Universität Karlsruhe (TH)

Studienarbeit

PAT-Trees zur Automatischen Informationsextraktion aus semi-strukturierten Webdaten

Student:	Ronny Händel
Betreuer:	Prof. Dr. A. Waibel
Betreuer:	Dipl.-Inform. H. Holzapfel
Tag der Abgabe:	31.01.2008

Wintersemester 2007/2008

Fakultät für Informatik
Institut für Theoretische Informatik (ITI)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Das World Wide Web ist inzwischen die größte und reichhaltigste frei zugängliche Informationsquelle geworden. Lediglich dessen Struktur macht es schwierig, diese Informationen auf systematischem Wege zu nutzen. In der vorliegenden Studienarbeit wird ein Algorithmus zur automatischen Generierung von Extraktionsmustern für semistrukturierte Websites eingesetzt. Jener wurde von Chia-Hui Chang, Chun-Nan Hsu und Shao-Chen Lui entwickelt [CHL03]. Er wurde von uns modifiziert und um einen Klassifikator ergänzt, welcher von den generierten Mustern automatisch geeignete auswählt, mit ihnen Informationen extrahiert und diese in strukturierter Form ausgibt.

1.1 Vision, Ziel dieser Arbeit

Im Rahmen des *Sonderforschungsbereiches 588 (Humanoide Roboter)*, kurz SFB 588[1] wird an der Universität Karlsruhe ein menschenähnlicher Roboter entwickelt, der einfache Aufgaben im Haushalt übernehmen soll. Neben dem Erlernen solcher Tätigkeiten, sollte der Roboter auch dazu in der Lage sein, einen möglichst natürlichen Dialog mit den Menschen in seiner Umgebung zu führen, in dem er die für seine Aufgaben benötigten Informationen sammelt. Dazu zählen auch die Namen der ihn umgebenden Personen. Im Rahmen des SFB 588 wird ein System entwickelt, das bekannten Personen Auskünfte erteilen und unbekannte Personen kennen lernen kann. Holzapfel et al. stellen in „A Robot learns to know people - First Contacts of a Robot" [HSE+06] ein erstes System vor, das dazu in der Lage ist.

Ein Beispielszenario sieht wie folgt aus: Ein auf dem Flur stehender Roboter erkennt, ob eine vorbeigehende Person sich mit ihm unterhalten möchte. Mit Hilfe einer Gesichtserkennung kann das Gesicht der Person erkannt und diese als bekannte Person identifiziert werden. Ist sie hingegen unbekannt, soll der Roboter sie ansprechen und so ihre Identität in

[1] `http://www.sfb588.uni-karlsruhe.de/`

Erfahrung bringen. Dazu wird der Name der Person erfragt und zusammen mit Bildern ihres Gesichtes (FaceID) sowie ihrem charakteristischen Stimmmuster (VoiceID) abgespeichert. Außerdem wird das Vokabular des Spracherkenners um den Namen der Person erweitert, falls jener bislang nicht bekannt war.

Um den Dialog natürlicher gestalten zu können, identifiziert Felix Putze in seiner Diplomarbeit [Put08] soziale Beziehungen, Interessengruppen und Rollen. Personen werden durch Beziehungen gewichtet miteinander verknüpft. In dem entstehenden Netzwerk können Ballungen, also Interessengruppen, identifiziert werden. Eine Art solcher Beziehungen ist die Koautor-Beziehung. Sind zwei Autoren an derselben Veröffentlichung beteiligt, impliziert dies ein gemeinsames Interessengebiet. Damit der Roboter nicht im Dialog abfragen muss, was die Person bereits alles publiziert hat und welche Arbeit mit wem zusammen geschah, soll er die Möglichkeit haben, in der Zeit, in der er sich nicht mit jemandem unterhält, im Internet nach den ihm bekannten Personen zu suchen, sie miteinander in Beziehung zu setzen und über die Koautoren-Beziehung Namen von bisher unbekannten Personen finden, denen er jedoch mit einer gewissen Wahrscheinlichkeit in Zukunft noch begegnen wird. Außer der Anwendung des so neu hinzu gelernten Wissens im Dialog, wäre, basierend auf den recherchierten Daten, auch eine dynamische Website des Instituts denkbar. Dort könnte ein „Who is Who" der Institutsmitarbeiter samt ihrer Veröffentlichungen dargestellt und Arbeitsgruppen herausgestellt werden.

1.2 Formalisierung der Aufgabe

Formal lautet das Ziel der vorliegenden Studienarbeit wie folgt:

Gegeben seien Vor- und Nachname einer Person. Gesucht ist nun eine Liste wissenschaftlicher Publikationen dieser Person mit den Attributen *Autoren* samt *Vor-* und *Nachname*, *Zugehörigkeit* und *E-Mail-Adresse*, dem *Publikationstitel*, *Jahr* und *Abstract* der Publikation. Jene Liste ist durch automatische Webrecherche zusammenzustellen und als CSV-Datei (comma separated values) auszugeben.

2 Terminologie und Grundlagen

2.1 Informationsextraktion

Unter *Informationsextraktion* (engl. *Information Extraction, IE*) versteht man die ingenieursmäßige Anwendung von Verfahren aus der praktischen Informatik, der künstlichen Intelligenz und der Computerlinguistik auf das Problem der automatischen maschinellen Verarbeitung von unstrukturierter Information mit dem Ziel, Wissen bezüglich einer im Vorhinein definierten Domäne zu gewinnen [Wik07].

Im Gegensatz zum *Information Retrieval* (*IR*), welches als Hauptanliegen das Auffinden relevanter Dokumente in einer Ansammlung hat, produziert die Informationsextraktion strukturierte Daten, die leicht weiterverarbeitet werden können, was von entscheidender Wichtigkeit für viele Anwendungen des Text Minings ist.

Wie später gezeigt wird, liegt in diesem Fall das Wissen, also die Attribute von Publikationen, nicht vollkommen unstrukturiert vor, was den Computerlinguistik-Beitrag etwas in den Hintergrund rücken lässt.

2.2 Webrecherche

Eine Webrecherche läuft nach dem in Abb. 2.1 dargestellten Schema ab. Zunächst wird an eine gängige Suchmaschine, z. B. Google oder Yahoo, eine Anfrage mit dem Vor- und Nachnamen der Person, konkateniert mit „publications“ bzw. „publikationen“, gesendet. Rückgabe ist eine Menge von URLs zu potentiellen Publikationsseiten. Auf diesen Seiten wird nach Publikationen gesucht. Werden keine gefunden, wird die Seite abgelehnt. Wenn welche gefunden werden, besteht der nächste Schritt darin, zu jeder auf dieser Website befindlichen Publikation die zugehörigen Attribute zu extrahieren. Dabei sind auf der Publikationsseite je Veröffentlichung meist Autoren, Titel, Jahr, Konferenz bzw. Buchtitel sowie ein Link zu einer PDF-Datei oder ein Link zu einer weiteren Website, auf der oft

die PDF-Datei auffindbar ist, angegeben. Attribute wie Zugehörigkeit, E-Mail-Adresse und Abstract hingegen sind meist nur in der PDF-Datei selbst zu finden. Nach dem Extraktionsschritt liegt je Publikationsseite eine Tabelle vor, in welcher jede Zeile die Attribute einer Publikation enthält. Diese müssen nun sinnvoll vereinigt werden. Sinnvoll bedeutet, dass nicht einfach der Inhalt aller Tabellen in eine gemeinsame Tabelle kopiert und Duplikate eliminiert werden können. Da die Angabe von Vor- und Nachnamen im Allgemeinen eine Person nicht eindeutig identifiziert, muss unterschieden werden, welche Publikationsseiten tatsächlich von derselben Person stammen.

Abbildung 2.1: Ablauf einer Webrecherche

Der Ansatz, erst die Website und danach die PDF-Dateien zu untersuchen hat einige Vorteile. Zum einen sind Publikationsseiten (semi-) strukturiert (siehe 2.4), was es ermöglicht, unüberwachte Verfahren anzuwenden, die hauptsächlich auf der Struktur, in der die Daten vorliegen, beruhen, zum anderen ist die Klassifikation der Attribute im PDF-Dokument einfacher, wenn bereits Autoren und Titel feststehen. Auch sollte nicht vergessen werden, dass viele Publikationen nicht als PDF- oder PostScript-

Dateien öffentlich verfügbar sind. In solchen Fällen würde ein Ansatz, der allein auf den PDF-Dokumenten beruht, vollständig versagen.

2.3 Publikationsseiten

In Abbildung 2.2 ist ein Ausschnitt einer typischen Publikationsseite[1] dargestellt. Am Anfang steht die Überschrift „Publications", gefolgt von mehreren Abschnitten, welche die Publikationen nach Veröffentlichungsjahr gliedern. Je Publikation sind nun als erstes die Autoren in Kursivschrift, danach der Publikationstitel in fetter Schrift, gefolgt von der Konferenz oder dem Buchtitel, unter der/dem die Veröffentlichung stattfand, und schließlich ein Link zu einem PDF-Dokument aufgeführt. Es ist nicht immer ein PDF-Link angegeben. Weiterhin fällt auf, dass die Jahresangabe nicht nur in der Überschrift des Abschnittes, sondern auch in der Konferenz- bzw. Buchtitel-Zeile zu finden ist.

Der HTML-Code zu obiger Beispielseite ist auszugsweise in Abbildung 2.3 aufgeführt. Es ist gut erkennbar, dass die interessanten Attribute durch HTML-Tags voneinander separiert sind und jede Publikation ein ähnliches, wenn nicht sogar gleiches Muster aufweist.

Das ist auf Publikationsseiten häufig so. Die Daten liegen in Tabellen, Listen oder einer ähnlichen Struktur vor, in welcher linguistisches Wissen nur beschränkt weiterhilft. Doch nicht nur das. Diese Tabellen, Listen etc. haben auf verschiedenen Publikationsseiten ihr eigenes einheitliches Layout und Format. Keine allgemeine „Grammatik-Regel" kann alle möglichen Layouts und Formate beschreiben, so dass es nicht möglich ist, einen festen Extraktor für alle Publikationsseiten zu bauen. Somit benötigt jede Tabelle/Liste einen spezialisierten Extraktor, was es unpraktikabel macht, solche per Hand zu erstellen.

2.4 Strukturiertheit von Daten und Dokumenten

Der Begriff „semi-strukturiert" ist, bezogen auf das World-Wide Web, nicht eindeutig definiert. In der KI, wie auch in dieser Arbeit, bezieht sich der Begriff auf die Art und Weise, wie eine Website dargestellt wird, während er aus Datenbanken-Sicht die Art und Weise beschreibt, wie die inhaltstragenden Informationen auf einer Website organisiert sind.

[1] `http://isl.ira.uka.de/~hartwig/publications.html`

Publications

2007

Hartwig Holzapfel and Alex Waibel
Behavior Models for Learning and Receptionist Dialogs
Interspeech 2007, Antwerp, Belgium, 2007
download(pdf)

Catherina R. Burghart, Ralf Mikut, and Hartwig Holzapfel
Cognition-Oriented Building Blocks of Future Benchmark Scenarios for Humanoid Home Robots
In: Gal A. Kaminka and Catherina R. Burghart (Eds.), Evaluating Architectures for Intelligence. Papers from the 2007 AAAI Workshop, Menlo Park, California: AAAI Press

Stephan Könn, Hartwig Holzapfel, Hazim Kemal Ekenel, Alex Waibel
Integrating Face-ID into an Interactive Person-ID Learning System
International Conference on Computer Vision Systems (ICVS'07), Bielefeld, Germany, 2007
download(pdf)

2006

H. Holzapfel, A. Waibel
Providing Cognitive Functions for Interactive Learning with Speech and Multimodal Processing
Toward Cognitive Humanoid Robots Workshop at Humanoids 2006, Genoa, Italy, 2006

C. Fügen, P. Gieselmann, H. Holzapfel, F. Kraft
Natural Human Robot Communication
Human Centered Robotic Systems, HCRS, München, Germany, 2006
download(pdf)

Hartwig Holzapfel, Alex Waibel
A Multilingual Expectations Model for Contextual Utterances in Mixed-Initiative Spoken Dialogue
Interspeech 2006 - ICSLP, Pittsburgh PA, USA, 2006
download(pdf)

Abbildung 2.2: Ausschnitt einer typischen Publikationsseite

Wir sehen eine Website, die einzeln aufgeführte Informationen bietet, als *strukturiert* an, wenn jedes Attribut in einem Tupel mit Hilfe von uniformen syntaktischen Mustern korrekt extrahiert werden kann. Solche syntaktischen Muster können z. B. Begrenzer (delimiters) oder die Reihenfolge der Attribute sein. Sie sind immer dann anwendbar, wenn die Website automatisch, z. B. durch ein Script, generiert wurde, wobei jedes Attribut vom nächsten durch ein oder mehrere HTML-Tags getrennt wird. Andererseits ist eine Website *unstrukturiert*, wenn linguistisches Wissen benötigt wird, um die Attribute korrekt zu extrahieren. [HD98] gibt auch hierfür ein Beispiel. Es sollte betont werden, dass die *Strukturiertheit* einer Website von der Granularität der Attribute abhängt, die extrahiert werden sollen. Wenn zum Beispiel eine Website Kleinanzeigen von Mietwohnungen enthält, wobei jede Anzeige aus einem Satz, der die Wohnung

näher beschreibt, besteht, ist diese Seite unstrukturiert, wenn Attribute wie Wohnfläche, Zimmeranzahl und Preis von Interesse sind. Wird jedoch der ganze Satz zu einer Wohnung als ein Attribut aufgefasst, ist die Seite strukturiert.

Semistrukturierte Websites sind nicht unstrukturiert. Sie unterscheiden sich von strukturierten Seiten durch Tupel mit fehlenden Attributen, mehrwertige Attribute, variierende Attributpermutationen, Ausnahmen und Tippfehler. Daher ist anzunehmen, dass der Inhalt einer semistrukturierten Website nicht in ein strukturiertes Datenmodell passt, er also aus semistrukturierten Daten besteht. Andersherum ist es natürlich möglich, strukturierte Daten semistrukturiert oder sogar unstrukturiert darzustellen. Publikationsseiten sind, wie am Beispiel im Abschnitt 2.3 erkennbar, im Allgemeinen semistrukturiert oder gar strukturiert. Daher genügt für diese Studienarbeit ein Informationsextraktionsansatz für semistrukturierte Websites.

```
1  <font face="Verdana, Arial, Helvetica, sans-serif"><font size=5>Publications</font></font>
2
3  <br><br><font face="Verdana, Arial, Helvetica, sans-serif"><font size=4>2007</font></font>
4
5  <!-- Interspeech '07 -->
6  <br><br>
7  <font id=author>
8  Hartwig Holzapfel and Alex Waibel
9  </font><br><font id=pub-title>
10 Behavior Models for Learning and Receptionist Dialogs
11 </font><br><font id=pub-conf>
12 Interspeech 2007, Antwerp, Belgium, 2007
13 </font>
14 <BR><a href="pd/is2007_holzapfel.pdf" target="_top"><img src="img/pdf_sm.jpg" border="0"
       align="middle">download(pdf)</a>
15
16 <!-- AAAI '07 -->
17 <br><br>
18 <font id=author>
19 Catherina R. Burghart, Ralf Mikut, and Hartwig Holzapfel
20 </font><br><font id=pub-title>
21 Cognition-Oriented Building Blocks of Future Benchmark Scenarios for Humanoid Home Robots
22 </font><br><font id=pub-conf>
23 In: Gal A. Kaminka and Catherina R. Burghart (Eds.), Evaluating Architectures for
       Intelligence. Papers from the 2007 AAAI Workshop, Menlo Park, California: AAAI Press
24 </font>
25
26 <!-- ICVS '07 -->
27 <br><br>
28 <font id=author>
29 Stephan Käouml;nn, Hartwig Holzapfel, Hazim Kemal Ekenel, Alex Waibel
30 </font><br><font id=pub-title>
31 Integrating Face-ID into an Interactive Person-ID Learning System
32 </font><br><font id=pub-conf>
33 International Conference on Computer Vision Systems (ICVS'07), Bielefeld, Germany, 2007
34 </font>
35 <BR><a href="pd/icvs07-paper.pdf" target="_top"><img src="img/pdf_sm.jpg" border="0" align
       ="middle">download(pdf)</a>
36
37 <!--2006 -->
38 <br><br><font face="Verdana, Arial, Helvetica, sans-serif"><font size=4>2006</font></font>
39
40 <!-- Humanoids 2006 -->
41 <br><br>
42 <font id=author>
43 H. Holzapfel, A. Waibel
44 </font><br><font id=pub-title>
45 Providing Cognitive Functions for Interactive Learning with Speech and Multimodal
       Processing
46 </font><br><font id=pub-conf>
47 Toward Cognitive Humanoid Robots Workshop at Humanoids 2006, Genoa, Italy, 2006
48 </font>
49
50 <!-- HCRS 2006 -->
51 <br><br>
52 <font id=author>
53 C. Fügen, P. Gieselmann, H. Holzapfel, F. Kraft
54 </font><br><font id=pub-title>
55 Natural Human Robot Communication
56 </font><br><font id=pub-conf>
57 Human Centered Robotic Systems, HCRS, München, Germany, 2006
58 </font>
59 <BR><a href="pd/hcrs06.pdf" target="_top"><img src="img/pdf_sm.jpg" border="0" align="
       middle">download(pdf)</a>
```

Abbildung 2.3: Ausschnitt einer typischen Publikationsseite - der HTML-Code

3 Vergleich gegenwärtiger Ansätze

Zentrale Bestandteile des im Abschnitt 2.2 vorgestellten Ablaufs einer Webrecherche sind die Suche nach Publikationen auf einer Website und die Extraktion der zugehörigen Attribute. Beide sind eng miteinander verknüpft, schließlich wird eine Publikation durch ihre Attribute identifiziert. Es ist also sinnvoll, einen Ansatz zu wählen, der einerseits eine Klassifizierung der Websites in publikationsenthaltende und nicht-publikationsenthaltende Seiten ermöglicht, und andererseits aus publikationsenthaltenden Seiten sämtliche verfügbaren Attribute extrahiert.

