DIPLOMARBEIT

Informationsextraktion aus semistrukturierten Publikationsdokumenten im PDF mit Conditional Random Fields

von

Hans Matthias Ernst

Institut für Anthropomatik
International Center for
Advanced Communication Technologies
Karlsruher Institut für Technologie (KIT)

Betreuer : Prof. Dr. Alex Waibel
Betreuer : Dr. Hartwig Holzapfel

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Karlsruhe, den 31.05.2010

Hans Matthias Ernst

Danksagung

Ich bedanke mich bei allen Freunden und Bekannten, die mir bei der Durchführung dieser Arbeit geholfen haben.

Besonderen Dank möchte ich meinem Betreuer Hartwig Holzapfel aussprechen, der mir trotz neuer Arbeitsstelle und Nachwuchs bis zum Abschluss meiner Arbeit zur Seite stand.

Weiterhin bedanke ich mich ganz herzlich bei meiner Familie für ihre unerschöpfliche Unterstützung und unermüdlichen seelischen Beistand in allen Lebenslangen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Aufgaben, die für einen Menschen trivial erscheinen, sind oft nur sehr schwierig für eine Maschine zu lösen. Wenn ein Mensch eine Zeitung liest oder ein Bild anschaut, versteht er nicht nur den Inhalt, sondern hat sich auch unbewusst ein komplexes, mentales Modell zurechtgelegt, welches ihm die Aufgabe des Verstehens erst ermöglicht.

Jeder Mensch, der gelernt hat zu lesen, muss sich nicht darauf konzentrieren, die strukturellen Eigenschaften eines Dokumentes aufzuschlüsseln, um den Titel eines Artikels zu finden - ein Blick genügt. Eine Maschine hat leider noch keine vergleichbaren Fähigkeiten. Obwohl sie in vielen Bereichen dem Menschen überlegen ist, gehört die Mustererkennung nach wie vor zu einer der am schwierigsten zu lösenden Aufgabe.

Dokumente mit zusätzlichen Daten zu versehen und Daten aus Dokumenten sinnvoll zu extrahieren ist eine Grundvoraussetzung für viele Aufgabenstellungen im Bereich der automatischen Informationsverarbeitung. Sind die Informationen in den Dokumenten eindeutig strukturiert, kann eine Maschine das Problem mithilfe einfacher Regelsysteme elegant lösen, nur ist dies oft so nicht der Fall. Gerade bei Medien, die nur für den menschlichen Gebrauch geschaffen worden sind, gestaltet sich die Informationsextraktion besonders schwierig, und im Falle von Seitenbeschreibungssprachen wie Postscript oder PDF ist keine triviale Lösung mehr zu finden.

Diese Diplomarbeit stellt ein System vor, welches unüberwacht Informationen aus im PDF vorliegenden Veröffentlichungen extrahiert und klassifiziert.

1.1 Motivation

Die Unabhängigkeit der Darstellung eines in einer Seitenbeschreibungssprache verfassten Dokumentes von der darstellenden Software und des darunter liegenden Betriebssystems und Gerätes ist bestimmt ein Faktor, der zur breiten Verwendung von Formaten wie PDF und PostScript geführt hat. Je mehr Informationen in dieser Art gespeichert werden, desto größer ist der Bedarf, diese Information zu katalogisieren oder anderweitig zu verarbeiten.

Die Idee zu dieser Arbeit entstammt der Bemühung, Dialogsysteme mit zusätzlichen Informationen zu unterstützen. In [Händel (2008)] stellt Ronny Händel bereits ein System zur Informationsextraktion aus semistrukturierten Webdaten vor, welches nach einer Internetrecherche zu spezifiziertem Vor- und Nachnamen eine Liste wissenschaftlicher Publikationen bereitstellt. Die Anzahl der dabei entdeckten Publikationen im PDF motiviert die Schaffung eines Systems, aus diesen Dateien ähnliche Informationen zu extrahieren.

