentiert.
- Jedes Dokument kann zur absoluten Identifizierung das Prädikat `url(object, ADRESS)` aufweisen.
- Elemente in den Dokumenten werden durch definierte syntaktische Deklarationen angegeben. Jedes Element ist eine Instanz einer Strukturklasse. Elemente werden allgemein in folgender Form angegeben.

  `structure(object, CLASS)`

  Wobei `CLASS` der Name der Klasse ist und `object` die jeweilige Instanz eindeutig refenziert, z.B. `structure(obj32, 'p')` beschreibt ein P-Tag eines HTML Dokumentes.
- Elementstrukturen in Dokumenten können weitere Elemente beinhalten, diese Beziehung zwischen Elementen wird durch folgende Relation deklariert.

  `contains(parent, object)`

Die Relation beschreibt somit eine Kante in einem Graphen von einem Elternknoten *parent* zur Instanz *object*.

- Elemente können Attribute in der Form $tag\ a_1 = v_1, ..., a_n = v_n,\ n \geq 0$ besitzen. Jedes Attribut ist wieder ein Element. Die Attributbeziehung der beiden Elemente wird durch folgende Relation ausgedrückt.

  `attribute(parent, object)`

- Werte von Elementen werden durch das Prädikat `value(object, 'VALUE')` dargestellt.
- Werte von Text-Strukturen werden durch das Prädikat `text_value(object, 'TEXT')` beschrieben.

Die Grundmenge der Prädikate erlaubt eine vollständige Repräsentation einer logischen Dokumentstruktur. Instanzen von Strukturklassen in den Dokumenten werden durch das Prädikat `structure(object, CLASS)` dargestellt. Dadurch können unterschiedliche Dokumententypen und vor allem unbekannte Strukturen ohne zusätzliches Hintergrundwissen repräsentiert werden. In dem Prozess der (semi-) automatischen Regelerzeugung für eine Menge von Dokumenten werden weitere Prädikate definiert, die die Menge der Prädikate zur Repräsentation eines Dokumentes einschränken.

Formale Repräsentation von Dokumentrelationen

Beziehungen zwischen Dokumenten werden gesondert betrachtet. Im Allgemeinen werden die Relationen durch folgendes Prädikat ausgedrückt

$$relation(doc1, doc2).$$

Diese Aussage beschreibt eine Relation beziehungsweise eine gerichtete Kante im Graphenmodell von einem Dokument `doc1` zu einem Dokument `doc2`. Es sein angemerkt, dass diese allgemeine Deklaration nicht zwischen Instanzen unterschiedlicher Dokumentenmengen unterscheidet. In der (semi-) automatischen Regelerzeugung wird das Prädikat daher auf eine differenzierte Darstellung unterschiedlicher Dokumentenmengen erweitert.

Einschränkungen: Durch die Grundmenge der definierten Prädikate können XML und HTML Dokumente in ihrer logischen Struktur repräsentiert werden. Folgende strukturelle Eigenschaften eines Dokumentes bleiben in der formalen Darstellung jedoch unbeachtet:

- Die Reihenfolge des Auftretens eines Elementes.
- Document Type Angaben im header.

Eine grundlegende Einschränkung wird in Bezug auf die Anwendbarkeit des Verfahrens gemacht. Das Ziel des Verfahrens ist die Generierung einer Kategoriebeschreibung für eine bestimmte Domäne (*closed world assumption*) und somit repräsentiert eine derartige Generierung keine *open World Beschreibung*.

2.2 Definition des Lernproblems

Die definierte Prädikatenmenge bildet eine eingeschränkte Sprache zur Beschreibung logischer Strukturen eines DOM. Das Lernproblem für diesen Ansatz lässt sich wie folgt definieren.

Gegeben:

- Eine Menge von strukturierten Dokumenten D_1 aus einer Kategorie U und eine Menge von Dokumenten D_2 aus einer Kategorie W. Wobei gilt $D_1 \cap D_2 = \square, D_1 \neq 0, D_2 \neq 0$ und $U \neq W$.
- Eine Sprache λ zur eingeschränkten Beschreibung einer logischen DOM Struktur

Gesucht:

- Ein Klassifikator k für ein Dokument (D), welcher in λ repräsentiert ist, für den gilt:

$$k(D) = \begin{cases} 0 & : \quad D \notin U \\ 1 & : \quad D \in U \end{cases}$$

Durch die formale Repräsentation von Web Dokumenten entsteht eine festgelegte Menge von Prädikaten. Das Ziel ist es nun, eine Regel zu finden, die die Menge der positiven Beispiele beschreibt, ohne dabei Instanzen der negativen Beispiele zu klassifizieren. Diese strikte Deklaration kann durch die Angabe von y Prozent *noise* abgeschwächt werden. Dabei bedeutet diese Angabe, dass alle positiven Beispiele und y Prozent der negativen Beispiele durch eine Regel beschrieben werden dürfen.

Der allgemeine Lernprozess lässt sich wie folgt darstellen:

1. Generierung der Menge aller verfügbaren positiven und negativen Beispiele.
2. Erstellung eines Data Sets: Eine Zufallsverteilung generiert n Training Sets und m Test Sets. Auf diese Sets werden alle positiven Beispiele entsprechend definierter Relationen und Einschränkungen verteilt.
3. Weitere Zufallsverteilung ordnet jedem positiven Beispiel entsprechend dem gegebenen Verhältnis R negative Beispiele zu.
4. Jedes Training Set wird mit der Beschreibung des Hypothesenraumes und dem Hintergrundwissen *Progol* als Eingabe übergeben.
5. Vergleich und statistische Auswertung der generierten Regeln für jedes Training Set.
6. Anwendung aller unterschiedlichen Regeln auf das Test Set mit statistischer Auswertung.

Die Anzahl der TrainingSets n und das Verhältnis R von positiven zu negativen Beispielen wird zu Beginn vom Lehrer angegeben.

3 Vorverarbeitung der Dokumente

Das Verfahren beginnt mit einer *Vorverarbeitung der Dokumente*, die überprüft, ob die Dokumente in einer *einheitlichen Verzeichnisstruktur* und in einem *gemeinsamen syntaktischen Format* vorliegen.

3.1 Syntaktische Transformation

Die *syntaktische Transformation* erfolgt auf Basis eines *Document Object Models (DOM)* [Consortium, 2000a], so dass es dadurch möglich ist alle Dokumententypen, die sich als DOM darstellen lassen, mit diesem Verfahren bearbeiten und entsprechend der formalen Repräsentation transformieren zu lassen.

Die Transformation beruht auf einem *allgemeinen Algorithmus zur Transformation* und einer *generalisierten Repräsentation* von Text und Werten der Elemente.