3.1 Ansätze zur Informationsextraktion aus dem Internet

Es existieren derzeit viele Verfahren, die Informationsextraktion leisten. Traditionelle Informationsextraktion geschieht auf Klartext. Dabei wird linguistisches Wissen benutzt, wie Lexika, die den Wortschatz bestimmen, und Parser, welche gewisse grammatikalische Strukturen erkennen. Je nach Anwendungsbereich können solche Regeln manuell erstellt oder auch automatisch gelernt werden. Das geht in seiner Komplexität bis hin zum Verstehen natürlicher Sprache (*Natural Language Processing*, NLP), [Rog05].

Websites im Internet bieten allerdings die Möglichkeit, Formatierungsinformationen - dargestellt durch verschiedene *HTML-Tags*, [TWWWC99] - als Hilfmittel zur korrekten Informationsextraktion einzubeziehen. Im Allgemeinen gibt es zwei Ansätze, HMTL-Tags als Grundlage zur Informationsextraktion zu nutzen: sog. *Wrapper* und sog. *Patterns*. Beide lassen sich manuell erstellen sowie automatisch generieren bzw. maschinell lernen. Ein *Wrapper* besteht aus einer Menge von *Separatoren*, einen für jedes Attribut. Ein Separator besteht aus dem linken Kontext s^L und dem rechten Kontext s^R des entsprechenden Attributes. Für das Attribut *Autor* der

ersten Publikation in Abb. 2.3 ergibt sich z. B. bei Kontexttiefe vier der folgende Wrapper: s^L :=`Spc(_) Html(<br>) Html(<br>) Html(<font>)`, s^R :=`Html(</font>) Html(<br>) Html(<font>) C1Alph(Behavior)`. In seiner Dissertation [KWD97] stellt Kushmerick ein induktives Lernverfahren zur automatischen Generierung solcher Wrapper aus Websites mit 2 bis 44 Lernbeispielen vor. Hsu und Dung stellen in [HD98] Wrapper basierend auf endlichen Automaten dar, die sie mit *SoftMealy* betiteln. Auch sie benötigen dazu einige wenige Lernbeispiele je Website mit abweichender Struktur. Ein *Pattern* (*Muster*, im Folgenden auch *Extraktionsmuster*) ist eine Folge von HTML-Tags, welche sämtliche Attribute eines zu extrahierenden Datentupels enthält. Zum Beispiel würde das folgende Muster in dem Code in Abb. 2.3 auf die erste, dritte und fünfte Publikation passen und den Text, welchen die fett gedruckten Bestandteile beinhalten, extrahieren:

<**text**>
<**text**>
<**text**>
<**text**>
<**a**><img><text>. Einen ausführlicheren Überblick über verschiedene Arten von Extraktionsmustern gibt Muslea in [Mus99].

3.2 Wahl eines geeigneten Ansatzes

Da traditionelle Informationsextraktionsverfahren die zusätzlichen Informationen, welche HTML-Tags bieten, nicht nutzen können und Publikationen sehr selten in Form natürlichsprachlicher Sätze aufgelistet werden, scheidet die Anwendung traditioneller Ansätze auf semistrukturierte Websites aus. Weiterhin ist es nicht praktikabel, für jede mögliche Publikationsseite mehrere Wrapper oder ein Pattern manuell zu erstellen. Um Wrapper automatisch zu lernen - sog. *Wrapper Induction*, wird je Publikationsseite mindestens ein Lernbeispiel, besser noch zwei oder mehr, benötigt und ein maschinelles Lernverfahren angewendet. Auch das ist im betrachteten Szenario nicht möglich, da die Menge der zu untersuchenden Seiten nicht abgeschlossen ist und einem Passanten im Dialog mit dem Roboter nicht zugemutet werden kann, ein paar seiner Veröffentlichungen zu annotieren, ihm also spezifische *Lernbeispiele* zu geben, bevor das eigentliche Gespräch beginnt.

In der Arbeit „Automatic Information Extraction from Semi-Structured Web Pages By Pattern Discovery" [CHL03] stellen Chang, Hsu und Lui ein Verfahren vor, Patterns zu gegebenen Websites automatisch zu generieren. Das von den Autoren entwickelte System IEPAD (ein Akronym für Information Extraction based on Pattern Discovery) findet Extrakti-

onsmuster für gegebene Websites ohne vom Benutzer annotierte Beispiele. Dabei werden einige Mustererkennungs-Techniken, wie PAT-Trees, Ausrichtung mehrerer Zeichenketten aneinander (*Multiple String Alignment*) sowie Algorithmen zur Übereinstimmungssuche von Mustern in Texten (*Pattern Matching*) eingesetzt.

4 PAT-Trees zur Informationsextraktion aus Websites

Die Idee von Chang et al. basiert darauf, dass Informationen auf semistrukturierten Websites oft in einer speziellen Anordnung und Reihenfolge ausgerichtet sind und somit reguläre und kontinuierliche Muster aufweisen. Durch Finden solcher Muster auf Websites, kann ein Extraktor generiert werden. Chang et al. beschreiben in [CHL03] einen Mustererkennungsalgorithmus, der auf Websites ohne Trainingsbeispiele anwendbar ist.

In vorliegender Studienarbeit wird jener Algorithmus in leicht abgeänderter Form benutzt. In den folgenden Abschnitten beschreiben wir das Verfahren und gehen insbesondere auf die Abweichungen unserer Implementierung von der Vorlage ein.

4.1 Das Referenzsystem IEPAD

Das von Chang, Hsu und Lui implementierte System mit dem Namen **IEPAD** besteht hauptsächlich aus drei Komponenten: Dem *Pattern Discoverer*, der eine Website als Eingabe erhält und potentielle Muster erkennt, welche die zu extrahierenden Zieldaten enthalten. Dem *Rule Generator*, der eine graphische Benutzungsoberfläche bietet (den sog. *Pattern Viewer*) und die gefundenen Muster anzeigt. Hier ist es an dem Nutzer, das Muster auszuwählen, welches die gewünschten Daten extrahiert, und den Attributen Namen zuzuordnen, sie also zu klassifizieren. Jene so generierte *Extraktionsregel* wird für spätere Anwendungen abgespeichert. Die dritte Komponente ist schließlich der *Extractor* selbst. Er extrahiert, basierend auf der ausgewiesenen Extraktionsregel, gewünschte Informationen aus ähnlichen Websites.

Der größte Unterschied unserer Implementierung zu IEPAD liegt in der Auswahl geeigneter Patterns. Unser System geht weiter und automatisiert auch diesen Schritt. Daher fällt der *Rule Generator* incl. *Pattern*

Viewer weg. Die Extraktionsregeln werden innerhalb des *Extractors* automatisch und ohne weitere Benutzerinteraktion generiert. Das ist natürlich nur möglich, da es sich um eine abgegrenzte Domäne - die Publikationen - handelt, zu der außerdem hinreichend großes Hintergrundwissen verfügbar ist. Mehr dazu im Abschnitt 4.3.3.

Wie viele „Wrapper Induction"-Systeme sucht IEPAD zunächst Muster, um die Ränder der Datensätze zu erkennen. Danach erfolgt eine mehrstufige Ausrichtung (*Multi-Level Alignment*), um diese Datensätze in Attribute zu zerlegen. Jene Ausrichtung geschieht allerdings unter Leitung des Benutzers. Daher entschieden wir uns dafür, die Datensätze von vornherein in Attribute zu zerlegen. Das lässt zwar die zur Mustererkennung verwendete Datenstruktur (siehe Abschnitt 4.2.2) etwas größer werden, stellt jedoch angesichts der immer größer und billiger werdenden Arbeitsspeicherkapazitäten heutiger Computer kein Problem dar.

Der Pattern Discoverer besteht aus einem *Token Encoder*, einem *PAT-Tree Constructor*, einem *Pattern Filter* sowie einem *Extraction Rule Composer*. Ausgabe ist eine Menge von Muster, die auf der in Token übersetzten Website gefunden wurden. Die PAT-Tree-Technik [Mor68, GBYS92] ist der Schlüssel, der es ermöglicht, sich wiederholende Datenmuster effizient und präzise auf Websites zu finden.

Der Ansatz unterscheidet sich von früheren Arbeiten zur „Wrapper Induction" (unter Anderem von Hsu selbst [HD98], einen Überblick gibt [MMK99]) dadurch, dass hier die Extraktionsregeln auf *Muster* (*Patterns*) basieren, während dort *Begrenzer* (*Delimiters*) zu Grunde liegen. Genauer gesagt, benutzen deren Extraktionsregeln Begrenzer, um festzulegen, welche Zeichenkette auf der Website welchem Datenattribut entspricht. Zum Beispiel könnte eine Begrenzer-basierte Regel besagen, dass das Attribut „Publikationsjahr" mit dem Präfix `<i>` beginnt und durch ein Suffix mit einer Zahl gefolgt von dem HTML-Tag `</i>` endet. Dann würde eine Zeichenkette wie `<i>1999</i>` mit jener Regel übereinstimmen und `1999` als „Publikationsjahr" extrahiert werden. Im Gegensatz dazu würde eine Muster-basierte Regel besagen, dass das Attribut „Publikationsjahr" durch das Muster `<i><text></i>` dargestellt wird, wobei `<text>` für eine beliebige Text-Zeichenkette steht. Die Nutzung musterbasierter anstelle begrenzerbasierter Extraktionsregeln ist darin begründet, dass die zu extrahierenden Daten oft mit Hilfe vordefinierter HTML-Vorlagen generiert werden. Das motiviert natürlich die Idee, nach solchen Vorlagen (oder Muster) zu suchen. Zusätzlich kommen diese Muster gewöhnlich *regelmäßig* ausgerichtet und *kontinuierlich* aufeinanderfolgend vor, so dass sie für den Betrachter leicht verständlich sind. Jene Eigenschaften erlauben es

auch, automatisch Extraktionsmuster solcher Vorlagen zu finden. Anders gesagt, die Aufgabe der Generierung von Extraktionsregeln kann durch Mustererkennung ohne vom Benutzer annotierte Trainingsbeispiele, wie sie für „Wrapper Induction"-Systeme unabdingbar sind, gelöst werden.

4.2 Pattern Discoverer

In Abbildung 4.1 ist der Ablauf innerhalb des Pattern Discoverers sowie des Extractors dargestellt. Zu einer gegebenen HTML-Seite wird zunächst der Token-Encoder die Seite in eine Aufzählung abstrakter Repräsentationen der HTML-Tags und des dazwischen befindlichen Textes, im Folgenden *Token-String* genannt, kodieren. Jedes Token wird durch eine natürliche Zahl repräsentiert. Der PAT-Tree Constructor [Mor68, GBYS92] benutzt den Token-String, um einen PAT-Tree aufzubauen, welcher im Anschluss vom Pattern Discoverer genutzt wird, um Muster, so genannte *maximale Wiederholungen*, zu finden. Diese maximalen Wiederholungen werden einem Filter übergeben, welcher unerwünschte Muster herausfiltert und Kandidat-Muster produziert. Schließlich überarbeitet der Extraction Rule Composer jedes Kandidat-Muster und erstellt eine Extraktionsregel in Form eines regulären Ausdrucks. Die nun folgenden Abschnitte beschreiben diese Komponenten im Detail.

4.2.1 Kodierung von Websites

Da HTML-Tags die Basis der Repräsentation von Dokumenten im Internet sind und sie selbst, wie Abschnitt 2.3 entnehmbar, die strukturellen Informationen tragen, ist es intuitiv, die Tag-Token-Sequenz, wie sie von den HTML-Tags zwischen `<body>` und `</body>` gebildet wird, zu untersuchen und den Text zwischen den Tags zu vernachlässigen. Daher bietet sich folgende Abstraktion an:

1. Jedes Tag wird als Tag-Token `Html(<tag_name>)` kodiert.
2. Der Text zwischen zwei Tags wird in einem speziellen `<text>`-Token gekapselt.
3. In `Html(<a>)`-Token wird zusätzlich die URL des `href`-Attributs gespeichert.
4. Kommentare (zwischen `<!--` und `-->`) und Text, der nur Leerzeichen, Tabulatoren und Zeilenumbrüche enthält, werden ignoriert.

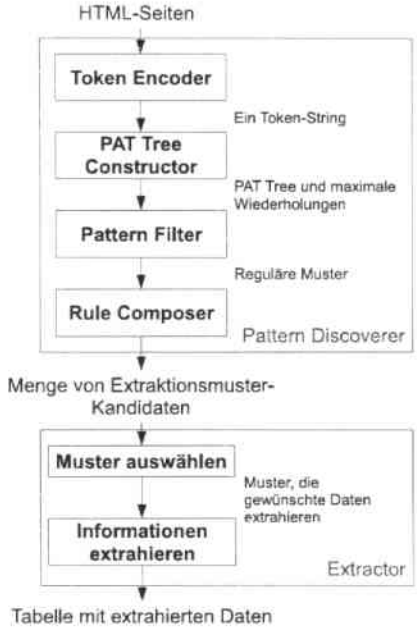


Abbildung 4.1: Flussdiagramm des Pattern Discoverers und Extractors

Jedes Token wird durch eine natürliche Zahl repräsentiert. Das ist die erste Abweichung vom Referenzsystem IEPAD. Dort wird nämlich jedem Token ein Binärcode fester Länge zugeordnet und der PAT-Tree auf klassischem Wege als Binärbaum implementiert. Um unsere Implementierung jedoch so einfach wie möglich zu halten, entschieden wir uns für die Kodierung mit natürlichen Zahlen. Die zweite Abweichung liegt in der Existenz eines speziellen `Html(<a>)`-Tokens und der Abspeicherung des Textes in `<text>`-Token. Chang et al. speichern stattdessen in jedem Token die Byte-Position des entsprechenden Tags im ursprünglichen HTML-Dokument, um den Text dort später wiederzufinden. Durch Speicherung der Texte und Link-URLs erübrigt es sich, zur Informationsextraktion noch einmal in das HTML-Dokument zurückkehren zu müssen. Es entsteht ein Token-String, der alle Informationen für die weiteren Verarbeitungsschritte enthält.