1.2 Fragestellung

Gegeben sei eine im PDF vorliegende Veröffentlichung. Anhand visueller Eigenschaften sollen Informationen extrahiert und zur Verfügung gestellt werden.

Die zu extrahierenden Informationen umfassen die Autoren der Veröffentlichung inklusiver ihren Affiliationen mit Institutszugehörigkeit, Anschrift, Adresse, Telefonnummer, Internet- und Emailadressen. Des weiteren sollen der Titel der Veröffentlichung, die Zusammenfassung, angegebene Schlüsselwörter und das Veröffentlichungsdatum bestimmt werden.

Einmal trainiert soll das System in der Lage sein, diese Informationen aus spezifierten Publikationen im PDF automatisch und unüberwacht zu extrahieren, und über eine Anwendungsschnittstelle anderen Programmen zur Verfügung zu stellen.

1.3 Überblick

1.3.1 System

Zur Erfüllung der Fragestellung ist ein mehrstufiges System entwickelt worden, welches in der Lage ist, ein im Vektorformat vorhandenes Doku-

ment in eine zweidimensionale Datenstruktur zu transformieren, ein passendes Gestaltungsraster zu finden, Elemente der Makro- und Mikrotypographie in Merkmalsvektoren zu aggregieren und mithilfe eines CRF zu klassifizieren.

Das System gliedert sich in fünf aufeinander aufbauende Untersysteme:

PDFBox Liest das spezifizierte PDF-Dokument ein und stellt eine Liste sogenannter *Glyphen* (siehe Abschnitt 4.1.1 auf Seite 25) den anderen Komponenten zur Verfügung. Sie bildet den Eintrittspunkt in das System.

Parser Filtert bzw. korrigiert die Glyphen und speichert sie in eine mehrlagige *Corner Stitching Datenstruktur* (siehe Abschnitt 4.1.2 auf Seite 27) zur effizienten Weiterverarbeitung. Gleichzeitig werden Informationen wie die Seitenausmaße, gefundene Fonts und gegebenenfalls vorhandene Metainformationen extrahiert.

Segmenter Gruppiert die Glyphen zu zusammenhängenden Gestaltungsrastern anhand visueller Eigenschaften.

Tagger Erkennt makro- und mikrotypographische Merkmale und aggregiert diese in Merkmalsvektoren. Auf Segmentebene bestimmen Tagger die Lesereihenfolge und einige Merkmale der Makrotypograpfie (das Seitenformat, Zeilenbreite und -abstand, die Spaltengliederung sowie die Schriftgröße), auf Glyphebene zusätzlich die Auszeichnungsarten (u.a. fett, kursiv) und die Fontfamilie.

Labeler Klassifiziert die Segmente anhand der gesammelten Merkmalsvektoren in Klassen der vom System gesuchten Informationen mithilfe eines *Conditional Random Field* (siehe Kapitel 2.2 auf Seite 16).

Eine besondere Herausforderung ergibt sich aus den Abhängigkeiten zwischen den Untersystemen: ein Fehler im Parser propagiert sich in allen weiteren Untersystemen, fehlerhafte Segmentierungen stellen die Tagger vor große Probleme, und kein CRF kann aus fehlerhaften Beobachtungen sinnvolle Rückschlüsse ziehen.

Trotzdem haben sich einige elegante Lösungen finden lassen, die in den folgenden Kapiteln vorgestellt werden.

1.3.2 Ausarbeitung

Kapitel 2 befasst sich mit den mathematischen Grundlagen relationalen Lernens, insbesondere der Anwendung graphischer Modelle auf spezifische

Problemstellungen. Nach einer kurzen Einführung in Markov Prozesse, und bevor auf die Conditional Random Fields eingegangen wird, werden ähnliche Modelle vorgestellt, die letztendlich zu der Entwicklung der CRF geführt haben: beginnend mit Hidden Markov Modellen, über Maximum Entropy Markov Modelle, deren Vorzüge und Nachteile, bis schließlich den Markov Random Fields, den direkten Vorfahren der CRF. Abschließend wird präsentiert, wie ein CRF auf ein spezifisches Problem eingestellt werden kann.