Allgemeine Transformation
Die allgemeine Transformation traversiert rekursiv einen DOM Baum und übersetzt dessen Struktur in die formale Repräsentation. Der Algorithmus ist in pseudo-code in Abbildung 2 dargestellt.

```
syntactic_transformation(DOM)
{
    if NODE.DOCUMENT_NODE:
        logic_sentences ← document(ID).
        transform_structure(NODE.getDocumentElement)
    if NODE.ELEMENT_NODE:
        logic_sentences ← con-
        tains(parentNode, ID).
        CLASS = generali-
        ze_weak(NODE.NAME)
        logic_sentences ← structure(ID,
        CLASS).
        Attribute = NODE.getAttributes()
        for (i=0; i<Attribute.length();
        i++) {
          Attribute[i] = generali-
          ze_weak(Attribute[i])
          logic_sentences ← attribu-
          te(NODE,Attribute[i])
          if (Attribute[i] has Value) {
           Values = Attribu-
           te[i].getValues();
           for (u=0;
           u<Values.length(); u++)
           {
            Values[u] = generali-
            ze_weak(Values[u])
            logic_sentences ← va-
            lue(Attribute[i], Va-
            lues[u]).
           }
          }
        }
        forall Node.Childs do trans-
        form_structure(Child)
    if NODE.TEXT_NODE:
        logic_sentences ← generali-
        ze_strong(Text)
}
```

Abbildung 2: Algorithmus syntaktische Transformation

Der Algorithmus beschreibt generell das Traversieren eines Dokumentes und die Durchführung von Vorverarbeitungsschritten auf die einzelnen Attribute des Dokumentes. Das Ergebnis stellt eine Menge von logischen Ausdrücken dar, die das Dokument repräsentieren.

Vorverarbeitung von Text
Bei der Darstellung von Konstanten in den Prädikaten wird folgende Methode zur Repräsentation verwendet: Für *Struktur- bzw. Elementnamen* und *Werte von Elementen* wird jeweils eine **abgeschwächte Repräsentation** durchgeführt. Dies bedeutet eine Darstellung der Konstanten in Kleinbuchstaben. Bei Instanzen von *Text-Strukturen* wird eine **verstärkte Abschwächung** durchgeführt, was zusätzlich die Entfernung von Sonderzeichen (festgelegte Menge) beinhaltet.

`Willkommen auf meiner Homepage!!!`

wird zu

`willkommen auf meiner homepage`

abgeschwächt. Dadurch soll eine erhöhte Klassifizierungsrate für Instanzen von Text-Strukturen erreicht werden. Andernfalls wäre beispielsweise die Konstante `Homepage` ungleich der Konstante `homepage` und würde unter Umständen als *irrelevante Eigenschaft* der Beispiele nicht beachtet werden. Diese Abschwächung wird auf der Annahme durchgeführt, dass syntaktisch ähnliche Worte (Gross- Kleinbuchstaben und Worte mit/ohne Sonderzeichen) eine Instanz eines abstrahierten syntaktischen Konstruktes sind.

Das *eigentliche* Wort bleibt durch die Abschwächung bestehen. Jedoch werden bestimmte Namen oder Wortmarken bei einer starken Generalisierung, in denen Sonderzeichen vorkommen, ebenfalls in der abstrakten Form dargestellt. Dies führt dazu, dass Eigenschaften für diese Sonderfälle ebenfalls in der abstrakten Darstellung gelernt werden. Für eine geschlossene Menge von Dokumenten trifft dies auf alle Instanzen zu und lässt sich somit ohne Veränderung weiter verwenden. Zur Anwendung der Klassifizierungsregel auf ungesehene Beispiele kann zusätzlich Hintergrundwissen definiert werden, welches Regeln für die ursprüngliche Repräsentation bereitstellt oder ungesehene Dokumente nach dem beschriebenen Verfahren transformiert.

Der Text immer als eine Menge von einzelnen Worten betrachtet. Als Trennsymbol der Elemente in einer Textmenge wird das Leerzeichen verwendet. Folgen von Leerzeichen werden auf ein einzelnes Zeichen reduziert, wobei eine Menge mit nur einem Leerzeichen nicht zulässig ist. Die leere Menge, die durch eine Transformation aufgrund der Generalisierung entstehen kann, wird nicht beachtet. Für eine Instanz einer Text-Struktur mit n Elementen (nach obiger Definition) ergibt das n Prädikate in der formalen Repräsentation. Das mehrfache Auftreten von Prädikaten der Form `text_value(ID, K)` mit einer identischen Konstante K ist erlaubt. Dieser Fall kann durch syntaktisch gleiche Worte im (Ursprungs-) Text oder durch die *abgeschwächte Repräsentation* entstehen.

Die allgemeine syntaktische Transformation liegt allen generierten logischen Aussagen zugrunde. Ausnahmen bestehen bei der Behandlung bestimmter HTML Strukturen. Die jeweils verwendete Transformation wird im folgenden Abschnitt gesondert beschrieben.

4 Formale Repräsentation der Dokumente

Bei unbekannten Dokumenten stellt die (semi-) automatische Regelerzeugung eine multiple Anwendung der *syntaktischen Transformation* von Instanzen aus der Menge aller Dokumente dar. Anschliessend werden die *formalen Beschreibungen* zusammen mit der Beschreibung des Hypothesenraumes und dem ggf. vorhandenem Hintergrundwissen dem ILP System als

Eingabe zur Verfügung gestellt.

Bei bekannten Strukturen, in dieser Arbeit auf HTML Dokumente beschränkt, wird ein spezielles Verfahren zur syntaktischen Transformation ausgewählter Strukturen vorgestellt. Dadurch werden nur bestimmte Strukturen beachtet, wodurch eine kleinere Menge an Prädikaten entsteht. Die einzelnen Strukturen werden im Folgenden vorgestellt und deren Verarbeitung erläutert. Die aufgeführten Beispiele sind Experimenten mit Dokumenten aus dem Bereich der theoretischen Informatik der Universität Bremen entnommen, welche in [Hartmann, 2002] durchgeführt wurden.

4.1 Repräsentation spezieller Dokumentstrukturen

Die Deklaration einer festgelegten Strukturmenge basiert auf der Annahme, dass bestimmte Strukturen[4] in Dokumenten häufiger vorkommen und relevante Informationen zur Klassifikation beinhalten können. Daraus folgt ein erhöhtes Potential zur Klassifizierung dieser Strukturen. Die jeweilige Repräsentation dieser Eigenschaften in formalen Ausdrücken wird im Folgenden beschrieben.

Repräsentation von Dokumententiteln

Die Verwendung von Dokumententiteln beruht auf der Annahme, dass das *title* Element ein erhöhtes Potential (gegenüber anderer HTML Konstrukte) zur Klassifizierung von Dokumenten bietet. Experimente in [Pierre, 2000] haben ergeben, dass 89 Prozent der untersuchten Web Dokumente das *Title* Tag verwenden, wobei ein Titel durchschnittlich ein bis zehn Worte enthält.

Beispiel: Der HTML Titel T eines Dokumentes lautet exemplarisch:

```
<title>Publications - Research Group
Theoretical Computer Science</title>
```

Für das Lernen von Klassifikationsregeln, basierend auf *Dokumententiteln* wird das binäre Prädikat $doctitle(D, T_i)$ definiert. Wobei für *doctitle* gelten muss

$$doctitle(D, T_i) \leftarrow descendant(D, Q) \wedge structure(Q, 'title') \wedge contains(Q, W) \wedge text\_value(W, T_i)$$

Wobei gilt

$$descendant(A, B) \leftarrow contains(A, B)$$
$$descendant(A, C) \leftarrow contains(A, B) \wedge descendant(B, C)$$

Zur Erhöhung der potentiellen Klassifizierungsrate wird eine *starke Abschwächung* auf T angewendet. Die Menge T_{SG} ist das Ergebnis der starken Abschwächung von T. Das Prädikat *doctitle* beschreibt somit sub-Titel $T_i \in T_{SG}$ und wird durch `D` eindeutig referenziert.

$$T_{SG} = \{publications, research, group, theoretical, computer, science\}$$

Die formale Repräsentation von T_{SG} lautet

```
doctitle(doc135_obj0, 'publications').
doctitle(doc135_obj0, 'research').
doctitle(doc135_obj0, 'group').
doctitle(doc135_obj0, 'theoretical').
doctitle(doc135_obj0, 'computer').
doctitle(doc135_obj0, 'science').
```

Die Reihenfolge der einzelnen Elemente bleibt dabei unbeachtet. Die somit entstehende Abstraktion dient zur Erhöhung der Erkennungsrate von weiteren sub- Titeln. Daher können Klassifikationsregeln, wie im folgenden Ergebnis gezeigt, gelernt werden, die einzelne Prädikate *doctitle* aus der Menge T_{SG} enthalten.