In IEPAD wird zusätzlich zwischen sog. *Block-Level Tags* und *Text-Level Tags* unterschieden. Die einen definieren die Struktur eines Dokuments und die anderen die Charakteristiken (Format und Stil, etc.) der Textinhalte [CHL03, WDG07]. Block-Level Tags sind z. B. Überschriften, Textcontainer, Listen und weitere Klassifikationen wie Tabellen und For-

mulare. Text-Level Tags legen z. B. die Schriftart und -größe, Fett- und Kursivschrift sowie Links und Bilder fest. Diese Abstraktionsstufen erlauben es, Muster auf verschiedenen Ebenen zu produzieren. In unserem Fall sind allerdings Domäne und Granularität der zu extrahierenden Attribute klar. Daher haben wir die Unterscheidung in Block- und Text-Level Tags zwar implementiert, jedoch wird stets die feinere Aufteilung der Text-Level Tags benutzt.

In Abbildung 4.2 ist als Beispiel der Congo Code aufgeführt. Er wird in [KWD97] und [CHL03] ebenfalls unter diesem Namen referenziert.

```
1  <html>
2  <head>
3  <title>The Congo Code</title>
4  </head>
5  <body>
6
7  <H1>Country Code </H1><UL>
8  <LI>Congo <I>242</I>
9  <LI>Egypt <I>20</I>
10 <LI>Belize, 501
11 <LI>Spain <I>34</I></UL>
12
13 </body>
14 </html>
```

Country Code

- Congo *242*
- Egypt *20*
- Belize, 501
- Spain *34*

Abbildung 4.2: Der Congo Code (links) und dessen Ansicht im Webbrowser

Aus dem Code in Abb. 4.2 ergibt sich bei Block-Level-Kodierung, welche hier lediglich aus Gründen der Übersichtlichkeit gewählt wurde, folgender aus 13 Token bestehender Token-String:

```
Html(<h1>)Text(_)Html(</h1>)Html(<ul>)
Html(<li>)Text(_)
Html(<li>)Text(_)
Html(<li>)Text(_)
Html(<li>)Text(_)Html(</ul>)
```

Die Notation wurde von Hsu in [HD98] eingeführt. Jedes Token wird als $t(v)$ geschrieben, wobei t die Token-Klasse und v eine Zeichenkette ist. In der genannten Arbeit werden neben der `Html()`-Klasse diverse Textklassen eingeführt, z. B. für Wörter in Großbuchstaben, numerische Zeichenketten oder Interpunktionszeichen. Für den PAT-Tree-Ansatz genügt jedoch die

Unterscheidung zwischen `Html()`-Token und `Text()`-Token. Der Unterstrich stellt eine Generalisierung von Zeichenketten dar und steht somit für beliebige Strings der entsprechenden Klasse.

Token	Code
`Html(<ul>)`	0
`Html(</ul>)`	1
`Html(<li>)`	2
`Html(</li>)`	3
`Html(<h1>)`	4
`Html(</h1>)`	5
`Text(_)`	6

Tabelle 4.1: Codierung der Token

Legt man nun eine einfache Codierung (wie z. B. in Tabelle 4.1) zugrunde, sieht der codierte Token-String wie folgt aus: `4650262626261`

4.2.2 PAT-Trees für maximale Wiederholungen erstellen

Mit dem Begriff *Wiederholung* ist eine Teilzeichenkette gemeint, welche in der Gesamtzeichenkette mindestens zweimal vorkommt. Das Konzept der *maximalen Wiederholung* bezieht sich auf das längste Muster und reduziert somit die Anzahl infrage kommender Muster. Die Idee ist, die Wiederholung in beide Richtungen bis zu ihrer längsten Ausdehnung zu erweitern. Wie in [CHL03] heißt eine Wiederholung *linksmaximal* (rechtsmaximal), wenn sie nicht für alle Vorkommen nach links (rechts) erweiterbar ist (siehe auch [CLW01]). Eine Wiederholung heißt *maximal*, wenn sie sowohl links- als auch rechtsmaximal ist. Formal lautet die Definition wie folgt:

Definition 1 (*Maximale Wiederholung*)

Sei S eine Zeichenkette. Eine maximale Wiederholung *α ist eine Teilzeichenkette von S, die an k paarweise verschiedenen Positionen $p_1, p_2, \ldots, p_k$ in S vorkommt, so dass für mindestens ein i, j-Paar, $1 \leq i < j \leq k$, das $(p_i - 1)$-te Token vom $(p_j - 1)$-ten Token in S verschieden ist (genannt* linksmaximal*), und für mindestens ein x, y-Paar, $1 \leq x < y \leq k$, das $(p_x + |\alpha|)$-te Token vom $(p_y + |\alpha|)$-ten Token verschieden ist (genannt* rechtsmaximal*).*

Um automatisch sich wiederholende Muster zu finden, wird eine Datenstruktur mit dem Namen *PAT-Tree* verwendet. Jene indiziert alle Suffixe des kodierten Token-Strings. Ein PAT-Tree ist ein *Patricia Tree* (Practical Algorithm To Retrieve Information Coded In Alphanumeric [Mor68]), konstruiert über alle möglichen *SiStrings* eines Textes.

Die Bezeichnung SiString steht für *semi infinite String* und benennt eine Teilfolge von Zeichen eines gegebenen Textes, in diesem Fall des Token-Strings. Sie beginnt an einer bestimmten Position und dehnt sich unbegrenzt weit nach rechts aus - längstens bis zum Textende. Im Beispiel des Congo Codes aus Abb. 4.2 ergeben sich somit folgende 13 SiStrings:

s1: 4650262626261
s2: 650262626261
s3: 50262626261
s4: 0262626261
s5: 262626261
s6: 62626261
s7: 2626261
s8: 626261
s9: 26261
s10: 6261
s11: 261
s12: 61
s13: 1

Ein Patricia Tree ist in seiner ursprünglichen Form eine spezielle Implementierung eines binären Digitalbaumes mit Pfadkompression (Teile von Zeichenketten, die nicht zu Verzweigungen führen, werden übersprungen). Jeder interne Knoten im Baum stellt ein unterschiedliches Bit der Zeichenketten-Suffixe desselben Teilbaums dar. Wie bereits erwähnt, haben wir eine abgeänderte Form des Patricia Trees implementiert, bei der ein innerer Knoten nicht nur zwischen 0 und 1 unterscheidet, sondern für jedes Zeichen des verwendeten Alphabets - hier also natürliche Zahlen - einen direkten Unterbaum erlaubt. Das erspart die etwas aufwendigere Binärkodierung der Token, hält den Baum klein, da innerhalb des Codes eines Tokens keine Verzweigungen mehr nötig sind, und macht das Token, welches die Verzweigung verursacht hat, direkt identifizierbar. Wie ein Suffixbaum [Gus97] speichert der Patricia Tree alle Suffix-Zeichenketten in den Blattknoten. Für einen Token-String der Länge n ergeben sich n SiStrings, also n Suffixe. Der resultierende PAT-Tree hat n Blattknoten

und im worst-case (alle internen Knoten haben Verzweigungsgrad 2) $n-1$ interne Knoten und somit eine Größe in $O(n)$.

PAT-Trees strukturieren die Eingabe derart, dass alle Suffixe mit demselben Präfix im selben Teilbaum abgelegt werden. Die daraus resultierenden Eigenschaften erweisen sich als nahezu ideal für das Auffinden sich wiederholender Muster:

1. Alle Suffixe in einem Teilbaum haben ein gemeinsames Präfix. Das ist gerade die *Pfadbeschriftung* von der Wurzel des Baumes bis zur Wurzel des Teilbaumes.

2. Die Anzahl der Blätter im Teilbaum entspricht genau der Anzahl an Vorkommen der Pfadbeschriftung im Token-String.

3. Jede Pfadbeschriftung repräsentiert eine rechtsmaximale Wiederholung im Token-String.

Der aus dem codierten Token-String des Congo Codes (`4650262626261`) entstandene PAT-Tree ist in Abbildung 4.3 aufgeführt. Er wurde aus dreizehn Ganzzahlsequenzen aufgebaut. Jedes Blatt wird durch ein Quadrat, beschriftet mit der Startposition des SiStrings, dargestellt. Zum Beispiel entspricht Blatt 11 dem SiString, der mit dem elften Token im Token-String beginnt. Jeder interne Knoten wird durch einen Kreis repräsentiert, dessen Beschriftung die Position im Token-String angibt, an der die SiStrings im Teilbaum, dessen Wurzel der interne Knoten darstellt, sich zum ersten Mal unterscheiden. Zum Beispiel steht das erste unterschiedliche Token von SiString 6 und 10 an Position 4 dieser beiden SiStrings. Das gemeinsame Präfix ist die Pfadbeschriftung von der Wurzel bis zum internen Knoten 4, lautet also `626`.

In einem PAT-Tree werden also alle SiStrings mit demselben Präfix in denselben Teilbaum eingeordnet. Daher bietet diese Datenstruktur sehr effiziente Lösungen linearer Zeitkomplexität für komplexe Zeichenketten-Suchprobleme, unter Anderem zur Suche maximaler Wiederholungen [Mor68, Gus97]. Da jeder interne Knoten eines PAT-Trees eine Verzweigung aufzeigt, impliziert dieser ein unterschiedliches Token, das auf ein gemeinsames Präfix mehrerer SiStrings folgt. Somit repräsentiert die Kantenbeschriftung von der Wurzel bis zu einem internen Knoten eine rechtsmaximale Wiederholung im Token-String. Allerdings ist nicht jede dieser Wiederholungen auch linksmaximal. Zum Beispiel ist die Pfadbeschriftung des internen Knotens 4 nicht linksmaximal, da die SiStrings 6, 8 und 10 alle dasselbe Vorgängertoken `Html(<li>)` mit dem Code 2 haben. Sei der

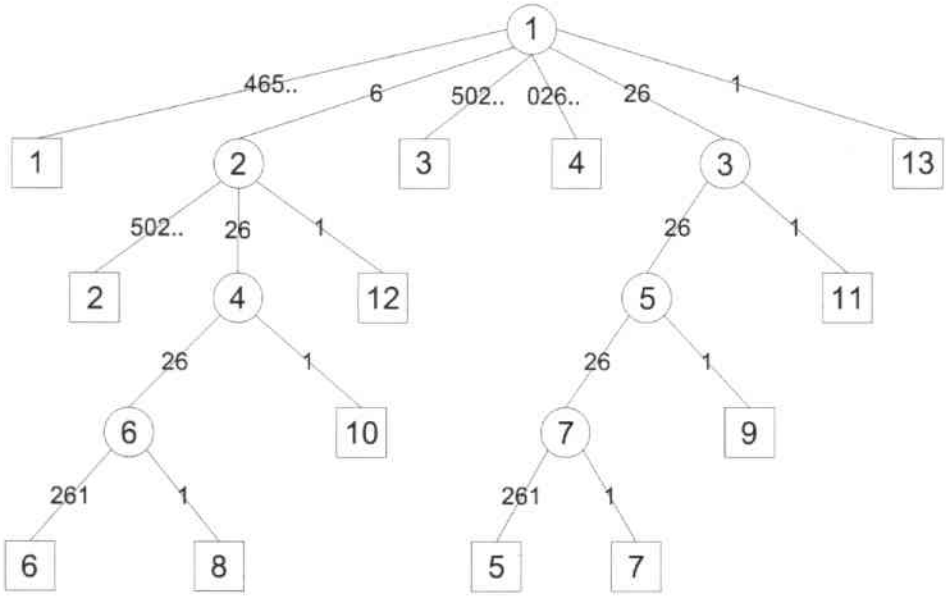


Abbildung 4.3: Der PAT-Tree zum Congo Code

Code des $(p_k - 1)$-ten Tokens als *Vorgängercode* des SiStrings an Position p_k bezeichnet. Damit die Pfadbeschriftung eines internen Knotens v eine maximale Wiederholung ist, müssen mindestens zwei Blätter (SiStrings) in dem Unterbaum von v verschiedene Vorgängercodes aufweisen. Solch ein Knoten v heiße *linksverschieden*. Der Definition nach pflanzt sich die Eigenschaft linksverschieden zu sein im PAT-Tree aufwärts fort. Folglich können alle maximalen Wiederholungen in linearer Zeit, bezogen auf die Baumgröße, gefunden werden.

Somit muss der PAT-Tree bei gegebener Mindestwiederholungsanzahl k und Mindestmusterlänge $|\alpha|$ lediglich durch eine Tiefensuche traversiert und alle Pfadbeschriftungen aufgesammelt werden, um alle rechtsmaximalen Wiederholungen zu finden. An jedem internen Knoten wird durch Überprüfung der Vorgängercodes aller Blätter des Teilbaumes die Linksmaximalität verifiziert. Sind alle Vorgängercodes identisch, könnte diese Wiederholung erweitert werden, ist somit nicht linksmaximal und kann verworfen werden. Aufgrund des Aufbaus des PAT-Trees ist die linksmaximale Erweiterung jener Wiederholung in einem anderen Teilbaum auffindbar. Zum Beispiel ist die Pfadbeschriftung des internen Knotens 4 nicht linksmaximal, weil die SiStrings 6, 8 und 10 den gemeinsamen Vorgängercode 2 haben. Die linksmaximale Erweiterung dieser Wieder-

holung wird schließlich beim Erreichen des internen Knotens 5 gefunden. Die Pfadbeschriftung ergibt als maximale Wiederholung den Code 2323, also die Token-Sequenz `Html(<li>)Text(_)Html(<li>)Text(_)`, welche im Token-String dreimal vorkommt.

4.2.3 Reguläre und kontinuierliche Muster finden

Wie bereits gesagt, ist der überwiegende Teil der gewünschten Informationen basierend auf vordefinierten Vorlagen generiert worden und kommt daher *regelmäßig* ausgerichtet und *kontinuierlich* aufeinanderfolgend vor. Um diese Eigenschaften abzudecken, definieren Chang et al. zwei Maße, „Varianz" und „Dichte". Mit deren Hilfe kann entschieden werden, ob eine maximale Wiederholung ein vielversprechendes Extraktionsmuster darstellt. Seien die Vorkommen einer maximalen Wiederholung α aufsteigend nach ihrer Position geordnet, so dass $p_1 < p_2 < p_3 < \ldots < p_k$ gilt, wobei p_i die Position im kodierten Token-String angibt.

Definition 2 (*Varianz*)

Die Varianz *eines Musters wird unter Verwendung der Varianz des Intervalls $(p_{i+1} - p_i)$ zwischen zwei unmittelbar aufeinanderfolgenden Vorkommen berechnet. Sie ist der Quotient aus der Standardabweichung des Intervalls und der mittleren Intervalllänge.*

$$variance(\alpha) = \frac{\sigma(d_i | 1 \leq i < k, d_i = p_{i+1} - p_i)}{(p_k - p_1)/(k-1)} \tag{4.1}$$

Definition 3 (*Dichte*)

Die Dichte *ist definiert als der Prozentsatz der Wiederholungen im Intervall zwischen dem ersten und dem letzten Vorkommen.*

$$density(\alpha) = \frac{(k-1) \cdot |\alpha|}{p_k - p_1} \tag{4.2}$$

wobei $|\alpha|$ die Anzahl an Token in α sei.

Im Allgemeinen sind die relevanten Informationen auf script-generierten Websites in Vorlagen mit kleiner Varianz (nahe Null) und großer Dichte (nahe Eins oder größer) angeordnet. Um nun potentiell gute Muster herauszufiltern, genügt ein Schwellwert für jedes dieser Maße. Nur Muster mit einer Varianz kleiner als der Varianz-Schwellwert und einer Dichte größer als der Dichte-Schwellwert werden als Kandidat-Muster angesehen.

Der beschriebene Ansatz ist leicht implementierbar. Allerdings kann er bei manchen Layout-Vorlagen fehlschlagen, wenn der Varianz-Schwellwert nicht korrekt gesetzt ist. Dies liegt darin begründet, dass auch regelmäßige Muster einen großen Varianz-Koeffizienten haben können. Jener kommt z. B. durch Zwischenüberschriften oder Werbebanner zustande, welche die zu extrahierenden Informationen in mehrere Teile zerlegen. Solche Ausnahmen verursachen maximale Wiederholungen mit großer Varianz.

4.2.3.1 Ballung der Vorkommen (Occurrence Clustering)

Um mit Mustern mit einer größeren Varianz als dem spezifizierten Schwellwert umgehen zu können, werden die Vorkommen eines Musters zusammengeballt. Dadurch lässt sich herausfinden, ob eine Partition der Vorkommen einen unabhängigen und regelmäßigen Block bildet. Eine einfache Schleife kann diese eindimensionale Ballung vornehmen [CHL03]. Für dieses Vorgehen wird ein weiterer Varianz-Schwellwert benötigt, welcher die maximal erlaubte Varianz einer Partition angibt. Jener sollte auf einen kleinen Wert gesetzt werden, um die Anzahl generierter Partitionen klein zu halten.