Im Kapitel 3 wird auf verwandte Arbeiten sowohl im Bereich der Informationsextraktion mit CRF als auch im Bereich der Segmentierung von Texten eingegangen. Besonderes Augenmerk liegt hier auf den Ansatz und die Ergebnisse von [Peng und McCallum (2004)], da das Thema ihrer Arbeit der hier betrachteten Fragestellung sehr ähnelt. Es wird gezeigt, in welchen Punkten sich die Ansätze unterscheiden, und welche Beschränkungen der von den Autoren vorgestellte Ansatz mit sich bringt.

Die eigene Implementierung mit den wichtigsten Algorithmen wird im darauf folgenden Kapitel 4 vorgestellt. Ein exemplarischer Lauf einer Publikation durch das System bringt die Algorithmen miteinander in Verbindung, und stellt im Groben die Arbeitsweise des Gesamtsystems vor.

Zur Evaluation wird im Kapitel 5 zunächst auf die Entstehen des Korpus eingegangen und der Tagger vorgestellt, der im Rahmen dieser Diplomarbeit für die Erstellung der Trainingsdaten programmiert wurde. Der Korpus ist auf eine spezielle Art und Weise definiert, so dass mit den richtigen Metriken sowohl der Segmentierer als auch das Conditional Random Field von ihm profitieren. Die Präsentation dieser Metriken und die Auswertung der Ergebnisse folgen direkt im Anschluss, und eine Diskussion der Resultate und deren Implikationen schließt das Kapitel ab.

Im letzten Kapitel werden die Ergebnisse dieser Arbeit noch einmal im Ganzen zusammengefasst, und ein Blick in mögliche, zukünftige Entwicklungen und Erweiterungen geworfen.

Kapitel 2

Grundlagen

Die Aufgabe, jeder Beobachtung $x \in X$ einer Beobachtungssequenz $\vec{X}$ bestehend aus möglichen Beobachtungen X eine Kennzeichnung[1] $y \in Q$ aus einer finiten Menge Q möglicher Kennzeichen zuzuweisen, ist in Forschungsgebieten wie der Bioinformatik, Computerlinguistik und Spracherkennung ein bekanntes Problem. Grafische Modelle stellen eine natürliche Art dar, stochastische Abhängigkeiten zwischen den Beobachtungen und ihrer Klassifizierung zu modellieren.

In diesem Kapitel stelle ich das in dieser Arbeit verwendete Conditional Random Field zur Modellierung bedingter Wahrscheinlichkeitsverteilungen vor, sowie die Vorzüge seiner Verwendung im Gegensatz zu generativen Modellen. Im Folgenden werden zunächst Modelle vorgestellt, die dem CRF ähnlich und deren Kenntnis zum Verständnis des Kapitels notwendig sind.

2.1 Vorgänger des Conditional Random Fields

2.1.1 Markov-Kette 1. Ordnung und Markov Eigenschaft

Markov Prozesse beschreiben Vorgänge, die sich als Sequenz $\vec{X} = x_1, \ldots, x_T$ mit $x_t \in Q$, $t \in [1, T]$ von Zuständen aus einer endlichen Zustandsmenge $Q = \{q_1, \ldots, q_n\}$ darstellen lassen. Meist haben diese Vorgänge einen Zeitbezug, und es kann zu jedem Zeitpunkt immer nur ein Zustand aktiv.

Zusätzlich sind mögliche Transitionen $(q_i, q_j), q_i, q_j \in Q$ zwischen allen Zuständen aus der Zustandsmenge Q definiert und besitzen eine *Eintritts-*

[1] im Folgenden auch als *Label* bezeichnet

wahrscheinlichkeit, die angibt, wie wahrscheinlich der Wechsel vom Zustand q_i in den Zustand q_j ist.