Progol[Muggleton, 1995] liefert folgende Regel für das o.g. Beispiel:

```
document(A) :- doctitle(A,research).
```

Die gelernte Regel klassifiziert die Beispiele im Test Set zu 100 Prozent. Eine Erweiterung des Ansatzes auf die Menge der Überschriften in einem Dokument ist denkbar. Jedoch kann dies zu einer grossen Menge von Prädikaten im DataSet führen, was die gesonderte Betrachtung zur Reduktion der Prädikatenmenge hinfällig machen würde.

Repräsentation von e-Mail Adressen

Die Verwendung von e-Mail Adressen stellt ein weiteres Instrument zur Erkennung genereller Strukturen in Web-Dokumenten dar. Die definierte Syntax und eine relativ geringe Datenmenge in Bezug auf das Potential zur Klassifizierung bieten die Möglichkeit gemeinsame Eigenschaften effizient zu erkennen.

Die formale Repräsentation wird durch das Prädikat *mail* wie folgt definiert.

$$mail(D, Q, W)$$

Besteht die Menge der zu untersuchenden Dokumente aus Dokumenten einer Domain, so kann zwischen *internen* und *externen e-Mail Adressen* unterschieden werden. Interne e-Mail Adressen einer Domain *Dom* werden alle e-Mail Adressen genannt, die als Zieldomain *Dom* enthalten. Alle anderen e-Mail Adressen werden externe e-Mail Adressen genannt. Diese Unterscheidung erfolgt in der Absicht, zum einen Aussagen über eine geschlossene Menge von Dokumenten unabhängig ihrer Relationen zu weiteren Instanzen (anderer Klassen) treffen zu können und zum anderen, um nur die Relationen zu externen Instanzen analysieren zu können. Eine Kombination ergibt die komplette Menge an e-Mail Adressen einer Dokumentenmenge.

Für die differenzierte Betrachtung von e-Mail Adressen wird eine e-Mail Adresse in zwei Teilen dargestellt. Zum einen wird zwischen dem e-Mail Postfach (hier Q) und zum anderen zwischen der Domain (hier W) unterschieden, z.B.

$$\underbrace{hartmann}_{Q} @ \underbrace{aifb.uni-karlsruhe.de}_{W}$$

Für *mail* gilt daher

$$mail(D, Q, W) \leftarrow descendant(D, S) \wedge structure(S, 'a') \wedge attribute(S, R) \wedge$$

[4] Insbesondere Strukturen die laut Definition vorhanden sein müssen und Elemente, die mit hoher Wahrscheinlichkeit relevante Klassifizierungsdaten enthalten können.

$$structure(R,'href') \wedge$$
$$value(R,'mailto:' + Q + '@' + W)$$

Es sei angemerkt, dass das an dieser Stelle verwendete Prädikat `value(...)` keine korrekte Schreibweise darstellt, es dient nur der abstrakten Darstellung. Die Software führt auf den gesamten Link mehrere Extraktionen und Umformungen durch, so dass Q und W gewonnen werden können.

Zusätzlich zu der eigentlichen e-Mail Adresse können Attribute mit Wertzuweisungen deklariert werden, exemplarisch wird im Folgenden dem Attribut *Betreff* (subject) der Wert `test` zugewiesen.

```
hartmann@aifb.uni-karlsruhe.de?subject=test
```

Progol liefert bspw. folgende Regel:

```
document(A) :- relation(A,B), relation(B,C),
mail(C,helga,'informatik.uni-bremen.de').
```

Repräsentation von Metadaten

Eine weitere syntaktische Struktur, die in HTML Dokumenten auftreten kann, sind Metadaten. Deren Relevanz zur Klassifizierung wurde unter anderem im Rahmen der Arbeit von [Ghani *et al.*, 2001a] untersucht. Metadaten können in HTML Dokumenten durch sogenannte *Meta-Tags* spezifiziert werden [Consortium, 1999]. Sie können allgemeine Informationen über das Dokument, den Inhalt und notwendige Verarbeitungshinweise des Dokumentes beinhalten.

Beispiel[5]:

```
<META http-equiv="content-type"
content="text/html"; charset=iso-8859-1>
<META http-equiv="Content-Script-Type"
content="text/javascript">
<META http-equiv="Content-Style-Type"
content="text/css">
<META name="description" content="Research
Group Theoretical Computer Science">
<META name="keywords" content="theoretical, computer
science, University of Bremen, Kreowski,
publications" lang="en">
<META name="robots" content="follow">
<META name="revisit-after" content="10 days">
<META http-equiv="expires" content="0">
```

Ein Meta-Tag kann allgemein als Struktur mit einem Elternknoten K und zwei Attributen A_1, A_2 im Graphenmodell dargestellt werden. A_1 wird als Menge aller Instanzen der Attributklasse `http-equiv` und `name` angesehen. A_2 enthält Instanzen der Attributklasse `content`.

Für die formale Repräsentation von Meta-Tags wird das Prädikat

$$metatag(I, N, C)$$

definiert. Für die Konstante $N \in metatag$ gilt $attribute(I,Q) \wedge value(Q,N)$. Für die Konstante C gilt $attribute(I,W) \wedge value(W,C)$. Daraus folgt

$$metatag(I,N,C) \leftarrow descendant(D,I) \wedge structure(I,meta) \wedge attribute(I,Q) \wedge attribute(I,W) \wedge structure(Q,x) \wedge value(Q,N) \wedge structure(W,y) \wedge value(W,C)$$

[5] http://www.tzi.de/theorie/public.html, zugegriffen am 18.12.2001

wobei $x \in A_1$, $y \in A_2$ und I eine interne ID ist.

Die Repräsentation der Attributwerte N und C erfolgt in der vorgestellten *abgeschwächten Repräsentation*. Daraus folgt, dass jedes Prädikat $metatag_{ID}$ die Relation **eines** Elementes aus A_1 zu je **einem** Element aus der Menge von A_2 darstellt. Die Anzahl aller (Metadaten-) Prädikate eines Dokumentes besteht aus der Summe

$$\sum_{i=1}^{n} \mid p_i \mid * \mid v_i \mid \quad (1)$$

wobei n die Anzahl der MetaTags in einem Dokument, p die Menge der Werte aus A_1 (i.d.R. p=1) und v die Menge der Werte aus A_2 ist. Falls $\mid A_2 \mid = 0$ dann gilt

$$metatag(I, N).$$

Eine syntaktische Transformation des oben angeführten Beispiels ergibt:

```
metatag(doc135_obj0, 'content-type', 'text/html').
metatag(doc135_obj0, 'content-script-type',
'text/javascript').
metatag(doc135_obj0, 'content-style-type', 'text/css').
metatag(doc135_obj0, 'description', 'research').
metatag(doc135_obj0, 'description', 'group').
metatag(doc135_obj0, 'description', 'theoretical').
metatag(doc135_obj0, 'description', 'computer').
metatag(doc135_obj0, 'description', 'science').
metatag(doc135_obj0, 'keywords', 'theoretical').
metatag(doc135_obj0, 'keywords', 'computer').
metatag(doc135_obj0, 'keywords', 'science').
[...]
```

Metadaten sind ein umfangreiches Instrument, um Informationen über *Syntax* und *Semantik* in Dokumenten zu hinterlegen. In Bezug auf die Klassifizierung von Dokumenten bedeutet dies eine geringe Datenmenge mit einem erhöhten Potential zur Klassifizierung.