4.2.4 Extraktionsmuster bilden

Zusätzlich zu der großen Varianz verursachen Muster mit einer Dichte kleiner als Eins ein weiteres Problem. PAT-Trees erzeugen stets Muster für exakte Übereinstimmungen, Vorlagen mit Ausnahmen können mit PAT-Trees nicht gefunden werden. Um auch nicht-exakte bzw. annähernde Übereinstimmungen zu ermöglichen, wird die *Multiple String Alignment*-Technik verwendet.

Angenommen, ein Kandidat-Muster habe k Vorkommen $p_1, p_2, \ldots, p_k$ im kodierten Token-String. Sei P_i der SiString, der an p_i beginnt und an Position $p_{i+1} - 1$ endet. Das Problem liegt nun darin, die gegenseitige Ausrichtung der $k - 1$ SiStrings $S = \{P_1, P_2, P_3, \ldots, P_{k-1}\}$ zu finden, so dass das generalisierte Muster benutzt werden kann, um alle benötigten Datensätze zu extrahieren. Angenommen, es wäre z. B. das Muster „143“ im kodierten Token-String „1432414352143524143“ gefunden worden. Weiterhin hätten wir das folgende Multiple String Alignment für die Tokenfolgen „14324“, „14352“ und „143524“:

```
1 4 3 - 2 4
1 4 3 5 2 -
1 4 3 5 2 4
```

Das Extraktionsmuster kann somit zu „`143[5|-]2[4|-]`“ generalisiert werden, um alle drei Vorkommen abzudecken. Mit Hilfe solch regulärer Ausdrücke von Datensatz-Mustern können Ausnahmen, wie fehlende Attribute, mehrere Attributwerte sowie verschiedene Attributpermutationen, die auf Websites vorkommen und deren Semi-Strukturiertheit ausmachen, behandelt werden [HD98].

„Multiple String Alignment“, also das Ausrichten mehrerer Zeichenketten aneinander, ist eine Generalisierung des Ausrichtens zweier Zeichenketten. Letzteres ist, seien n und m die Längen der Zeichenketten, in $O(n \cdot m)$ durch *dynamische Programmierung* zur Berechnung der optimalen *Editierdistanz* lösbar. Die Erweiterung des dynamischen Programms auf k Zeichenketten lässt die zweidimensionale Editierabstandsmatrix jedoch auf einen k-dimensionalen Kubus, durch den ein optimaler Pfad berechnet werden muss, anwachsen. Der Aufwand liegt also in $O(n^k)$, wobei n die Länge der längsten vorkommenden Zeichenkette ist. Als Alternative existiert ein Approximationsalgorithmus mit relativer Gütegarantie von 2 [Gus97, S. 348ff]. Das heißt, dass der gemeinsame „*Score*“[1] des Approximationsalgorithmus' nicht größer als das Doppelte des optimalen „Scores“ der k Zeichenketten wird. Zunächst berechnet der Algorithmus den so genannten *Center String* S_c. Das ist diejenige Zeichenkette mit dem kleinsten aufsummierten Editierabstand („*Consensus Error*“ genannt) zu den restlichen $k-1$ Zeichenketten. Ist der Center String einmal gefunden, wird jede Zeichenkette iterativ daran ausgerichtet und so das „Multiple String Alignment“ konstruiert. Im Anschluss wird genau diese Ausrichtung der Zeichenketten aneinander genutzt, um das Extraktionsmuster zu konstruieren.

Es wird somit auf jedes Muster mit einer Dichte kleiner als 1 der *Center Star*-Approximationsalgorithmus zum „Multiple String Alignment“ angewendet, um das Extraktionsmuster zu generalisieren. Es ist zu beachten, dass der Erfolg dieser Technik auf der Annahme beruht, dass die Extraktionsmuster oft kontinuierlich aneinander grenzen. Es kommt vor, dass die Ausrichtung Extraktionsmuster mit zu vielen Alternativen hervorbringt. Solche sind eher uninteressant, da es für l Alternativen 2^l mögliche Permutationen gibt. Daher wird eine obere Schranke m für die Anzahl an Alternativen festgelegt. In den Experimenten von Chang et al. wurde m gewöhnlich auf 10 gesetzt. In unseren Untersuchungen stellte sich jedoch

[1] Gemeint ist hier der sog. „*sum of pairs (SP) score*“, der aufsummierte paarweise Editierabstand einer Zeichenkette zu allen anderen $k-1$ Zeichenketten. Siehe [Gus97, S. 343]

heraus, dass gerade auf größeren Publikationsseiten, auf denen verschiedene Publikationstypen aufgeführt sind, höhere Werte benötigt werden. So haben wir diesen Wert meist auf 100 gesetzt.

4.2.4.1 Muster-Rotation (Pattern Rotation)

Sei „$c_1c_2c_3 \ldots c_n$" ein generalisiertes Extraktionsmuster, wobei c_i $(1 \leq i \leq n)$ entweder ein Symbol oder eine Teilmenge von $\Sigma \cup \{-\}$ (Symbole, die an Position i vorkommen können) ist. c_1 muss jedoch nicht zwangsläufig tatsächlich der Anfang eines gewünschten Datensatzes sein. Aus diesem Grund wird eine *Rechts-Rotation*sprozedur angewendet, die das Muster mit der Zeichenkette vor dem ersten Vorkommen p_1 von rechts nach links vergleicht. Anliegen ist, auf diese Weise die korrekte Startposition zu finden und ein neues Muster „$c_jc_{j+1} \ldots c_nc_1 \ldots c_{j-1}$" zu generieren. Der Prozess wird solange fortgeführt, bis der Vorgängercode des ersten Vorkommens nicht mit dem Code des letzten Tokens übereinstimmt. Analog wird eine *Links-Rotation*sprozedur angewendet, welche das Muster mit der Zeichenkette nach dem letzten Vorkommen p_k von links nach rechts vergleicht. Ist der Nachfolgercode des letzten Vorkommens derselbe wie der des ersten Tokens, so wird das Muster zu „$c_2c_3 \ldots c_nc_1$" rotiert. Dieser Prozess wird fortgesetzt, solange der Nachfolgercode des letzten Vorkommens mit dem Code des ersten Tokens übereinstimmt. Durch diese Muster-Rotation kann der Rand der Datensätze korrekt gefunden werden.

4.2.5 Umsetzung

Bis hier hin ist unser System dem von Chang, Hsu und Lui sehr ähnlich. Es können alle maximalen Wiederholungen, deren Länge und Anzahl an Vorkommen gewisse Schwellwerte überschreiten, im kodierten Token-String mit Hilfe des aufgebauten PAT-Trees effizient gefunden werden. Weiterhin werden durch den Varianzkoeffizienten und die Dichte diejenigen maximalen Wiederholungen herausgefiltert, welche vielversprechende Muster ergeben. Datensätze von Mustern mit großer Varianz können durch Ballung in kleinere Blöcke unterteilt werden. Muster mit kleiner Dichte werden, basierend auf der Annahme kontinuierlichen Vorkommens, mit Hilfe von „Multiple String Alignment" generalisiert, um vollständigere Extraktionsmuster zu erhalten.

Die Unterschiede zu Chang et al. liegen zum einen in der Kodierung der HTML-Token, wir verwenden Ganzzahlcodes anstelle einer Binärkodierung, was sich auch auf den Aufbau des PAT-Trees auswirkt. Zum An-

deren benutzt unser System von vornherein Text-Level-Tags, anstatt erst mit Hilfe von Block-Level-Tags die grobe Struktur ausfindig zu machen und später die einzelnen Segmente noch einmal zu unterteilen. Des Weiteren werden in den `<text>`- und `<a>`-Token die Texte und URLs aus dem Quell-HTML-Dokument gespeichert, um zur späteren Extraktion nicht noch einmal jenes Dokument durchsuchen zu müssen.

4.3 Extractor

Die maximalen Wiederholungen, welche vom Pattern Discoverer automatisch gefunden wurden, entsprechen Datensätzen, die auf einer semistrukturierten Website vorkommen. Diese Muster werden allein auf Basis der Folge von HTML-Token erkannt. Um jedoch zu unterscheiden, welche Muster nützlich sind und welche nicht, ist Hintergrundwissen erforderlich. Wie bereits erwähnt, bieten Chang et al. hierzu dem Benutzer eine graphische Schnittstelle. Dort werden ihm die gefundenen Muster sowie der jeweils extrahierte Inhalt angezeigt. Er wählt ein oder mehrere Muster aus und klassifiziert die Attribute. Jene Muster werden abgespeichert und können im Anschluss dazu verwendet werden, Informationen von ähnlichen Websites[2] zu extrahieren.

Im gegebenen Anwendungsfall (siehe Kapitel 1) lässt sich die Domäne, aus der die zu extrahierenden Informationen stammen, allerdings gut eingrenzen. Das ermöglicht es, unter Zuhilfenahme von Domänenwissen aus einer Datenbank, auch die Schritte der Musterauswahl und Attributklassifizierung zu automatisieren. Wie das im Detail geschieht, beschreibt Abschnitt 4.3.3. Zuvor müssen jedoch die Informationen erst einmal extrahiert und in eine zur Weiterverarbeitung günstigere Form gebracht werden. Die folgenden zwei Abschnitte beschreiben, wie das erreicht werden kann.

4.3.1 Vorkommen der Muster finden

Jedes vom Pattern Discoverer gelieferte Muster wird zunächst mit dem kodierten Token-String verglichen, um alle Vorkommen, auf die der reguläre Ausdruck passt, zu finden. Chang et al. schlagen zu diesem Zweck gängige Suchalgorithmen für Zeichenketten, wie Knuth-Morris-Pratt oder Boyer-Moore vor. Die von mir verwendete Kodierung der Token erlaubt

[2] Als *ähnlich* werden dabei Websites angesehen, die von derselben Quelle stammen (meist die Ergebnisseite einer Suchmaschinenanfrage), also von demselben Script generiert wurden und somit anderen Inhalt bei gleicher Struktur aufweisen.

allerdings einen Trick, der es ermöglicht, Standard-Bibliotheken zur Suche von Übereinstimmungen in Texten zu verwenden. Sowohl im Token-String als auch in den Extraktionsmustern wird jedes Token durch eine natürliche Zahl identifiziert. Für die in [CHL03] aufgeführten HTML-Tags (22 Block-Level-Tags und 25 Text-Level-Tags) sind das $47 \cdot 2$ (öffnende und schließende Tags), also 94 Codes. Hinzu kommen noch je zwei Codes für `<text>`-, `<whitespace>`- und `<unk>`-Token. Letztere werden benötigt, um auch mit nicht standard-konformem HTML-Code und Spracherweiterungen umgehen zu können. Das sind somit 100 verschiedene Codes, z. B. 0 bis 99. Jene Codes können auch als Werte des Datentyps `Character` interpretiert werden. Transformiert man nun durch Addition eines Offsets p diese Codes in einen Unicode-Zeichenbereich, der keine Steuerzeichen oder für reguläre Ausdrücke reservierte Metazeichen enthält, erhält man aus dem Token-String eine Zeichenkette vom Typ `String`, in welcher mit Standard-Methoden aus String-Bibliotheken nach Teilzeichenketten, auf die der reguläre Ausdruck passt, gesucht werden kann. Ergebnis ist eine Menge solcher passender Teilzeichenketten, im Folgenden einfach „Vorkommen" genannt.

4.3.2 Informationen extrahieren

Nachdem die Vorkommen im Token-String gefunden wurden, wird jedes von ihnen am Muster ausgerichtet. Dadurch entsteht direkt eine Segmentierung der Datensätze, bei der jedes `<text>`- und jedes `<a>`-Token potentiell einem Attribut entspricht. Man beachte, dass jene Token aus dem kodierten Token-String bereits alle extrahierbaren Informationen enthalten. Die übrigen HTML-Token tragen nur strukturelle Informationen und können im weiteren Prozess unbeachtet bleiben. Die genaue Prozedur, welche die Menge aller Datensätze, die auf ein Muster α passen, segmentiert, wird in [CHL03] aufgeführt. Damit können bei m `<text>`- und `<a>`-Token in einem Extraktionsmuster alle passenden Token-Folgen in m Blöcke aufgeteilt werden. Ist das Muster ein gewünschtes, d. h. stehen in den `<text>`- und `<a>`-Token die zu extrahierenden Attribute der Publikationen, so enthält im Idealfall eine passende Token-Folge genau die Attribute (eines pro Block) einer Publikation. Ordnet man alle Token-Folgen untereinander und deren Blöcke nebeneinander an, entsteht eine Matrix, genannt *Blockmatrix*.

Ein Beispiel. Betrachte den Codeabschnitt einer Publikationsseite in Abb. 2.3 auf Seite 8. Unter Anderem wird das Muster

```
<br><br><font><text></font><br><font><text></font><br>
<font><text></font>
```

durch Traversierung des PAT-Trees gefunden. Jenes wird durch „Multiple String Alignment" generalisiert. Es ergibt sich aufgrund des nicht immer vorhandenen Links zu einer PDF-Datei folgendes generalisiertes Muster:[3]

```
<br><br><font><text></font><br><font><text></font><br>
<font><text></font>[<br>][<a>][<img>][<text>][</a>]
```

Jede zu diesem Muster passende Token-Folge kann in fünf Blöcke aufgeteilt werden, von denen die letzten beiden (also die Blöcke, welche den optionalen Token `[<a>]` und `[<text>]` entsprechen) nicht immer gefüllt sind. Die Blockmatrix mit den Daten aus Codeabschnitt 2.3 sieht dann wie in Tabelle 4.2 dargestellt aus. Jede Zeile enthält einen einer Publikation entsprechenden Datensatz. In den Spalten stehen die Attribute.

Wie an obigem Beispiel bereits erkennbar, enthält nicht jede Spalte der Blockmatrix nützliche Informationen. Auch steht noch nicht fest, in welche Spalte welches Attribut einsortiert wurde. Diese müssen erst noch klassifiziert werden. Die dritte Spalte enthält sogar zwei Attributtypen: die Konferenz sowie das Publikationsjahr.

4.3.3 Klassifizierung der Attribute, Muster und Websites

Nützliche Muster. Nachdem nun die extrahierten Daten in Blockmatrizen, eine je Extraktionsmuster, angeordnet wurden, muss zwischen *nützlichen Mustern* (welche gewünschte Daten extrahieren) und *nicht nützlichen Mustern* unterschieden werden. Letztere extrahieren z. B. ein Navigationsmenü, Informationen auf zu hohem Abstraktionsgrad (wie z. B. Zwischenüberschriften, welche die Daten nach Jahr gliedern) oder ganz andere regelmäßig und kontinuierlich vorkommende Datensätze auf Publikationsseiten. Es steht noch nicht einmal fest, ob die gerade untersuchte

[3] Genau genommen würde bei der Generalisierung durch die Zwischenüberschrift „2006" in Zeile 38 von Abb. 2.3 am Ende des Musters noch die optionale Token-Folge „[
][
][][][<text>][][]" entstehen. Um das Beispiel überschaubar zu halten, werden diese Token hier ignoriert.

<text>	<text>	<text>	[<a>]	[<text>]
Hartwig Holzapfel and Alex Waibel	Behavior Models for Learning and Receptionist Dialogs	Interspeech 2007, Antwerp, Belgium, 2007	pd/is2007_holzapfel.pdf	download(pdf)
Catherina R. Burghart, Ralf Mikut, and Hartwig Holzapfel	Cognition-Oriented Building Blocks of Future Benchmark Scenarios for Humanoid Home Robots	In: Gal A. Kaminka and Catherina R. Burghart (Eds.), Evaluating Architectures for Intelligence. Papers from the 2007 AAAI Workshop, Menlo Park, California: AAAI Press		
Stephan Könn, Hartwig Holzapfel, Hazim Kemal Ekenel, Alex Waibel	Integrating Face-ID into an Interactive Person-ID Learning System	International Conference on Computer Vision Systems (ICVS'07), Bielefeld, Germany, 2007	pd/icvs07-paper.pdf	download(pdf)
H. Holzapfel, A. Waibel	Providing Cognitive Functions for Interactive Learning with Speech and Multimodal Processing	Toward Cognitive Humanoid Robots Workshop at Humanoids 2006, Genoa, Italy, 2006		
C. Fügen, P. Gieselmann, H. Holzapfel, F. Kraft	Natural Human Robot Communication	Human Centered Robotic Systems, HCRS, München, Germany, 2006	pd/hcrs06.pdf	download(pdf)

Tabelle 4.2: Blockmatrix zu Codeabschnitt 2.3

Website überhaupt eine Publikationsseite ist. Weiterhin enthält die Blockmatrix eines nützlichen Musters im Allgemeinen auch Spalten, die keine sinnvollen Informationen liefern (siehe letzte Spalte in Tabelle 4.2).