Sei $p(q_i, q_j)$ die Wahrscheinlichkeit für die Transition aus dem Zustand q_i in den Zustand q_j, so lässt sich die Markov-Kette 1. Ordnung definieren über

- die endliche Zustandsmenge $Q = \{q_1, \ldots, q_n\}$
- die Transitionsmatrix $A = \begin{pmatrix} p(q_1, q_1) & \cdots & p(q_1, q_n) \\ \vdots & \ddots & \vdots \\ p(q_n, q_1) & \cdots & p(q_n, q_n) \end{pmatrix}$
- den Initialisierungsvektor $\vec{\pi} = (p(x_1 = q_1), \ldots, p(x_1 = q_n))$, der die Wahrscheinlichkeit für jeden Zustand definiert, Startzustand zu sein.

Es gilt:

$$\sum_{q \in Q} p(q) = 1 \text{ sowie } \sum_{q, \tilde{q} \in Q} p(q, \tilde{q}) = 1$$

Diese Definition erfüllt gleichzeitig die *Markov Eigenschaft*: der aktuelle Zustand in der Zustandssequenz $\vec{X}$ ist stochastisch nur von einer endlichen Anzahl Vorgängerzustände abhängig. Im Fall der Markov-Kette erster Ordnung ist dies genau ein Zustand. So sind also auch keine Abhängigkeiten auf zukünftige Zustände definiert, woraus sich direkt der Zeitbezug einer Markov-Kette erklären lässt.

2.1.2 Hidden Markov Modell

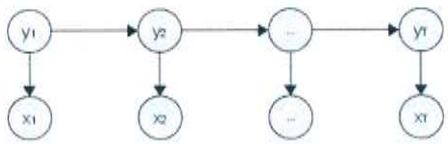

Abbildung 2.1: Das graphische Modell eines Hidden Markov Modells

Dieses Modell entspricht der Markov-Kette mit dem Unterschied, dass die dort spezifizierten Zustände jetzt nicht mehr direkt Beobachtbar sind. Es ist aber bekannt, dass beim Eintritt in einen der versteckten Zustände eine von mehreren möglichen Beobachtungen gemacht werden kann, deren Auftrittswahrscheinlichkeit direkt vom versteckten Zustand abhängig ist.

Die grundlegende Idee ist, aus einer Beobachtungssequenz $\vec{X} = x_1, \ldots, x_T$ des Beobachtungsalphabets X auf die wahrscheinlichste Sequenz der versteckten Zustände $\vec{Y} = y_1, \ldots, y_T$ zu schließen. Die versteckten Zustände stammen aus einer endlichen Zustandsmenge $Q = \{q_1, \ldots, q_n\}$, es gilt also $\forall y_t \in \vec{Y} : y_i \in Q$.

Neben den bereits von der Markov-Kette bekannten Definitionen der endlichen Zustandsmenge Q, der Transitionsmatrix A und dem Initialisierungsvektor $\vec{\pi}$ ist für die Definition eines HMM noch die *Generierungsmatrix* B nötig, die die Wahrscheinlichkeit angibt, irgendein $x_i \in X$ im Zustand q_j zu beobachten. Sei $|X| = m$ die Menge möglicher Beobachtungen, dann gilt:

$$B = \begin{pmatrix} p(x_1|q_1) & \cdots & p(x_1|q_n) \\ \vdots & \ddots & \vdots \\ p(x_m|q_1) & \cdots & p(x_m|q_n) \end{pmatrix}$$

Im Folgenden bezeichnet $M = (\vec{\pi}, A, B)$ ein vollständig definiertes HMM.