Folgende Regel konnte aus dem o.g. Beispiel gewonnen werden, welche zu 100 Prozent die Trainigsdaten beschreibt:

```
document(A) :- metatag(A,keywords,theoretical).
```

4.2 Repräsentation nicht spezifizierter Dokumentenstrukturen

Die HTML Spezifikation [Consortium, 1999] bietet vordefinierte syntaktische Konstrukte, um HTML Dokumente zu generieren. Eine Grundstruktur ist dabei zwingend vorgegeben. In Fällen, in denen keine HTML Dokumente, sondern XML [Consortium, 2000b] Dokumente vorliegen, können die bisher vorgestellten Mechanismen nicht oder nur mit Einschränkungen übernommen werden. Ein gezieltes *Suchen* nach bestimmten syntaktsichen Strukturen, mit denen bereits eine Semantik verbunden wird (wie bei HTML Dokumenten), kann bei XML Dokumenten (aufgrund der Möglichkeit zur freien Definition struktureller Elemente) nicht durchgeführt werden.

Eine allgemeine Vorgehensweise zur Erkennung genereller Eigenschaften bei unbekannten Strukturen beruht auf einer kompletten syntaktischen Transformation der zu untersuchenden Dokumente. Dies bedeutet, dass alle syntaktischen Strukturen innerhalb der Dokumente in die formale Repräsentation transformiert

werden. Dadurch kann eine grosse Menge von Prädikaten entstehen und vor allem eine grosse Anzahl an Relationen (Elemente eines Dokumentes stehen in einer bestimmten Relation zueinander).

Exemplarisch wurden Protokolle eines studentischen Projektes[6] verwendet, um einen gemeinsamen Klassifikator zu lernen. Zu den XML Dokumenten lag kein weiteres Hintergrundwissen vor und somit wurde die gesamte Struktur transformiert. Progol liefert folgende Regel für die Trainingsbeispiele als Ergebnis:

```
document(A) :- contains(A,B),
structure(B,protocol).
```

Durch die syntaktische Transformation der gesamten Struktur der XML Dokumente lassen sich demnach Klassifikationsregeln lernen, wodurch die Möglichkeit zur Verabeitung *syntaktisch* unbekannter XML Dokumente gegeben ist. Jedoch ist die semantische Aussagefähigkeit einer solchen Regel unklar.

Die Transformation der gesamten Struktur kann jedoch zu grossen Mengen von Prädikaten[7] führen. Dies hat eine entsprechende Vergrösserung des Hypothesenraumes zur Folge. Die Suche im Hypothesenraum kann in diesem Fall durch Einschränkungen (search bias) durch den Lehrer verringert werden. Alternativ kann in diesen Fällen eine Aufteilung grosser Trainingsmengen in mehrere kleinere Mengen durchgeführt werden. Die Hypothese lässt sich dann inkrementell durch ein ILP System generieren.

4.3 Repräsentation von Dokumentrelationen

Das Lernen von Relationen wird benutzt, um generelle Informationen über Beziehungen zwischen Dokumenten zu erhalten. Dadurch lassen sich, im Gegensatz zu reinen Attribut-Werte Darstellungen, ausdrucksstärkere Regeln repräsentieren.

Relationen werden in HTML Dokumenten durch Links deklariert [Consortium, 1999]. Diese können als gerichtete Graphen aufgefasst werden [Kleinberg *et al.*, 1999], wie exemplarisch in Abbildung 3 dargestellt. Die Abbildung beschreibt eine mögliche relationale Struktur von Dokumenten. Wir unterscheiden zwischen Relationen innerhalb der zu untersuchenden Dokumentenmenge (DataSet) und Relationen zu externen Dokumenten. Diese Unterscheidung kann sinnvoll sein, wenn Eigenschaften einer geschlossenen Menge (Dokumente einer Domain beispielsweise) beschrieben werden sollen. Analog dazu kann die gesonderte Betrachtung von Relationen zu Instanzen anderer Klassen betrachtet werden.

Die allgemeine Darstellung von Relationen wird wie folgt angegeben:

$$relation(D_1, D_2)$$

Der Ausdruck beschreibt eine Relation von D_1 zu D_2, wobei D_1 und D_2 Dokumente sind. Die allgemeine Beziehung zweier Dokumente wird wie folgt definiert.

$$relation(D_1, D_2) \leftarrow descendant(D_1, Q) \wedge$$

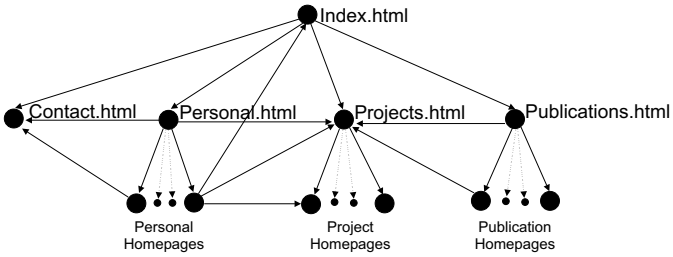

Abbildung 3: Exemplarische Struktur einer Web Domain als Graph

$$structure(Q, a) \wedge attribute(Q, W) \wedge$$
$$structure(W, href) \wedge value(W, Z) \wedge$$
$$url(D_2, Z)$$

In einem Experiment konnten folgende Regeln für die Trainingsbeispiele erzeugt werden[8]:

```
document(A) :- relationto(A,
'http://www.tzi.de/tzi/PROJEKTE/projekte.html').
document(A) :- relationto(A,
'http://www.tzi.de/tzi/projekte.html').
document(A) :- relationto(A,
'http://www.tzi.de/tzi/Projekte_neu').
```

Das Lernen relationaler Strukturen zwischen Dokumenten kann eingesetzt werden, um Dokumente zu klassifizieren. Dabei können Strukturen in den Dokumenten (abgesehen von der Relation selbst) unbeachtet bleiben. Dies kann in strukturell heterogenen Mengen von Vorteil sein, da in zu stark heterogenen Mengen eine Klassifizierungregel womöglich nicht generiert werden kann.

Zur weiteren Differenzierung wird im Folgenden genauer auf interne und externe Relationen eingegangen.

Als *interne Relationen* werden alle Beziehungen von Dokumenten innerhalb der Menge der zu untersuchenden Dokumente (DataSet) beschrieben. Interne Relationen können gesondert betrachtet werden, um nur Informationen über die Beziehungen der Dokumente eines DataSets zu erhalten. Beispielsweise können Regeln verifiziert werden, die für jedes Dokument eine Relation zu einem anderen Dokument (z.B. einer Startseite einer Internetpräsenz) fordert.