Wir haben es also mit einem Klassifizierungsproblem auf drei Ebenen zu tun. Auf unterster Ebene müssen die Spalten jeder Blockmatrix in nützliche und nicht nützliche klassifiziert werden, wobei die Klassenmenge der nützlichen Spalten gerade der Potenzmenge der infrage kommenden Attributklassen entspricht. Auf der mittleren Ebene muss entschieden werden, ob das Extraktionsmuster nützlich ist, d. h. ob dessen Blockmatrix als Ganzes gewünschte Informationen beiträgt. Auf oberster Ebene ist schließlich die Frage zu beantworten, ob es sich bei der untersuchten Website um eine Publikationsseite handelt.

Hintergrundwissen. Das für die Klassifizierung verwendete Hintergrundwissen stammt aus der „Computer Science Bibiography" (DBLP) der Uni-

versität Trier [Ley07]. Diese Datenbank umfasst über 955000 Artikel auf dem Themengebiet Informatik und ist als ca. 400 MB große XML-Datei [TWWWC97] herunterladbar. Aus jener Datei lassen sich 244940 Vor- und Nachnamen von Wissenschaftlern, 207584 Wörter aus Publikationstiteln, 4391 Wörter aus Konferenznamen und Buchtiteln sowie 73 verschiedene Jahresangaben gewinnen[4] (Stand vom 22.01.2008). Auf dieser Grundlage wird zunächst für jedes aus der DBLP gewonnene Wort ein statistisches Maß für dessen Relevanz bezüglich der Klasse, in der es vorkommt, berechnet. Dafür bietet sich das beim Information Retrieval zur Gewichtung von Schlüsselwörtern oder in der Spracherkennung bei der Adaption von Sprachmodellen [Rog05, S. 305ff] eingesetzte Maß *tfidf* an. Es bestimmt die Wichtigkeit eines Wortes w für ein Dokument D_i und ist das Produkt der Faktoren *tf* (*term frequency*) und *idf* (*inverse document frequency*):

$$tf(w, D_i) = \frac{\#w \text{ in } D_i}{|D_i|} \tag{4.3}$$

Der Wert $tf(w, D_i)$ ist also Quotient aus der Anzahl der Vorkommen von w im Dokument D_i und der Größe von D_i (Gesamtanzahl der Wörter in Dokument i).

$$idf(w) = log\left(\frac{n}{\#D_j \text{ mit } D_j \text{ enthält } w}\right) \tag{4.4}$$

Als Dokument gilt hierbei jeweils eine Ansammlung von *Namen* von Wissenschaftlern, *Publikationstiteln*, *Konferenznamen und Buchtiteln* sowie von *Jahresangaben*, wie sie in der DBLP-XML-Datei stehen. n ist die Anzahl an Dokumenten. In diesem Fall gilt also $n = 4$. Ist w ein Wort, das in nahezu jedem Dokument sehr oft vorkommt - z. B. „and" - dann ist zwar $tf(w, D_i)$ relativ groß - zumindest größer als für die meisten anderen Wörter - $idf(w)$ jedoch ist nahezu null, da der Nenner des Bruches in Formel 4.4 nahe n ist. Somit ist $tfidf(w, D_i) = tf(w, D_i) \cdot idf(w) \approx 0$. Wörter, die in sehr wenigen Dokumenten vorkommen, für diese also als typisch angesehen werden können, haben einen relativ hohen *idf*-Wert.

[4] Es stellte sich heraus, dass die Extraktionsergebnisse besser sind, wenn Wörter mit weniger als 4 Zeichen von der Berechnung ausgeschlossen werden. Dann ergeben sich folgende Anzahlen: 241847 Vor- und Nachnamen, 194714 Wörter aus Publikationstiteln, 3728 Wörter aus Konferenznamen und Buchtiteln sowie 73 verschiedene Jahresangaben

Klassifizierung. Die *tfidf*-Werte aller Wörter der vier Attributtypen, welche die DBLP-XML-Datei liefert, können einmalig im Voraus berechnet werden. Danach lassen sich die t Spalten der Blockmatrix wie folgt klassifizieren: Berechne für jede Attributklasse D_i $(1 \leq i \leq 4)$ in jeder Spalte S_j $(1 \leq j \leq t)$ einen akkumulierten *tfidf*-Wert, der sich aus allen *tfidf*-Werten der Wörter dieser Spalte ergibt:

$$tfidf(S_j, D_i) := \sum_{w \in S_j} tfidf(w, D_i) \tag{4.5}$$

Ordne nun jeder Attributklasse D_i diejenige Spalte S_j zu, für die $tfidf(S_j, D_i)$ maximal ist. Um zu ermöglichen, dass eine Attributklasse gar keine Spalte zugeordnet bekommt (das ist nötig, da nicht immer alle Attribute auf der Publikationsseite aufgeführt sind bzw. die betrachtete Seite gar keine Publikationsseite sein könnte), eignet sich die Angabe eines Schwellwertes, der von $tfidf(S_j, D_i)$ überschritten werden muss, damit die Zuordnung vorgenommen wird.

Die Notwendigkeit einer weiteren Randbedingung resultiert aus der Art und Weise, wie die Extraktionsmuster entstehen. In dem Token-String „123123123123" z. B. kommt die Folge 123 vier mal vor. Allerdings hat auch 123123 zwei Vorkommen. Aus dem PAT-Tree würde also die zweite Folge als längste Wiederholung ermittelt werden. Wenn nun aber bereits die Token 123 alle gewünschten Informationen enthalten, erstreckt sich 123123 über zwei aneinandergrenzende Datensätze. Um die Wahrscheinlichkeit, dass die klassifizierten Spalten der Blockmatrix zu derselben Publikation gehören, zu erhöhen, muss also die maximale Entfernung zweier gewählter Spalten - sie sei als e_{max} bezeichnet - begrenzt werden. Wähle somit die Spalten einer Blockmatrix B innerhalb eines Bereiches der Breite e_{max} so, dass die Summe der *tfidf*-Werte aller gewählten Spalten maximal wird. Formal: Bezeichne j_i den Index der Spalte, die für Attributklasse D_i gewählt wird. Maximiere

$$tfidf(B) := \sum_{i=1}^{4} tfidf(S_{j_i}, D_i),\ 1 \leq j_i \leq t \tag{4.6}$$

unter der Randbedingung, dass für gewählte S_{j_i} und S_{j_l} $(i, l \in \{1, 2, 3, 4\}$, $i \neq l)$ gilt: $|j_i - j_l| \leq e_{max}$.

Auf diese Weise wird das Klassifizierungsproblem auf unterster Ebene gelöst. Spalten der Blockmatrix, welche eine oder mehrere Attributklassen zugewiesen bekommen haben, gelten als *nützlich* und werden weiter

betrachtet. Die anderen Spalten werden als *nicht nützlich* angesehen und deren Werte verworfen. Gleichzeitig lässt sich daraus ein Gütemaß für die gesamte Blockmatrix (und somit für das zugrunde liegende Extraktionsmuster) ableiten: *tfidf(B)* ist dafür gut geeignet. Es steigt linear mit der Anzahl der Spalten, denen ein Attributtyp zugeordnet werden konnte, und mit der Anzahl der Datensätze (den Zeilen der Blockmatrix), die durch das Muster extrahiert wurden. *tfidf(B)* wird daher im Folgenden auch als *Konfidenz* der Blockmatrix B bezeichnet.

Ein Problem ergibt sich jedoch aus der Verteilung der Daten des Hintergrundwissens. Die Anzahl der Wörter aus verschiedenen Klassen liegt in der DBLP-XML-Datei in ganz unterschiedlichen Größenordnungen. Demzufolge sind z. B. die *tfidf*-Werte der Jahresangaben bedeutend größer als diejenigen von Wörtern in Publikationstiteln. Das hat zur Folge, dass eine Blockmatrix, die nur Jahresangaben enthält, eine höhere Güte bekommt als eine Matrix, die alle Attribute bis auf die Jahreszahlen korrekt enthält. Um dem entgegenzuwirken, werden die *tfidf*-Werte je Attributtyp neu skaliert:

- Für jede Attributklasse D_i:
  - Bestimme den maximalen *tfidf*-Wert $m(D_i)$ aller Wörter der Klasse: $m(D_i) := \max_{w \in D_i} tfidf(w, D_i)$
  - Skaliere jeden *tfidf*-Wert der Klasse mit dem Reziproken des Maximums: $\forall w \in D_i : tfidf_{skaliert}(w, D_i) := \frac{1}{m(D_i)} tfidf(w, D_i)$

Dadurch wird in jeder Klasse ein maximaler *tfidf*-Wert von eins erreicht. Weil sich das relative Verhältnis der *tfidf*-Werte einer Klasse zueinander durch Multiplikation mit einem bezüglich der Klasse konstanten Faktor nicht ändert, ist es legitim, diese Skalierung gleich in die Vorausberechnung der *tfidf*-Werte einzubeziehen und von vornherein mit den skalierten Werten zu rechnen. Ersetze dazu in Gleichung 4.5 $tfidf(w, D_i)$ durch $tfidf_{skaliert}(w, D_i)$. Ideal wäre es jetzt, wenn ein Schwellwert für die in Gleichung 4.6 definierte Konfidenz einer Blockmatrix angegeben werden könnte, welcher Matrizen mit nützlichen Daten von solchen mit nicht nützlichen Daten trennt. Es sollte jedoch bedacht werden, dass je nach Anzahl an Publikationen auf einer Website und somit nach Größe der Publikationsseite selbst, die auf diesem Wege berechnete Konfidenz für verschiedene Seiten in unterschiedlichen Größenordnungen liegen kann. Wird aus der absoluten Konfidenz (*tfidf(B)*) einer Blockmatrix B und der Größe der

Website (deren Token-Anzahl) der Quotient gebildet, erhält man eine relative Konfidenz - ein Wert, der eher für eine Schwellwertfilterung geeignet sein sollte.

Resümee. Zusammengefasst extrahiert der Extractor für jedes vom Pattern Discoverer gelieferte Muster durch Übereinstimmungssuche die entsprechenden Datensätze aus dem Token-String, richtet sie aneinander aus und legt sie strukturiert in einer Blockmatrix ab. Danach klassifiziert er zunächst die Spalten der Blockmatrix in Attributtypen und nicht nützliche Spalten auf Basis von vorab berechneten und skalierten *tfidf*-Werten[5]. Durch Aufsummierung der *tfidf*-Werte aller als nützlich klassifizierter Wörter in der Blockmatrix wird schließlich ein Gütemaß berechnet, für das durch Schwellwertfilterung nützliche und nicht nützliche Blockmatrizen voneinander getrennt werden. Bleiben keine als nützlich klassifizierten Blockmatrizen übrig, wird die gerade untersuchte Website als *keine Publikationsseite* klassifiziert. Im anderen Fall werden die Datensätze der nützlichen Blockmatrizen vereinigt und ergeben die letztendliche Ausgabe des Extractors.

[5] An dieser Stelle sollte noch erwähnt werden, dass diese Art von Klassifizierung für die extrahierten Links (aus <a>-Token) nicht notwendig ist. Dass es sich um Links handelt, steht bereits fest. Die Klassifizierung in Links zu PDF-Dateien, Links zu HTML-Seiten und sonstige Links kann anhand der Dateieendung vorgenommen werden.

5 Evaluation

Allgemein ist die Menge der Daten, mit denen das System letztendlich arbeiten muss, nicht abgeschlossen. Es kann nicht vorhergesagt werden, ob die Ergebnisseite einer Suchmaschinenanfrage morgen noch dieselben Websiten referenziert, wie heute. Daher ist es sinnvoll, das System auf verschiedene Arten zu evaluieren, um einerseits unter kontrollierten Bedingungen wiederholbare Ergebnisse zu bekommen und aber andererseits ein dem praktischen Einsatz ähnliches Szenario betrachten zu können.

Somit sind folgende Evaluationsszenarien denkbar:

- Szenario 1: Abgeschlossene Datenmenge
  Es werden bestimmte Publikationsseiten als lokal gespeicherte HTML-Dateien vorgegeben. Die Aufgabe besteht darin, alle darin befindlichen Publikationen zu extrahieren.

- Szenario 2: Suchmaschinenergebnisse
  Zunächst werden bestimmte Publikationsseiten in Form von URLs im Internet definiert. Dann werden dem System Vor- und Nachnamen der entsprechenden publizierenden Personen übergeben, mit dem Ziel, die definierten Publikationsseiten zu finden.

- Szenario 3: Offene Datenmenge
  Hierbei werden Vor- und Nachnamen von Personen sowie eine Jahreszahl vorgegeben. Aufgabe ist es nun, alle Publikationen der entsprechenden Personen des spezifizierten Jahres zu finden. Dieses Szenario soll das Verhalten des Systems in realitätsnahem Betrieb zeigen. Die Auswahl wird auf ein Jahr beschränkt, um trotz der nicht abgeschlossenen Menge an Websiten, die zu untersuchen sein könnten, die Zieldatensätze im Vorraus festlegen zu können.

Die folgenden Abschnitte beschreiben Szenarium eins und zwei genauer und bewerten deren Ergebnisse. Szenario drei soll eine Möglichkeit aufzeigen, wie solch ein System praxisnah evaluiert werden kann. Im Rahmen dieser Arbeit wird allerdings hauptsächlich auf das erste Szenario eingegangen, da es unter vollständig kontrollierten Bedingungen abläuft und direkt aufzeigt, wo die Probleme des gewählten Ansatzes liegen.

5.1 Szenario 1: Abgeschlossene Datenmenge

In diesem Szenario werden bestimmte Publikationsseiten, die lokal als HTML-Dateien vorliegen, vorgegeben. Die Aufgabe besteht darin, möglichst alle aufgeführten Publikationen zu extrahieren. Dabei gilt eine Publikation als gefunden, wenn alle Autoren und der Titel gefunden und einander korrekt zugeordnet wurden.

Die Daten bestehen aus je 10 HTML-Dateien „Trainings"- und Evaluationsdaten. Wie schon mehrfach erwähnt, ist der gewählte Ansatz weder ein maschinelles Lernverfahren, noch ein System, welches annotierte Trainingsbeispiele benötigt, um funktionieren zu können. Es existieren jedoch einige Parameter, deren Feineinstellung die Extraktionsgüte beeinflusst. Aus diesem Grund wurden die Daten in zwei disjunkte Teilmengen unterteilt. Die „Trainings"-Daten waren während der gesamten Entwicklungszeit verfügbar und dienten dazu, Parameter des PAT-Trees, wie z. B. die Mindestlänge zu suchender Muster, die Mindestwiederholungsanzahl der Vorkommen im Token-String oder Schwellwerte für Varianz und Dichte, sowie die Schwellwerte bei der Klassifikation der Inhalte der Blockmatrizen (siehe Abschnitt 4.3.3) einzustellen. Die HTML-Dateien der Evaluationsdaten hingegen wurden zwar zu Beginn schon festgelegt, jedoch während der Entwicklung und Konfiguration des Systems nicht betrachtet. Sie sollen dazu dienen, eine Aussage über die Generalisierungsfähigkeit des Systems über ungesehene Daten treffen zu können.

Für beide Datenmengen wurde zuvor die Grundwahrheit, also die tatsächlich auf den Publikationsseiten befindlichen Informationen manuell herausgeschrieben. Jene Daten wurden zur Evaluation mit den extrahierten Daten verglichen und die Werte *Precision* sowie *Recall* berechnet:

$$Precision = \frac{\#Treffer}{\#Treffer + \#Fehlalarme} \tag{5.1}$$

$$Recall = \frac{\#Treffer}{\#zu\ extrahierender\ Publikationen} \tag{5.2}$$

Diese beiden Werte geben im Allgemeinen nur in Kombination Aufschluss über die Güte eines Klassifikators. Würde ein System z. B. auf einer Website mit 100 Publikationen genau eine korrekt und keine weitere finden, hätte Precision den Wert 1.0, also das Beste, was erreicht werden kann, da ja keine Fehler gemacht wurden. Erst durch die Angabe des Recalls - gelegentlich auch als *Retrieval Rate* bezeichnet - von 0.01 wird klar, dass jedoch gerade einmal 1% der gesuchten Publikationen gefun-

den wurde. Andersherum ist es möglich, dass ein Klassifikator zwar einen Recall-Wert von 1.0 erreicht, also alles extrahiert, was es zu extrahieren gilt, dabei jedoch allerhand Daten ausgibt, die gar keine Publikationsattribute sind. Das senkt wiederum den Wert der Precision.

Ein aus Precision und Recall berechenbares Maß, das jene beiden im Allgemeinen gegenläufigen Werte zusammenfasst, ist das *F-Maß* (F-measure). Es ergibt sich als harmonisches Mittel von Precision und Recall:

$$F\text{-}measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5.3}$$

5.1.1 Resultate auf den „Trainingsdaten"

In Tabelle 5.1 sind die Ergebnisse auf den zehn Trainingsdateien[1] aufgeführt. Eine Publikation gilt hierbei als erkannt, wenn all ihre Autoren sowie ihr Titel korrekt extrahiert und einander richtig zugeordnet wurden. Die Publikationsseiten der „Trainingsdaten" wurden bewusst so ausgewählt, dass sie einander möglichst unähnlich sind, um ein breites Spektrum abdecken zu können. Die zweite Spalte der Tabelle (#) gibt die Anzahl zu extrahierender Publikationen der jeweiligen Seite an. Die Werte in den mit (a) gekennzeichneten Spalten beziehen sich auf das erste Extraktionsmuster bei absteigender Sortierung nach deren Konfidenz, also der *tfidf*-Summe über alle Daten der Blockmatrix.

Es ist ersichtlich, dass die Werte für Precision und Recall sehr stark auseinandergehen. Aus vier Websiten konnten alle Publikationen fehlerlos

[1] Publikationsseiten von
Hartwig Holzapfel (`http://isl.ira.uka.de/~hartwig/pubM.html`),
interACT (Interactive Systems Laboratories, `http://isl.ira.uka.de/index.php?id=292`),
Rüdiger Dillmann (aus dem Jahr 2006, `http://wwwiaim.ira.uka.de/index_id-372_mod409tab-1_mod447tab-0.html`),
Dorothea Wagner (`http://i11www.iti.uni-karlsruhe.de/algo/people/index.php?algouser=wagner&pagetype=publications`),
Rainer Stiefelhagen (`http://isl.ira.uka.de/~stiefel/rainer.publications.html`),
Björn Hein (`http://wwwipr.ira.uka.de/~hein/?language=en&clicked=Publications`),
Uwe Brinkschulte (`http://ipr.ira.uka.de/292.php`),
Walter F. Tichy (`http://www.ipd.uka.de/Tichy/people.php?id=15`),
cpLab (Laboratory for Computational Perception and Statistical Learning, `http://www.cnbc.cmu.edu/cplab/publications.html`),
Chun-Nan Hsu (`http://kaukoai.iis.sinica.edu.tw/~chunnan/mypublications.html`)

Publikationsseite	#	Precision		Recall		Probleme
		(a)	(b)	(a)	(b)	
Hartwig Holzapfel	23	1	0.96	1	1	-
interACT-Publikationen	489	1	0.7	0.57	0.72	d, m
Rüdiger Dillmann (2006)	39	1	1	1	1	-
Dorothea Wagner	143	0	0	0	0	a, v
Rainer Stiefelhagen	86	1	1	0.28	0.33	d, m, ü
Björn Hein	11	1	1	1	1	-
Uwe Brinkschulte	23	0	0	0	0	a, m, n, ü
Walter F. Tichy	40	1	0.88	1	1	-
cpLab	52	0	0.23	0	0.21	a, ü
Chun-Nan Hsu	60	1	0.61	0.80	0.8	d, m

Tabelle 5.1: Extraktionsergebnisse des Musters mit höchster Konfidenz ((a)-Spalten) und der Vereinigung der ersten zwei Muster ((b)-Spalten) auf bekannten Daten

extrahiert werden. Demgegenüber stehen drei Publikationsseiten, von denen durch das erste Extraktionsmuster keinerlei korrekte Informationen gewonnen werden konnten. Aus den verbleibenden drei Seiten extrahiert das erste Muster einen gewissen Teil der Publikationen, macht dabei - zu sehen an dem Precision-Wert von 1 - allerdings keine Fehler. Die in der fünften Spalte angegebenen Kürzel bezeichnen die jeweils auftretenden Probleme, welche sich durch Analyse der gefundenen Muster und extrahierten Daten zeigen:

- -: Kein Problem; ein Muster extrahiert alle verfügbaren Publikationen korrekt.
- a: Ein Attribut (Autoren oder Titel) wird falsch klassifiziert.
- d: Das gefundene Muster deckt nicht alle Publikationen ab.
- m: Vereinigung der Daten mehrerer Muster notwendig.
- n: Namen sind über mehrere getrennte `Text(_)`-Token verteilt.
- s: Mehrere Attribute stehen in einer Spalte der Blockmatrix.
- ü: Überlappung mehrerer Publikationen.
- v: Zu hohe Varianz der Publikationsangaben.

Die drei Publikationsseiten, aus denen keine Publikationen korrekt extrahiert werden konnten, bedürfen einer näheren Betrachtung.

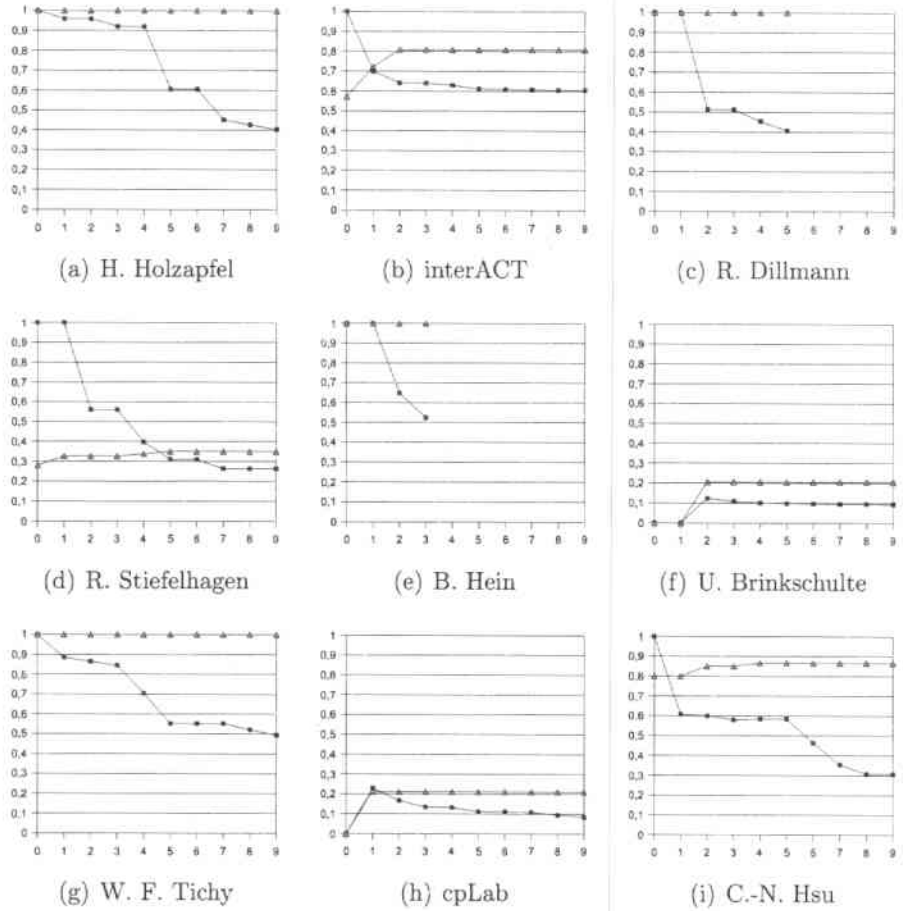
Auf R. Stiefelhagens Seite sind die Publikationen nach zwei verschiedenen Mustern aufgeführt, welche zum Teil gemischt auf der Seite vorkommen. Eines der beiden Muster wird korrekt erkannt und die Attribute richtig klassifiziert. Das andere Muster - bei dem im Gegensatz zu dem

ersten Titel und Autoren ihre Plätze in der Reihenfolge tauschen - wird ebenfalls gefunden. Hier gibt es aber wieder Klassifikationsfehler. Zwar werden Titel tatsächlich als Titel und Autoren ebenfalls als solche klassifiziert, jedoch werden die Attribute unmittelbar aufeinanderfolgender Publikationen einander falsch zugeordnet.

Die Publikationsangaben auf D. Wagners Seite weisen in ihrer Gesamtheit eine zu hohe Varianz auf, sodass durch das Multiple String Alignment nicht alle Titel in eine Spalte und alle Konferenznamen bzw. Buchtitel in eine andere Spalte der Blockmatrix abgelegt werden können. Die hohe Varianz ist vor allem darin begründet, dass Teile des Publikationstitels oder anderer Attribute kursiv geschrieben sind. Durch die `<i>`-Tags werden die Attribute in einzelne Stücke zerteilt, von denen jedes in einer anderen Spalte der Blockmatrix angeordnet wird. Dadurch ist es nicht einmal möglich, manuell ein Extraktionsmuster anzugeben, welches auf einen hinreichend großen Bereich der Publikationen passt und dabei alle Attribute korrekt extrahiert.

Die Publikationsseite des Laboratory for Computational Perception and Statistical Learning enthält auch die Abstracts der Publikationen sowie BibTeX-Code in Text-Tokens. Abstracts benutzen ein dem Titel sehr ähnliches Vokabular und sammeln aufgrund ihrer relativ hohen Wortanzahl viele *tfidf*-Werte auf. Das führt dazu, dass stets diejenige Spalte der Blockmatrix, welche die Abstracts enthält, als Titelspalte klassifiziert wird. Dieses Verhalten kann vermieden werden, indem ein Schwellwert für die maximale Wortanzahl eines Text-Tokens festgelegt wird, bei dessen Überschreitung der Text als zu lang angesehen und von der Berechnung der *tfidf*-Werte ausgeschlossen wird. Dem zweiten Problem lässt sich nicht so einfach aus dem Weg gehen. Die Autoren, die in dem BibTeX-Code innerhalb des BibTeX-Keys stehen, erreichen einen höheren summierten *tfidf*-Wert als diejenigen, die in für sich in Text-Tokens stehen.

In Abbildung 5.1 ist das Verhalten von Precision und Recall bei sukzessiver Vereinigung der von den nach Konfidenz absteigend sortierten Muster extrahierten Daten für die einzelnen Publikationsseiten abgetragen (Seiten, bei denen Precision und Recall stets null bleiben, sind nicht dargestellt). Wie man leicht sieht, sind diese Verläufe sehr unterschiedlich. Um eine allgemeine Aussage darüber treffen zu können, wie viele Muster im Durchschnitt für die Extraktion von einer Publikationsseite verwendet werden sollten, wurde je ein mit der Anzahl der auf der jeweiligen Seite befindlichen Publikationen gewichteter Durchschnitt von Precision und Recall und daraus das F-Maß berechnet. Das Ergebnis ist in Abbildung 5.2 zu sehen. Daraus kann abgelesen werden, dass nach Hinzunahme der



(a) H. Holzapfel (b) interACT (c) R. Dillmann

(d) R. Stiefelhagen (e) B. Hein (f) U. Brinkschulte

(g) W. F. Tichy (h) cpLab (i) C.-N. Hsu

Abbildung 5.1: Verlauf von Precision (blaue Quadrate) und Recall (rote Dreiecke) bei Hinzunahme von Datensätzen weiterer Extraktionsmuster auf bekannten Daten

Daten des zweiten Musters Recall zwar noch etwas steigt, Precision jedoch stärker fällt, was sich auch durch Verringerung des F-Maßes zeigt. Es liegt also nahe, für eine automatische Extraktion stets die Daten der ersten zwei Muster zu verwenden. Die sich für diesen Fall auf den „Trainingsdaten" ergebenden Werte für Precision und Recall stehen in Tabelle 5.1 in den mit (b) beschrifteten Spalten. Die Angabe eines Schwellwertes für die in Abschnitt 4.3.3 beschriebene relative Konfidenz einer Blockmatrix (und somit des Extraktionsmusters) ist trotz Normierung mit der Publikationsseitengröße nicht so einfach möglich. Für Muster, die nützliche

Daten extrahieren, schwankt jener Wert zwischen 0,01 und 0,16. Ob eine nichtlineare Funktion diese Klassifizierung vornehmen kann, müsste weiter untersucht werden. Es wäre wahrscheinlich sinnvoll, auch die Wortanzahl einzubeziehen.

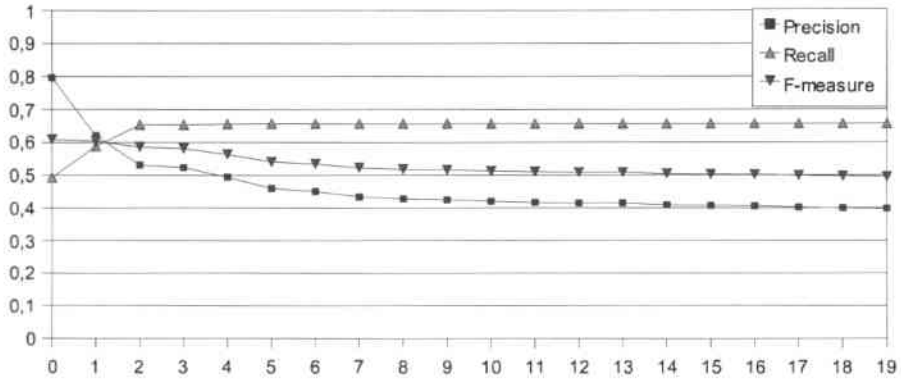


Abbildung 5.2: Verlauf der gewichteten Mittelwerte von Precision, Recall und F-Measure bei sukzessiver Vereinigung der Datensätze der ersten 20 Muster aus bekannten Publikationsseiten

5.1.2 Resultate auf ungesehenen Daten

Tabelle 5.2 zeigt die Extraktionsresultate des Musters mit höchster Konfidenz auf den zehn bisher unbetrachteten Dateien[2]. Die Kürzel in der

[2]Publikationsseiten von
ISAS (Intelligente Sensor-Aktor-Systeme, `http://isas.uka.de/site_de/publications/index.html`)
Tamim Asfour (`http://i61www.ira.uka.de/users/asfour/publications.html`)
Robert Görke (`http://i11www.iti.uni-karlsruhe.de/members/index.php?algouser=rgoerke&pagetype=publications`)
Tom Gelhausen (`http://www.ipd.uka.de/Tichy/people.php?id=31`)
Martina Zitterbart (`http://www.tm.uka.de/forschung/2004/pubs.html`)
ITI (Institut für Theoretische Informatik, `http://algo2.iti.uni-karlsruhe.de/publications.php`)
Mark R. Cutkosky (`http://www.ri.cmu.edu/people/cutkosky_mark.html`)
Stanford Telerobotics Laboratory (`http://telerobotics.stanford.edu/publications/index.html`)
System Architecture Group (`http://i30www.ira.uka.de/research/publications/papers/index.php?lid=en`)

Spalte „Probleme" tragen dieselbe Bedeutung, wie schon im vorigen Abschnitt. Es ist ersichtlich, dass das erste Extraktionsmuster hier wesentlich schlechter funktioniert. Etwas besser fallen die Ergebnisse aus, wenn die Restriktion, alle Autoren zu einem Titel zu extrahieren, fallen gelassen wird und eine Publikation schon dann als gefunden gilt, wenn mindestens ein Autor korrekt zugeordnet wird. Die sich in diesem Fall ergebenden Werte sind in Klammern notiert. Tabelle 5.3 zeigt die Ergebnisse für die vereinigten Daten der ersten zwei Extraktionsmuster, also diejenigen, die durch die im vorigen Abschnitt festgelegte Musteranzahl entstehen. In Abbildung 5.3 ist erkennbar, dass für einige Seiten durch spätere Muster doch noch korrekte Daten extrahiert werden, während für die Publikationsseiten, auf denen schon das erste Muster gut extrahiert hat, durch zunehmend fehlerhafte Daten die Precision-Werte abnehmen. Insgesamt betrachtet ist aber, wie schon auf den Trainingsdaten, die Vereinigung der von den ersten zwei Mustern extrahierten Daten das Optimum bezüglich des F-Maßes, wie in Abbildung 5.4 auf Seite 45 zu sehen.