Alle internen Dokumente werden durch das Prädikat `document(...)` deklariert, dadurch lässt sich ein Prädikat für interne Relationen, im Folgenden `internal_relation` genannt, wie folgt deklarieren

$$internal\_relation(A, B) \leftarrow relation(A, B) \wedge$$
$$document(A) \wedge document(B)$$

Als *externe Relationen* werden alle Beziehungen von Dokumenten aus der Menge der zu untersuchenden Dokumente zu externen Dokumenten (anderer Klassen) beschrieben. So lassen sich beispielsweise Relationen zu anderen Instanzen von Klassen entdecken oder verifizieren.

[6] Projekt ViVa Libertas, Universität Bremen

[7] entsprechend der Anzahl der Strukturbezeichner und deren Relationen in den Dokumenten

[8] Hier wird das Prädikat `relationto` angegeben, weil `relation(A,B)` nur eine Beziehung zwischen den Dokumenten A und B beschreibt. Für eine Verwendung als Regel ist jedoch die URL von B von Interesse. Dafür wurde Hintergrundwissen definiert, welches die URL von B deklariert: `relationto(A,U):-relation(A,B), url(B,U).`

Alle externen Dokumente werden durch das Prädikat `externaldoc(...)` deklariert, wodurch sich das Prädikat für Relationen von einem internen Dokument zu einem **externen Dokument**, im Folgenden `externeal_relation` genannt, beschreiben lässt:

$$external\_relation(A, B) \leftarrow relation(A, B) \wedge document(A) \wedge externaldoc(B)$$

Neben dem höheren Informationsgehalt und der Möglichkeit zur gezielten Analyse von Dokumentenmengen, ist eine derartige Unterscheidung notwendig, um korrekte relationale Regeln lernen zu können. Denn das ILP System hat in dieser Arbeit nur Zugriff auf das Data Set und kann dementsprechend nur Attribute von Dokumenten innerhalb des Data Sets analysieren. Das Lernen relationaler Regeln würde auch ohne Unterscheidung funktionieren, jedoch bedeutet dies aufgrund der *Negation as failure* Strategie, das beispielsweise ein externes Dokument keinen Dokumententitel, keine Metadaten etc. besitzt. Diese Aussage kann jedoch nicht einfach getroffen werden, da diese Informationen nicht bekannt sind.

5 Durchgeführte Experimente

Im Rahmen der durchgeführten Experimente wurde das entwickelte Verfahren auf *Umwelt-Informations-Systeme (UIS)* angewendet. Angelehnt an die EG Richtlinie[9] von 1990 wurde am 08.04.1994 das Umweltinformationsgesetz (UIG) in der BRD verabschiedet. Das UIG regelt einen freien Zugang zu Umweltinformationen in den Behörden. Nach diesem Gesetz ist eine Behörde zur Veröffentlichung von Umweltinformationen verpflichtet. Ein Grossteil der Behörden setzt dabei auf die Entwicklung eines UIS, welche teilweise durch eine Anbindung an das WWW frei zugänglich sind. Dadurch ist es möglich, eine grosse Menge von strukturierten Informationen und Daten in Form von Web Dokumenten für die Experimente zu nutzen.

[W. Roggendorf, 1995] beschreibt detailliert notwendige Entwicklungsschritte und Bestandteile eines UIS. Für eine Übersicht über bestehende UIS sei auf [Schliep, 2001] verwiesen.

Für die Experimente wurden folgende drei UIS benutzt.

- Bremer Umwelt Informations System (BUISY)
  `http://www.umwelt.bremen.de/buisy`
- Informationssystem des Umweltbundesamtes in Wien (UBAVIE)
  `http://www.ubavie.gv.at`
- UIS Bayern
  `http://www.umweltministerium.bayern.de`

Diese UIS weisen eine vergleichbare Strukturierung der Informationen auf und enthalten eine vergleichbare Anzahl von Instanzen in den Kategorien. Für die Experimente wurden folgende Kategorien dieser UIS verwendet: `Abfall, Boden, Luft, Natur` und `Wasser`.

Alle Dokumente wurden mittels `wget`[10] unter Linux lokal gespeichert und entsprechend dem vorgestellten Verfahren bearbeitet.

[9] Richtlinie über den freien Zugang zu Informationen über die Umwelt (90/313/EWG)

[10] wget -m -np -Ahtm,html -http://[UIS]

5.1 Durchführung

Für die Generierung von Klassifikatoren für HTML Dokumente wurde im Abschnitt 2 eine Menge von Prädikaten vorgestellt. Die Experimente dienen daher der Überprüfung dieser Menge, auf ihr Potential zur Klassifikation von Dokumenten. Anschliessend werden Klassifikationsregeln für einzelne Kategorien in den UIS gelernt und die Ergebnisse aufgezeigt.

Zur Generierung von Klassifikationsregeln wird das ILP System Progol [Muggleton, 1995] verwendet. Der **Progol** Ansatz von S. Muggleton basiert auf *inverse Entailment* als Refinement Operator, um die speziellste Hypothese zu generieren. Die Suche nach der Hypothese, die die Beispiele und das Hintergrundwissen am *besten* erklärt ist eine *general-to-specific* Suche, ähnlich der in FOIL [Quinlan and Cameron-Jones, 1993], durch einen *refinement Graph*. Progol bietet die Möglichkeit ausschliesslich von positiven Beispielen zu lernen [Muggleton, 1996]. Die Suche durch den Hypothesenraum kann durch die Definition von **Mode Declarations** eingeschränkt werden, welche durch den Lehrer definiert werden müssen. Dabei kann die Verwendung bestimmter Prädikate und deren Häufigkeit beschränkt werden. Weiter können Abbruchkriterien definiert werden, um die Suche zu beschleunigen. Für eine detaillierte Beschreibung sei auf [Muggleton, 1995] verwiesen.

Für jede Kategorie eines UIS wurden insgesamt fünf Durchläufe durchgeführt. In den Durchläufen eins bis vier wurden die verwendeten **Modeb Declarations** entsprechend der Tabelle 1[11] beschränkt. Im fünften Durchlauf sind alle Modeb Declarations (MD 1- 4) erlaubt und die generierte Hypothese stellt somit das Gesamtergebnis der jeweiligen Kategorie dar. Die Durchläufe eins bis vier dienen dabei zur Verifikation der definierten Prädikatenmenge aus dem Abschnitt 2.

Mode(b) Declarations		
MD	*Prädikate*	*Zusatz*
1.	$doctitle(D, T_i)$	
2.	$metatag(I, N, C)$ $metatag(I, N)$	
3.	$mail(D, Q, W)$ $mailto\_domain(D, W)$ $mailto\_postbox(D, Q)$	 mailto_domain(D,W) :- mail(D,Q,W). mailto_postbox(D,Q) :- mail(D,Q,W).
4.	$url(A, B)$	

Tabelle 1: Modeb Declarations der Experimente

Das Prädikat `document(+object)` ist dabei immer der *head* der zu erstellenden Hypothese (Modeh Declaration). Zusätzlich wird in jedem Durchlauf eine Modeb Declaration für das Prädikat

$$relation(D_1, D_2)$$

angegeben, um relationale Klassifikatoren lernen zu können .

Auf die erstellten TrainingSets werden diese Mode Declarations angewendet und die Ergebnisse für jeden Durchlauf ausgewertet. Dadurch soll das Klassifikationspotential der einzelnen Prädikate untersucht werden und das Gesamtergebnis für das wissensbasierte Dokumentenmanagement in Spectacle verwendet werden.

[11] Eine Zeile der Tabelle beschreibt dabei einen Durchlauf.