Publikationsseite	#	**Precision**	**Recall**	**Probleme**
ISAS-Publikationen	143	0	0	a, v
Tamim Asfour	40	0	0	a, n
Robert Görke	22	0	0	a
Tom Gelhausen	7	1	1	-
Martina Zitterbart (2004)	16	0	0	a, d, v
ITI-Publikationen	62	0.23 (0.56)	0.19 (0.47)	a, m
Mark R. Cutkosky	7	0	0	a, n
Stanford Telerobotics Lab	70	0.74 (1)	0.74 (1)	s
System Architecture Group	277	0	0	a
FZI - PDE-Group	64	1	1	-

Tabelle 5.2: Extraktionsergebnisse des Musters mit höchster Konfidenz auf unbekannten Daten

Im folgenden werden die Publikationsseiten, bei denen auch das zweite Muster keine korrekten Daten liefert, genauer betrachtet.

Die ISAS-Publikationsseite enthält Informationen zu jeder Publikation in mehrfacher Ausführung. Auch Abstracts sind aufgeführt. Diese stehen jedoch nicht jeder für sich innerhalb eines Text-Tokens, sondern sind jeweils durch `<br>`-Token (Zeilenumbrüche) in einzelne kleine Abschnitte

FZI - PDE-Group (Process- and Datamanagement in Engineering, `http://www.fzi.de/pde/eng/publikationen.php`)

Publikationsseite	#	Precision	Recall
ISAS-Publikationen	143	0	0
Tamim Asfour	40	0	0
Robert Görke	22	0	0
Tom Gelhausen	7	0.41	1
Martina Zitterbart (2004)	16	0	0
ITI-Publikationen	62	0.41 (0.57)	0.63 (0.87)
Mark R. Cutkosky	7	0	0
Stanford Telerobotics Lab	70	0.39 (0.52)	0.74 (1)
System Architecture Group	277	0.71	0.66
FZI - PDE-Group	64	0.53	1

Tabelle 5.3: Ergebnisse bei Vereinigung der von den ersten zwei Mustern extrahierten Daten bei unbekannten Publikationsseiten

zerteilt. Unser Ansatz schlägt somit aus zwei Gründen fehl: Einerseits ist dadurch, dass nicht jeder Abstract gleich lang ist, die Anzahl der Abschnitte sehr unterschiedlich, was dem korrekten Extraktionsmuster eine relativ hohe Varianz gibt. Andererseits führt das in diesen Abschnitten verwendete Vokabular, wie schon auf der cpLab-Seite bei den Trainingsdaten, zur Klassifikation jener Abschnitte als Titel. Anders als auf der cpLab-Seite schafft hier jedoch, aufgrund der starken Zerteilung, die Begrenzung der Wortanzahl eines Text-Tokens keine Abhilfe.

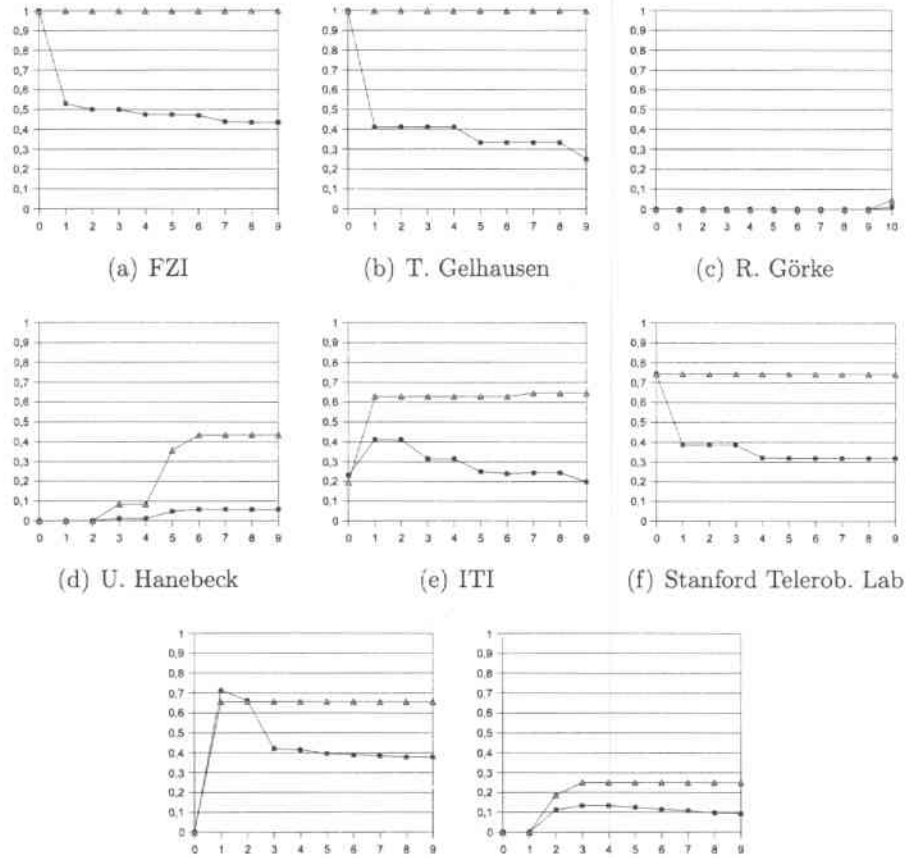
Auf T. Asfours Publikationsseite stellen die über mehrere Text-Token verteilten Autoren das größte Problem dar. Dadurch, dass sie in verschiedenen Spalten der Blockmatrix angeordnet werden, erreicht keine von ihnen eine *tfidf*-Summe, die zur korrekten Klassifikation notwendig wäre. Somit fehlen den extrahierten Datensätzen die Autoren vollständig.

Auf der Publikationsseite von R. Görke werden alle Autoren korrekt klassifiziert. Problematisch ist jedoch, dass vor den Autoren die Worte „Joint work with" stehen. Aufgrund der Beschaffenheit des Hintergrundwissens erzielt die Spalte der Blockmatrix, in welcher die Autoren gemeinsam mit jenen Worten stehen, eine höhere *tfidf*-Summe als die Spalte mit den Titeln, was in einer Fehlklassifikation der Titel-Spalte resultiert.

Von M. Zitterbarts Publikationsseite des Jahres 2004 extrahiert das Muster mit der höchsten Konfidenz eine Liste mit Links zu den Publikationsseiten weiterer Jahre. Das liegt vor Allem daran, dass jene Liste länger ist als die eigentliche Publikationsliste und somit eine höhere Konfidenz erreicht als die Daten, welche von anderen Mustern extrahiert werden. Dazu kommt eine recht hohe Varianz der HTML-Tags in den Publikationsanga-

ben, weshalb ein einziges Muster nicht alle Publikationen mit einem Mal extrahiert.

In M. R. Cutkoskys Publikationsangaben sind, wie bereits einige Male vorgekommen, die einzelnen Autoren durch HTML-Tags voneinander getrennt. Demzufolge werden sie in verschiedene Spalten der Blockmatrix eingeordnet und erreichen insgesamt keinen *tfidf*-Wert, der hoch genug ist, um sie als Autoren zu klassifizieren.



(a) FZI (b) T. Gelhausen (c) R. Görke

(d) U. Hanebeck (e) ITI (f) Stanford Telerob. Lab

(g) SysArch Group (h) M. Zitterbart

Abbildung 5.3: Verlauf von Precision (blaue Quadrate) und Recall (rote Dreiecke) bei Hinzunahme von Datensätzen weiterer Extraktionsmuster auf unbekannten Daten

Insgesamt bleiben die Evaluationsergebnisse hinter unseren Erwartungen zurück. Aus manchen Publikationsseiten wird alles korrekt extrahiert,

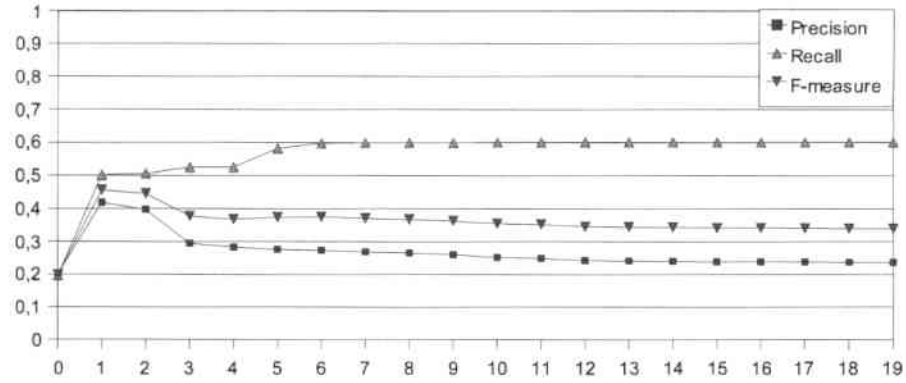


Abbildung 5.4: Verlauf der gewichteten Mittelwerte von Precision, Recall und F-Measure bei sukzessiver Vereinigung der Datensätze der ersten 20 Muster aus unbekannten Publikationsseiten

aus anderen wiederum nichts. Die meisten Fehler werden durch eine Fehlklassifikation der Attribute hervorgerufen. Zweitgrößtes Problem ist eine zu feine Granularität der Textabschnitte, die durch die HTML-Tags gebildet werden.

5.2 Szenario 2: Suchmaschinenergebnisse

Dieses Szenario betrifft lediglich den ersten Teil einer Webrecherche nach Abb. 2.1 auf Seite 4. Es werden zunächst bestimmte Publikationsseiten in Form von URLs im Internet festgelegt. Danach werden unter Einbezug der Vor- und Nachnamen der entsprechenden publizierenden Personen Suchanfragen an verschiedene Internet-Suchmaschinen gestellt, mit dem Ziel, die definierten Publikationsseiten zu finden. Damit soll untersucht werden, ob es überhaupt gelingt, Publikationsseiten automatisch ausfindig zu machen, und wenn ja, wie viele Websiten im Durchschnitt untersucht werden müssen, bis die richtige dabei ist.

Für diese Evaluation wurden die Namen und URLs der bereits in Abschnitt 5.1 eingesetzten Publikationsseiten verwendet und jeweils drei Anfragen an Google (`http://www.google.de/`), Microsoft Live Search (`http://search.live.com/`) sowie Yahoo (`http://de.search.yahoo.com/`) gestellt:

- „Vorname Nachname“ publications (Tabelle 5.4)

- „Vorname Nachname“ publikationen (Tabelle 5.5)
- Vorname Nachname publications (Tabelle 5.6)

Unter den Ergebnisseiten wurde herausgesucht, an welcher Position in der Rangfolge die bereits bestimmte Website steht. Eine Publikationsseite gilt dabei als gefunden, wenn das Suchergebnis mindestens auf eine Seite derselben Domain verweist, auf welcher durch Verfolgung höchstens eines Links die definierte URL erreicht wird. Solch eine Aufweichung ist aus zwei Gründen notwendig. Erstens unterteilen manche Autoren ihre Publikationen nach Jahren auf und schreiben jedes Jahr auf eine extra Seite. So lässt sich nicht eindeutig eine Seite als „die Publikationsseite“ definieren. Zweitens sind einige der oben definierten URLs Seiten, die im Normalfall innerhalb eines Frames einer übergeordneten Seite angezeigt werden.

Ergebnispositionen, welche direkt auf die spezifizierte URL verweisen, sind mit einem Stern (*) gekennzeichnet, solche, die als untergeordnete Ergebnisse einer Position aufgeführt werden, mit einem „b“. Des Weiteren ist in den Spalten (der Tabellen 5.4, 5.5 und 5.6) mit der Sub-Überschrift #S die jeweilige Anzahl der Websiten angegeben, welche mindestens drei Publikationsangaben der gesuchten Person enthalten und im Ranking vor der spezifizierten Seite stehen. Es wurden maximal die ersten 40 Ergebnispositionen betrachtet. Einträge, denen eine (2) folgt, benötigen eine Link-Verfolgung der Tiefe 2 innerhalb derselben Domain, um zu der spezifizierten Seite zu gelangen.

Wie die Werte in den Tabellen zeigen, erzielt die Suche bei den zehn Testseiten durchschnittlich die besten Ergebnisse, wenn bei der Anfrage Vor- und Nachname gemeinsam in Anführungszeichen stehen und von dem englischen Wort *publications* gefolgt werden. In diesem Fall lieferte von den drei verglichenen Suchmaschinen Microsoft Live Search die besten Resultate. Wird anstelle des Englischen das deutsche Wort *Publikationen* angefügt, bringt dies unter den gestellten Anfragen die durchschnittlich höchste Fehleranzahl, d. h. Fälle, in denen die gesuchte Website nicht unter den ersten 40 Suchergebnissen aufgeführt wird. Allerdings schneidet Google hierbei noch besser als im erstgenannten Fall ab.

Die für dieses Evaluationsszenario betrachtete Anzahl Publikationsseiten ist zwar sehr klein und somit nicht unbedingt als repräsentativ anzusehen. Jedoch zeigt jene Stichprobe, dass eine automatische Webrecherche prinzipiell möglich ist. Für 60% der Anfragen steht die gesuchte Seite sogar innerhalb der ersten drei Ergebnispositionen.

Publikationsseite	Google		MS Live		Yahoo		Arith.
	Rang	#S	Rang	#S	Rang	#S	Mittel
Hartwig Holzapfel	1*	-	1*	-	-	-	
Rüdiger Dillmann (2006)	39	6	17	4	3	-	
Dorothea Wagner	3*	1	1b	-	1 / 2*	-	
Rainer Stiefelhagen	1b*	1	1*	-	1	-	
Björn Hein	8(2)	1	2	-	-	-	
Walter F. Tichy	2	1	2*	-	2*	1	
Chun-Nan Hsu	18	10	4*	1	4*	1	
Tamim Asfour	1b*	-	1*	-	1*	-	
Robert Görke	1	-	1	-	1*	-	
Tom Gelhausen	1	-	1*	-	1	-	
# direkte Treffer	4		6		5		5
# Treffer (1 Link)	9		10		8		9
# Fehlschläge	1		0		2		1

Tabelle 5.4: Suchergebnisse bei Anfrage von: „Vorname Nachname" publications

Publikationsseite	Google		MS Live		Yahoo		Arith.
	Rang	#S	Rang	#S	Rang	#S	Mittel
Hartwig Holzapfel	1*	-	-	-	-	-	
Rüdiger Dillmann (2006)	2	1	1b	1	3	-	
Dorothea Wagner	3	1	1	-	4	-	
Rainer Stiefelhagen	-	5	-	-	-	-	
Björn Hein	1*	-	1*	-	1*	-	
Walter F. Tichy	1*	-	-	-	2(2)	-	
Chun-Nan Hsu	1*	-	-	-	-	-	
Tamim Asfour	1*	-	5(2)	-	-	1	
Robert Görke	1	-	1*	-	1	-	
Tom Gelhausen	1*	-	1(2)	-	1(2)	-	
# direkte Treffer	6		2		1		3
# Treffer (1 Link)	9		4		4		5.67
# Fehlschläge	1		6		6		4.33

Tabelle 5.5: Suchergebnisse bei Anfrage von: „Vorname Nachname" publikationen

Publikationsseite	Google		MS Live		Yahoo		Arith.
	Rang	#S	Rang	#S	Rang	#S	Mittel
Hartwig Holzapfel	1*	-	-	-	-	-	
Rüdiger Dillmann (2006)	37	7	9	2	3	-	
Dorothea Wagner	9*	3	1b	-	1 / 2*	-	
Rainer Stiefelhagen	1b*	1	1*	-	1	-	
Björn Hein	-	-	1*	-	-	-	
Walter F. Tichy	2	1	2*	1	2*	1	
Chun-Nan Hsu	5	4	4*	2	4*	1	
Tamim Asfour	1b*	-	1*	-	1*	-	
Robert Görke	-	4	1	-	1*	-	
Tom Gelhausen	4(2)	-	1*	-	1 / 2*	-	
# direkte Treffer	4		6		6		5.33
# Treffer (1 Link)	7		9		8		8
# Fehlschläge	3		1		2		2

Tabelle 5.6: Suchergebnisse bei Anfrage von: Vorname Nachname publications

6 Diskussion und Ausblick

6.1 Erkenntnisse

Die vorliegende Arbeit hat gezeigt, dass es prinzipiell möglich ist, die Struktur von Publikationsseiten im Internet als Basis zur automatischen Extraktion von Informationen heranzuziehen. Die Herausforderung liegt allerdings darin, die Attribute der extrahierten Datensätze korrekt zu klassifizieren und einander zuzuordnen.