Allen Experimenten liegt ein Verhältnis 1:2 von positiven zu negativen Beispielen zugrunde, d.h. jedem positiven Beispiel werden genau zwei negative Beispiele zugeordnet.

Für das Verhältnis der Beispiele in den Training Sets zu den Beispielen in den Test Sets wird ein Verhältnis von 70:30 verwendet, d.h. 70% der positiven Beispiele befinden sich in den Training Sets. Die verbleibenden 30% werden im Test Set verwendet.

Die *Berechnung der Accuracy* basiert auf folgenden vier Klassifikationsergebnissen:

1. *P(A)*: Die Menge der (tatsächlich) positiven Beispiele, die durch die Hypothese als positive Beispiele klassifiziert wurden.
2. $\neg P(A)$: Die Menge der (tatsächlich) positiven Beispiele, die durch die Hypothese als negativ klassifiziert wurden.
3. $\neg P(\neg A)$: Die Menge der (tatsächlich) negativen Beispiele, die durch die Hypothese als negativ klassifiziert wurden.
4. $P(\neg A)$: Die Menge der (tatsächlich) negativen Beispiele, die durch die Hypothese als positiv klassifiziert wurden.

Die Accuracy beschreibt das Verhältnis aller korrekt klassifizierten Beispiele zu der Menge aller vorhandenen Beispiele in einem Data-Set, sie wird wie folgt berechnet.

$$Acc. = \frac{P(A) + \neg P(\neg A)}{P(A) + \neg P(A) + \neg P(\neg A) + P(\neg A)} * 100$$

In Fällen, in denen keine Hypothese aus den Training Sets generiert werden konnte, wird jeweils ein '-' in den Tabellen angegeben.

5.2 Verifikation der Prädikatenmenge

Im Folgenden werden die definierten Prädikate aus Abschnitt 2 in Bezug auf ihre Klassifikationsrate untersucht und ausgewertet. Für die Bewertung des Potentials der einzelnen Prädikate wird die erreichte Klassifikationsrate auf das jeweilige Test Set angegeben. Für eine detaillierte Evaluierung der Prädikate sei auf [Hartmann, 2002] verwiesen. Es konnte gezeigt werden, dass die Prädikate $doctitle(D, T_i)$ und $metatag(I, N, C)$ noch ausreichende Klassifikationsraten erzielen können. Im Gegensatz dazu haben jedoch die Prädikate $mail(D, Q, W)$ und $url(A, B)$ schlechte Ergebnisse erzielt. Hierbei sei anzumerken, dass teilweise sehr unterschiedliche Klassifikationsraten zwischen den getesteten Systemen erzielt worden sind, welches auf eine unterschiedliche Struktur der Dokumente zurückzuführen ist. Dazu zählen insbesondere die Ergebnisse des *mail* Prädikates. Die einzelnen Ergebnisse sind in Tabelle 2 aufgeführt.

Prädikat	BUISY	Ubavie	Bayern	Gesamt Accuracy
$doctitle(D, T_i)$	89,4	99,78	72,83	87,34
$metatag(I, N, C)$	98,53	100	58,67	85,73
$mail(D, Q, W)$	-	98,63	20	39,54
$url(A, B)$	13,81	99,61	93,03	68,82

Tabelle 2: Gesamtauswertung nach einzelnen Prädikaten

5.3 Ergebnisse

Im folgenden Abschnitt werden die Ergebnisse der Anwendung auf die drei verwendeten Umwelt-Informations-Systeme beschrieben.

Das BUISY System
Das Bremer Umwelt-Informations-System (BUISY) stellt ein Umwelt-Informations-System für die Öffentlichkeit dar und dient der Behörde zur internen Kommunikation und Koordinierung von Arbeitsprozessen.

Ausgenommen von der Kategorie *Natur* konnten für alle Kategorien Klassifikationsregeln gelernt werden, die keine Missklassfikationen enthalten (siehe Tabelle 3).

Umweltbundesamt Wien
Das Wiener Gesamtergebnis (Tabelle 4) zeigt das hohe Potential und die Bedeutung von Metadaten in den Dokumenten. Eine einfache Attribut Werte Paar Repräsentation reicht in diesem Fall aus, um fünf Kategorien in einem UIS ohne Fehler zu klassifizieren.

UIS Bayern
Das UIS Bayern enthielt eine vergleichsweise hohe Anzahl von *heterogen* strukturierten Dokumenten, welches sich in den generierten Regeln niederschlägt. Dennoch konnten gute Klassifikationsraten erzielt werden (siehe Tabelle 5).

5.4 Anwendung im wissensbasierten Content-Management

Ein Verfahren zum *wissensbasierten Content-Management* wurde von der Freien Universität Amsterdam (Vrije Universiteit) und der niederländischen Unternehmung AIdministrator[12] mit dem Tool *Spectacle* [van Harmelen and van der Meer, 1999] vorgestellt. Der Ansatz verwendet Ontologien zur Beschreibung von Web-Seiten, die zur Verifikation der Dokumente eingesetzt werden. Der Schwerpunkt des Ansatzes liegt in der Administration von Web-Dokumenten. Die einzige Möglichkeit, Regeln in Spectacle zu deklarieren, besteht in der Benutzung einer graphischen Benutzerschnittstelle. Die Deklaration der Regeln, die Integritätszustände von Dokumenten beschreiben, setzt syntaktisches und unter Umständen auch semantisches Domänenwissen des Anwenders voraus. Mit dem vorgestellten Verfahren lassen sich nun ähnlich ausdrucksstarke Regeln semi-automatisch lernen, wodurch eine manuelle Deklaration hinfällig wird.

Die Anwendung der gelernten Regeln in Spectacle wird in diesem Abschnitt exemplarisch am Wiener Umweltbundesamt demonstriert. Dazu wurden in Spectacle die ersten vier Verzeichnisebenen (insgesammt 629 Dokumente) des UIS Wien und die erzeugten Regeln (siehe Tabelle 4) verwendet.

Die Anwendung wird dabei analog zu dem beschriebenen Verfahren durchgeführt. Im ersten Schritt wurden die Kategorien in Spectacle definiert (siehe Abbildung 5).

Anschliessend erfolgte die Definition der Regeln in der Spectacle GUI. Dieser Schritt ist nun ohne spezielles Hintergrundwissen über die zu untersuchenden Dokumente durchführbar. Exemplarisch wird die Regel für die Kategorie Natur in Abbildung 6 gezeigt.

[12] http://www.aidminist rator.nl

BUISY Gesamtergebnis						
Kategorie	*Hypothese(n)*	*P(A)*	*¬P(A)*	*¬P(¬A)*	*P(¬A)*	*Acc.*
Abfall	document(A) :- metatag(A,bereich,abfall).	13	0	26	0	100
Boden	document(A) :- metatag(A,keywords,bodenschutz). document(A) :- metatag(A,keywords,boden).	11	0	22	0	100
Luft	document(A) :- metatag(A,expires,thu). document(A) :- metatag(A,bereich,luft).	28	0	56	0	100
Natur	document(A) :- doctitle(A,richtlinie). document(A) :- metatag(A,author,'zdl30-13'). document(A) :- metatag(A,bereich,naturschutz).	20	1	42	0	98,41
Wasser	document(A) :- metatag(A,bereich,wasser). document(A) :- doctitle(A,fuer), metatag(A,generator,microsoft).	58	0	116	0	100