6.2 Extraktion der Kopfdaten wissenschaftlicher Publikationen aus PDF-Dokumenten

Es wurde beschrieben, wie auf effizientem Wege Attribute von Publikationen aus Websites extrahiert werden können. Dabei wurden die Attribute *Autoren* (samt *Vor-* und *Nachnamen*), *Titel* und *Jahr* der Veröffentlichungen, die *Konferenz* bzw. der *Buchtitel*, unter der/dem veröffentlicht wurde, sowie jeweils ein *Link* zu einer *PDF-Datei* oder einer *weiteren Website* gesammelt.

Es fehlen noch die Attribute *Zugehörigkeit* der Autoren zu Hochschulen, deren *E-Mail-Adressen* und der *Abstract* der jeweiligen Publikation. Jene Attribute sind häufig nur im Kopf des zur Veröffentlichung gehörenden PDF-Dokuments zu finden. Für deren Extraktion existieren allgemein folgende Ansätze:

Wie für die Informationsextraktion aus Websites, wie eingangs beschrieben, existieren auch zur Lösung des Extraktionsproblems aus PDF-Dokumenten statische regelbasierte Verfahren auf der einen und lernende Verfahren auf der anderen Seite. Zu den lernenden Verfahren gehören *Hidden Markov Modelle* (HMMs) [Lee97, SMR99], *Support Vector Machines* (SVMs) [HGM+03] sowie *Conditional Random Fields* (CRFs) [PM04], um nur die prominentesten zu nennen. Wie für maschinelle Lernverfahren üblich, benötigt jedes von ihnen eine recht große Menge repräsentativer an-

notierter Trainingsbeispiele, um über hinreichend viele wissenschaftliche Publikationen generalisieren zu können. Auf der Seite der statischen regelbasierten Verfahren stehen fest vorgegebene endliche Automaten. Jene entsprechen regulären Ausdrücken und können bei geringem Implementierungsaufwand ein mächtiges Extraktionswerkzeug darstellen, wenn bereits einige Anhaltspunkte bekannt sind.

Alle genannten Verfahren haben ihre Vor- und Nachteile. Lernende Verfahren treffen z. B. keine a-priori Annahme über die Struktur, benötigen dafür jedoch eine riesige Menge an Lernbeispielen, um vernünftig zu funktionieren. Bei regelbasierten Verfahren hingegen werden die Regeln manuell festgelegt. Wenn jedoch, wie in unserem Fall, Autoren und Titel einer Publikation bereits bekannt sind, sollten reguläre Ausdrücke genügen, um auch noch die restlichen Attribute zu extrahieren. Abbildung 6.1 veranschaulicht diesen Sachverhalt anhand der Kopfdaten einer typischen Publikation.

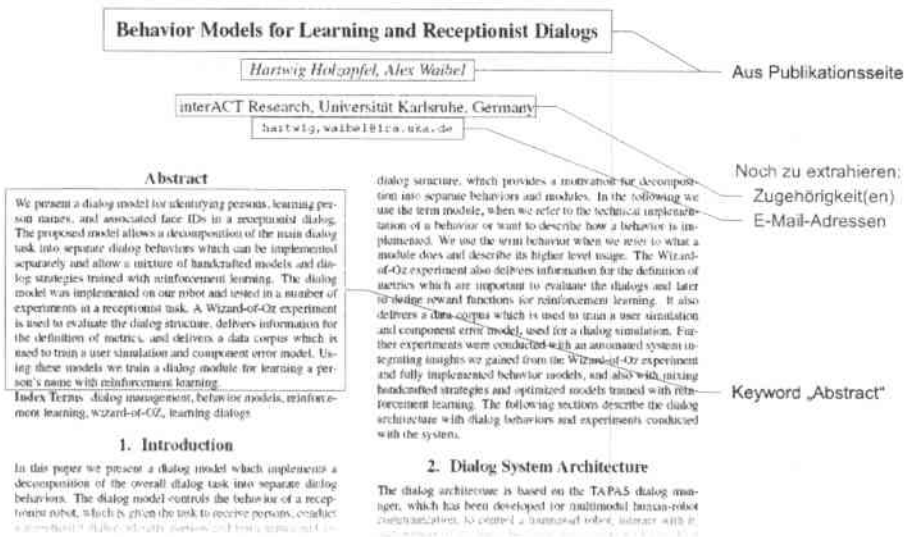


Abbildung 6.1: Eine typische Publikation im PDF-Format

Vorausgesetzt, Titel und Autoren sind von der Publikationsseite korrekt extrahiert worden, bieten diese bereits Anhaltspunkte für die restlichen Attribute. Ein großer Vorteil ist es bereits, dass jene Textstellen nicht mehr als andere Attribute infrage kommen. Weiterhin lässt sich aufgrund der syntaktischen Restriktionen E-Mail-Adressen mit Hilfe regulärer Ausdrücke finden. Der Abstract ist durch Suche nach dem Schlagwort „Ab-

stract" an deren Beginn und Wörter wie „Index Terms", „Keywords" oder „Introduction" am Ende leicht identifizierbar. Die korrekte Zuordnung der Autoren zu ihren Zugehörigkeiten und E-Mail-Adressen dürfte jedoch eine Herausforderung darstellen.

6.3 Weiterführende Möglichkeiten

Wie im vorigen Abschnitt beschrieben, könnten die aus einer Publikationsseite gewonnenen Informationen als Vorwissen zur Extraktion weiterer Attribute aus den entsprechenden PDF-Dokumenten genutzt werden. Umgekehrt ergäbe sich eine weitere Herangehensweise zur Fusion der Informationen aus den verschiedenen Quellen: Man könnte zunächst mit einem geeigneten Ansatz einige Attribute aus den PDF-Dokumenten extrahieren und die so gewonnenen Informationen für die Auswahl der Extraktionsmuster auf der Publikationsseite heranziehen, also ein sog. *Rescoring* des Gütemaßes der Muster durchführen. Es könnte sich außerdem vorteilig auswirken, mehr linguistisches Wissen einfließen zu lassen.

Eine Frage, die diese Arbeit nicht zu beantworten vermag, ist die nach der eindeutigen Identität einer Person. Bisher werden alle Personen mit gleichem Vor- und Nachnamen als dieselbe Person betrachtet. Die Trennung zweier Personen gleichen Namens ließe sich durch Betrachtung derer Koautoren oder über eine Analyse der Publikationsthemen, welche auf Grundlage des Abstracts gefunden werden könnten, vollziehen. Eine weitere Möglichkeit wäre, die Schnittstelle zur übergeordneten Schicht - dem Dialogsystem - zu erweitern. So könnte man anstelle einer einzigen gemeinsamen CSV-Datei eine je untersuchter Publikationsseite zurückliefern und die Identität durch eine gezielte Frage im Dialog klären.

Weiterhin wäre es möglich, das verwendete Hintergrundwissen durch Integration der extrahierten Publikationen zu erweitern. In diesem Zuge müsste ein Konfidenzmaß eingeführt werden, welches die Verlässlichkeit der jeweiligen Information angibt. Für Daten aus der DBLP.XML-Datei, die aufgrund ihrer Zusammentragung von Hand als wahr angesehen werden können, wäre das Maß maximal. Für neu hinzugewonnene Informationen wäre es initial klein steigt mit jeder weiteren Bestätigung - sei es im Dialog oder durch Extraktion desselben Sachverhalts aus anderer Quelle.

Eine zweite Möglichkeit, das Hintergrundwissen zu erweitern, wäre die Abfrage vorhandener Publikationsarchive wie IEEE Xplore[1] (1.731.070

[1] `http://ieeexplore.ieee.org/`

Dokumente), CiteSeer[2] (767.558 Dokumente), ACM Portal[3], Google Scholar[4], Live Academic[5], CS BibTeX[6], Libra[7] oder ScienceDirect[8]. Manche von ihnen unterstützen das „Open Archive Initiative Protocol for Metadata Harvesting" [LdSNW04], welches die automatische Abfrage erleichtert. Es wäre außerdem denkbar, diese gezielt abzufragen und deren Ergebnisse mit auszuliefern. Ein alleiniges Berufen auf solche Archive wäre allerdings nicht ausreichend. In die meisten von ihnen werden neue Publikationen von Hand und nur auf Antrag aufgenommen. Daher sind sie nicht erschöpfend. Außerdem ist zu beachten, dass die Datenbanken solcher Archive zwar frei abfragbar sind, für die Publikation als PDF-Dokument jedoch zum Teil gezahlt werden muss.

Des Weiteren könnte neben der Publikationsseite selbst und den darauf verlinkten PDF-Dokumenten eine dritte Informationquelle genutzt werden. Einige Autoren stellen zu ihren Publikationen BibTeX-Dateien zur Verfügung, welche ebenfalls auf der Publikationsseite verlinkt sind. Sie sind anhand der Dateiendung .bib leicht identifizierbar. Aufgrund des klar spezifizierten Dateiformats sind die darin enthaltenden Informationen leicht zugänglich.

Schließlich wäre ein Verfahren denkbar, welches auf Basis von Lernbeispielen die Parameter des PAT-Tree-Ansatzes automatisch optimieren.

[2] `http://citeseer.ist.psu.edu/`
[3] `http://portal.acm.org/`
[4] `http://scholar.google.de/`
[5] `http://academic.live.com/`
[6] `http://liinwww.ira.uka.de/bibliography/index.html`
[7] `http://libra.msra.cn/`
[8] `http://www.sciencedirect.com/`

7 Zusammenfassung

In dieser Studienarbeit wurde der von Hsu et al. entwickelte Algorithmus zur automatischen Generierung von Extraktionsmustern für semistrukturierte Websites eingesetzt, um Publikationsangaben aus semistrukturierten Publikationsseiten zu extrahieren. Dessen zentraler Bestandteil ist eine Datenstruktur namens PAT-Tree, welche es ermöglicht, effizient sich wiederholende Muster in einem Dokument zu finden. Er wurde von uns modifiziert und um einen Klassifikator ergänzt, der von den generierten Mustern automatisch geeignete auswählt, mit ihnen Informationen extrahiert und diese in strukturierter Form ausgibt. Das gesamte Verfahren läuft vollautomatisch und unüberwacht ab. Zur Klassifikation der Publikationsattribute werden Informationen aus der DBLP, der Computer Science Bibliography der Universität Trier, eingesetzt, auf deren Grundlage ein *tfidf*-Maß für die Relevanz der jeweiligen Wörter für die einzelnen Attribute berechnet wird. Die Extraktionsresultate sind sehr unterschiedlich und zeigen ein Verbesserungspotential des Klassifikators auf.

Literaturverzeichnis

[CHL03] CHANG, CHIA-HUI, CHUN-NAN HSU und SHAO-CHEN LUI: *Automatic Information Extraction from Semi-Structured Web Pages By Pattern Discovery.* Decision Support Systems, 35(1):129–147, April 2003. Special Issue on Web Retrieval and Mining.

[CLW01] CHANG, CHIA-HUI, SHAO-CHEN LUI und YEN-CHIN WU: *Applying Pattern Mining to Web Information Extraction.* In: *PAKDD '01: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Seiten 4–16, London, UK, 2001. Springer-Verlag.

[GBYS92] GONNET, GASTON H., RICARDO A. BAEZA-YATES und TIM SNIDER: *Information Retrieval: Data Structures & Algorithms*, Kapitel New Indices for Text: Pat Trees and Pat Arrays, Seiten 66–82. Prentice Hall, 1992.

[Gus97] GUSFIELD, DAN: *Algorithms on Strings, Trees, and Sequences: computer science and computational biology.* Cambridge University Press, New York, NY, USA, 1997.

[HD98] HSU, CHUN-NAN und MING-TZUNG DUNG: *Generating finite-state transducers for semi-structured data extraction from the Web.* Inf. Syst., 23(9):521–538, 1998.

[HGM+03] HAN, HUI, C. LEE GILES, EREN MANAVOGLU, HONGYUAN ZHA, ZHENYUE ZHANG und EDWARD A. FOX: *Automatic document metadata extraction using support vector machines.* In: *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, Seiten 37–48, Washington, DC, USA, 2003. IEEE Computer Society.

[HSE+06] HOLZAPFEL, HARTWIG, THOMAS SCHAAF, HAZIM KEMAL EKENEL, CHRISTOPH SCHAA und ALEX WAIBEL: *A*

Robot Learns to Know People - First Contacts of a Robot. In: FREKSA, C., M. KOHLHASE und K. SCHILL (Herausgeber): *KI*, Band 4314, Seiten 302–316, 2006.

[KWD97] KUSHMERICK, NICHOLAS, DANIEL S. WELD und R. DOORENBOS: *Wrapper Induction for Information Extraction.* In: *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Japan, 1997. University of Washington. Chairperson-Daniel S. Weld.

[LdSNW04] LAGOZE, CARL, HERBERT VAN DE SOMPEL, MICHAEL NELSON und SIMEON WARNER: *The Open Archives Initiative Protocol for Metadata Harvesting.* `http://www.openarchives.org/OAI/openarchivesprotocol.html`, 2004.

[Lee97] LEEK, TIMOTHY ROBERT: *Information Extraction Using Hidden Markov Models.* Diplomarbeit, University of California, San Diego, 1997.

[Ley07] LEY, MICHAEL: *DBLP Bibliography.* `http://www.informatik.uni-trier.de/~ley/db/`, September 2007.

[MMK99] MUSLEA, ION, STEVEN MINTON und CRAIG A. KNOBLOCK: *Hierarchical Wrapper Induction for Semistructured Information Sources.* In: *Proceedings of the 3rd International Conference on Autonomous Agents*, Seattle, WA, 1999.

[Mor68] MORRISON, DONALD R.: *PATRICIA Practical Algorithm To Retrieve Information Coded in Alphanumeric.* Journal of ACM, 15(4):514–534, Januar 1968.

[Mus99] MUSLEA, ION: *Extraction Patterns for Information Extraction Tasks: A Survey.* In: *Proceedings of AAAI'99: Workshop on Machine Learning for Information Extraction*, 1999.

[PM04] PENG, FUCHUN und ANDREW MCCALLUM: *Accurate information extraction from research papers using conditional random fields*, 2004.

[Put08] PUTZE, FELIX: *Social User Model Acquisition through Network Analysis and Interactive Learning.* Diplomarbeit, Universität Karlsruhe, 2008.

[Rog05] ROGINA, IVICA: *Sprachliche Mensch-Maschine-Kommunikation.* Februar 2005.

[SMR99] SEYMORE, KRISTIE, ANDREW MCCALLUM und RONI ROSENFELD: *Learning Hidden Markov Model Structure for Information Extraction.* In: *AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999.

[TWWWC97] THE WORLD-WIDE WEB CONSORTIUM, W3C: *Extensible Markup Language (XML).* `http://www.w3.org/XML/`, 1997.

[TWWWC99] THE WORLD-WIDE WEB CONSORTIUM, W3C: *HTML 4.01 Specification.* `http://www.w3.org/TR/REC-html40/`, 1999.

[WDG07] WEB DESIGN GROUP, WDG: *Wilbur - HTML 3.2.* `http://www.htmlhelp.com/reference/wilbur`, 2007.

[Wik07] WIKIPEDIA: *Informationsextraktion — Wikipedia, Die freie Enzyklopädie.* `http://de.wikipedia.org/w/index.php?title=Informationsextraktion&oldid=39831204`, 2007. [Online; Stand 29. Dezember 2007].

Stichwortverzeichnis

T

V

W