Tabelle 3: Experiment BUISY Ergebnis

Ubavie Gesamtergebnis						
Kategorie	*Hypothese(n)*	*P(A)*	*¬P(A)*	*¬P(¬A)*	*P(¬A)*	*Acc.*
Abfall	document(A) :- metatag(A,keywords,abfall).	22	0	44	0	100
Boden	document(A) :- metatag(A,keywords,boden).	39	0	78	0	100
Luft	document(A) :- metatag(A,keywords,luft).	7	0	14	0	100
Natur	document(A) :- metatag(A,keywords,natur).	33	0	66	0	100
Wasser	document(A) :- metatag(A,keywords,wasser).	120	0	240	0	100

Tabelle 4: Experiment Ubavie Ergebnis

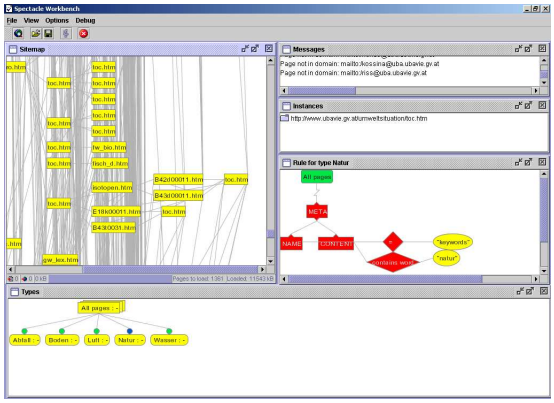

Abbildung 4: Anwendung in Spectacle

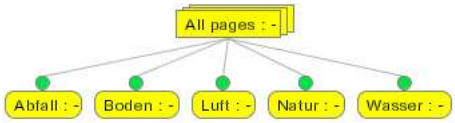

Abbildung 5: Typdeklaration in Spectacle

Das Ergebnis der Regelanwendung in Spectacle wird in Abbildung 7 gezeigt.

Zur Bewertung der Ergebnisse muss das Verhältnis der (tatsächlich) positiven Beispiele zu der Anzahl der positiv klassifizierten Beispiele bekannt sein. Deshalb wurde eine zusätzliche Regel in Spectacle für jede Kategorie definiert (mittlere Zeile in Abbildung 7), die alle (tatsächlich) positiven Beispiele beinhaltet[13]. Zuvor wurden die eigentlichen Regeln ohne diese Zusatzregel getestet. Das Ergebnis blieb unverändert. Eine Veränderung des Ergebnisses hätte eine Klassifikation von (tatsächlich) negativen Beispielen als positive Beispiele bedeutet. Wie dem Gesamtergebnis aus Abbildung 7 zu entnehmen ist, konnten ähnliche Klassifikationsraten, wie bei den lokalen Data Sets, erreicht werden.

[13]Diese Regel ist eine *Search Engine Rule*, welche alle Dateien in einem bestimmten Verzeichnis klassifiziert.

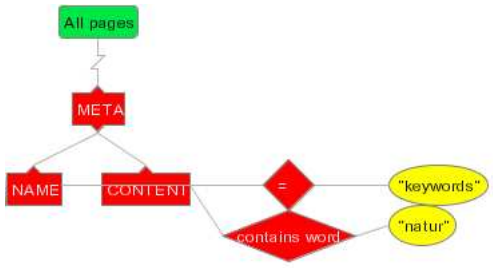

Abbildung 6: Regeldeklaration in Spectacle

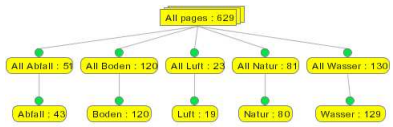

Abbildung 7: Ubavie Ergebnis in Spectacle

6 Zusammenfassung und Ausblick

In diesem Abschnitt wurde die Anwendung des entwickelten Verfahrens auf drei web- basierte Umwelt-Informations-Systeme beschrieben und die jeweiligen Ergebnisse ausgewertet. Im Durchschnitt konntent Klassifikationsraten von 98,41 Prozent erreicht werden. Eine detaillierte Auswertung der Ergebnisse ist in Tabelle 6 aufgeführt.

Die Erkennungsrate der Klassifikatoren innerhalb eines UIS ist nach den aufgezeigten Ergebnissen ausreichend. In Fällen, in denen schlechtere Raten vorliegen, kann neben einer möglichen Erhöhung der zu untersuchenden Dokumenteneigenschaften zusätzlich der Hypothesenraum in Progol auf die Negation erweitert werden. Dadurch können Regeln gelernt werden, die explizit Eigenschaften ausschliesen und somit Klassifikatoren ggf. verbessern. Es sei angemerkt, dass sich negierte Eigenschaften in Spectacle nicht repräsentieren lassen. Das präsentierte Verfahren lässt sich unabhängig von Spectacle verwenden, daher können ausdrucksstärkere Regeln für andere Anwendungen eingesetzt werden.

UIS-Bayern Gesamtergebnis						
Kategorie	*Hypothese(n)*	$P(A)$	$\neg P(A)$	$\neg P(\neg A)$	$P(\neg A)$	*Acc.*
Abfall	document(A) :- relation(A,A), relation(A,B), metatag(B,keywords,abfall).	6	0	12	0	100
Boden	document(A) :- relation(A,A), relation(A,B), metatag(B,keywords,bodenschutz).	7	0	14	0	100
Luft	document(A) :- relation(A,B), relation(A,C), metatag(C,keywords,luft).	4	1	10	0	93,33
Natur	document(A) :- relation(A,B), url(B,'[URL]/natur/baynetna/'). document(A) :- relation(A,B), url(B,'[URL]/natur/jahr/'). document(A) :- relation(A,B), url(B,'[URL]/natur/'). document(A) :- relation(A,B), url(B,'[URL]/natur/schutzge/ramsar/'). document(A) :- relation(A,B), url(B,'[URL]/natur/schutzge/').	9	8	34	0	84,31
Wasser	document(A) :- relation(A,B), url(B,'[URL]/wasser/bilder/'). document(A) :- relation(A,B), url(B,'[URL]/wasser/'). document(A) :- doctitle(A,unbenanntes), metatag(A,generator,win).	8	0	16	0	100
URL: http://www.umweltministerium.bayern.de/bereiche						

Tabelle 5: Experiment UIS-Bayern Ergebnis

Kategorie	$P(A) + \neg P(\neg A)$	$\neg P(A) + P(\neg A)$	Accuracy
BUISY - Abfall	39	0	100
BUISY - Boden	33	0	100
BUISY - Luft	84	0	100
BUISY - Natur	62	1	98,41
BUISY - Wasser	174	0	100
BUISY - Gesamt			99,68
Ubavie - Abfall	66	0	100
Ubavie - Boden	117	0	100
Ubavie - Luft	21	0	100
Ubavie - Natur	99	0	100
Ubavie - Wasser	360	0	100
Ubavie - Gesamt			100
Bayern - Abfall	18	0	100
Bayern - Boden	21	0	100
Bayern - Luft	14	1	93,33
Bayern - Natur	43	8	84,31
Bayern - Wasser	24	0	100
Bayern - Gesamt			95,53
Total			98,41

Tabelle 6: Gesamtauswertung der Experimente

Exemplarisch wurde die Software erweitert, so dass der Datensatz des $Web \rightarrow KB$ Projektes [Craven *et al.*, 2000] eingelesen werden kann[14]. Ergebnisse eines exemplarischen Lernprozesses wurden für Dokumente aus dem Bereich: *Veranstaltungsseite einer Universität (course)* durchgeführt. Für das Training-Set wurden Datensätze aus drei Universitäten verwendet und eine weitere Universität für das Test-Set ausgewählt. Dabei konnten Erkennungsraten von 64,29 und 64,44 Prozent erreicht werden. Jedoch stellt die gelernte Regel eine zu spezielle Regel dar (overfitting) und ist demnach nur bedingt für einen *open world* Einsatz zu verwenden.

Die Anwendbarkeit der Klassifikatoren in Spectacle konnte ebenfalls demonstriert werden. Die Annahme, dass Klassifikationsregeln für Web Dokumente semiautomatisch durch *die Extraktion von Struktureigenschaften* generiert werden können, konnte anhand einer Reihe von Experimenten im Bereich web-basierter Umwelt-Informations Systeme verifiziert werden. Dabei konnten Klassifikatoren mit einer durchschnittlichen Accuracy von 98,41 Prozent erzeugt werden (siehe Tabelle 6). Eine Überprüfung der Klassifikatoren durch Anwendung auf HTML Dokumente, welche nicht einem UIS angehören, wurden ebenfalls erfolgreich durchgeführt[15].

Eine Anwendung des Verfahrens auf *open World* Lernprobleme wurde exemplarisch an den WebKB Datensatz gezeigt. Die Ergebnisse machen jedoch weitere Arbeit in diesem Bereich notwendig. Insbesondere für den Bereich der Vorverarbeitung und des Lernprozesses werden weitere Methoden und Verfahren entwickelt werden müssen. Dies kann zu einer Entwicklung einer eigenen Lernstrategie führen, welche als seperate Schicht über dem ILP System.

Danksagungen

Ich möchte mich bei allen Mitarbeitern des AIFB und insbesondere bei Heiner Stuckenschmidt (VU Amsterdam) sowie bei Andreas Hotho für die gute und hilfreiche Unterstützung bedanken.

Literatur

[Benjamins and Fensel, 1998] R. Benjamins and D. Fensel. The ontological engineering initiative-ka, 1998.

[Benjamins *et al.*, 1998] V. Benjamins, D. Fensel, and A. Perez. Knowledge management through ontologies, 1998.

[Benjamins, 1998] V. Benjamins. Fensel: The ontological engineering initiative, 1998.

[Berners-Lee and Fischetti, 1999] T. Berners-Lee and M. Fischetti. Weaving the web: The original design and ultimate destiny of the world wide web by its inventor, 1999.

[Boley, 1996] Harold Boley. Knowledge bases in the world wide web: A challenge for logic programming (second, revised edition). Technical Report TM-96-02, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Germany, 1996.

[Boley, 2000] Harold Boley. Relationships between logic programming and xml. In *Workshop Logische Programmierung*, pages 19–34, 2000.

[14] Durch die notwendige Transformation der Dokumente in ein gültiges XHTML Format konnten ca. 1000 Dokumente von ca. 5300 Dokumente nicht transformiert werden.

[15] D.h. diese Dokumente wurden als negativ klassifiziert.

[Cabeza and Hermenegildo, 1997] D. Cabeza and M. Hermenegildo. Www programming using computational logic systems, 1997.

[Chakrabarti, 2000] Chakrabarti. Data mining for hypertext: A tutorial survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery Data Mining, ACM*, 1, 2000.

[Consortium, 1999] W3C World Wide Web Consortium. Html 4.01 specification, Dez. 1999.

[Consortium, 2000a] W3C World Wide Web Consortium. Document object model (dom) level 2 core specification, Nov. 2000.

[Consortium, 2000b] W3C World Wide Web Consortium. Extensible markup language (xml) 1.0 (second edition), Okt. 2000.

[Craven *et al.*, 2000] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigan, and Séan Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118(1–2):69–113, 2000.

[Davison and Loke, 1996] A. Davison and S. Loke. Logicweb: Enhancing the web with logic programming, 1996.

[Decker *et al.*, 1999] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology based access to distributed and semi-structured unformation. In *DS-8*, pages 351–369, 1999.

[Farquhar *et al.*, 1996] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: A tool for collaborative ontology construction, 1996.

[Ghani *et al.*, 2001a] R. Ghani, S. Slattery, and Y. Yang. Hypertext categorization using hyperlink patterns and meta data, 2001.

[Ghani *et al.*, 2001b] Rayid Ghani, Seán Slattery, and Yiming Yang. Hypertext categorization using hyperlink patterns and meta data. In Carla Brodley and Andrea Danyluk, editors, *Proceedings of ICML-01, 18th International Conference on Machine Learning*, pages 178–185, Williams College, US, 2001. Morgan Kaufmann Publishers, San Francisco, US.

[Guarino, 1998] N. Guarino. Formal ontology and information systems, 1998.

[Harmelen, 2000] Frank Van Harmelen. Ontology-based information visualisation, 2000.

[Hartmann and Stuckenschmidt, 2002] Jens Hartmann and Heiner Stuckenschmidt. Automatic Metadata Analysis for Environmental Information Systems. In *Proceedings of Environmental Informatics 2002*. Metropolis Verlag, Marburg (Germany), 2002.

[Hartmann, 2002] Jens Hartmann. Lernen struktureller Regeln zur Klassifikation von Web-Dokumenten. Master's thesis, University of Bremen (TZI), 2002.

[Heflin *et al.*, 1999] Jeff Heflin, James Hendler, and Sean Luke. SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078, 1999.

[Kleinberg *et al.*, 1999] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The Web as a graph: Measurements, models, and methods. In T. Asano, H. Imai, D. T. Lee, S. Nakano, and T. Tokuyama, editors, *Proc. 5th Annual Int. Conf. Computing and Combinatorics, COCOON*, number 1627. Springer-Verlag, 1999.

[Leighton and Srivastava, 1997] H. Leighton and J. Srivastava. Precision among www search services, 1997.

[Moore *et al.*, 1997] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher. Web page categorization and feature selection using association rule and principal component clustering, 1997.

[Muggleton, 1995] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[Muggleton, 1996] S. Muggleton. Learning from positive data. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pages 225–244. Stockholm University, Royal Institute of Technology, 1996.

[Pierre, 2000] J. Pierre. Practical issues for automated categorization of web pages, 2000.

[Pierre, 2001] J. Pierre. On automated classification of web sites, 2001.

[Quinlan and Cameron-Jones, 1993] J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In P. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning*, volume 667, pages 3–20. Springer-Verlag, 1993.

[Schliep, 2001] Rainer Schliep. Umwelt-informationssysteme, Jan. 2001.

[Stuckenschmidt *et al.*, 2002] Heiner Stuckenschmidt, Jens Hartmann, and Frank van Harmelen. Learning structural classification rules for web-page categorization. In *Proceedings of FLAIRS 2002, special track on Semantic Web*, 2002.

[Studer *et al.*, 1998] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1-2):161–197, 1998.

[van Harmelen and van der Meer, 1999] F. van Harmelen and J. van der Meer. Webmaster: Knowledge-based verification of web-pages. In I. Imam, Y. Kodratoff, and M. Ali, editors, *Twelfth International Conference on Industrial - Engineering Applications of Artificial Intelligence - Expert Systems IEA/AIE'99*, number 1611 in Lecture Notes in Artificial Intelligence, pages 256–265. Springer Verlag, 1999.

[W. Roggendorf, 1995] R. Stahl W. Roggendorf, F. Scholles. Ein leitfaden für den aufbau kommunaler umweltinformationssysteme, 1995.

[Yang *et al.*, 2002] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization, 2002.