Zum testen und trainieren des HMM sind zwei Fragestellungen bei gegebener Beobachtungssequenz $\vec{X}$ effizient zu berechnen:

1. Die Wahrscheinlichkeit
$$P(\vec{X}|M)$$
die angibt, wie wahrscheinlich eine Beobachtungssequenz $\vec{X}$ gegeben dem Modell M ist. Zum einem gibt dieser Wert einen ersten Hinweis auf die Güte des unterliegenden Modells an, und zum anderem kann ein Algorithmus auch für bestimmte Trainingsalgorithmen weiterverwendet werden. Diese Aufgabe kann z.B. mit dem *Forward-Backward*-Algorithmus gelöst werden.

2. Die wahrscheinlichste Zustandssequenz
$$\vec{Y}_{\max} = \underset{\vec{Y}}{\operatorname{argmax}} \left(P(\vec{X}|\vec{Y}, M) \right)$$
die die gegebene Beobachtungssequenz $\vec{X}$ generiert. Neben dem *Forward-Backward*-Algorithmus bietet sich hier der *Viterbi*-Algorithmus an, der nicht versagt, wenn Transitionen nicht definiert oder eine Wahrscheinlichkeit von 0 haben.

Zum Trainieren der Parameter A,B und $\vec{\pi}$ werden diese zufällig initialisiert (wobei natürlich die Summe aller Werte in den Parametern genau

1 sein muss) und mithilfe des $Baum - Welch$-Algorithmus oder anderen $Expectation - Maximization$-Algorithmen optimiert. Dabei ist zu beachten, dass diese je nach Implementierung in lokale Maxima konvergieren können.

Ein weiteres Problem ist die strikte Unabhängigkeitsannahme jedes Elementes der Beobachtungssequenz $\vec{X}$: jedes $x_t \in \vec{X}$ ist nur vom emittierenden, versteckten Zustand y_t stochastisch abhängig. Damit ist es unmöglich, komplexere und sequenzübergreifende Merkmale zur Klassifizierung von x_t heranzuziehen, um z.B. den Kontext einer Beobachtung mit einzubeziehen.

Weiter ist für eine reine Klassifizierung einer Beobachtungssequenz die Verwendung multivariater Verteilungen kontraintuitiv: die Fähigkeit der selbstständigen Generierung von Beobachtungssequenzen verkompliziert unnötigerweise die Anwendung dieses Modells.

2.1.3 Maximum Entropy Markov Modell

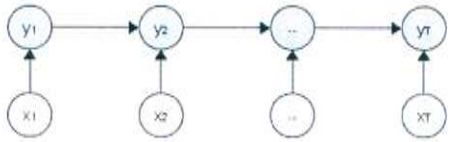

Abbildung 2.2: Das graphische Modell eines Maximum Entropy Markov Modells

Das Maximum Entropy Markov Modell - wie vorgestellt in [McCallum u. a. (2000)] - ist eine nicht-generative Weiterentwicklung des zuvor vorgestellten Hidden Markov Modells.

Im Gegensatz zum HMM sind die Elemente der Beobachtungssequenz $\vec{X}$ nicht atomar, sondern können aus mehreren sich gegenseitig überlappenden Merkmalen bestehen. Weiterhin wird nicht die multivariate Verteilung $P(\vec{X}, \vec{Y})$, sondern die bedingte Wahrscheinlichkeit

$$P(\vec{Y}|\vec{X}) = \prod_{t=1}^{T} P(y_t|y_{t-1}, x_t)$$

betrachtet, womit die oben genannten Defizite in der Klassifizierung teilweise umgangen werden können (siehe dazu auch [Sutton und Mccallum (2006)]). Der aktuelle Zustand hängt also sowohl vom vorherigen Zustand als auch von der aktuellen Beobachtung ab.

Wie zuvor ist $Q = \{q_1, \ldots, q_n\}$ die endliche Menge versteckter Zustände. Die in den Abschnitten 2.1.1 und 2.1.2 vorgestellten Transitionswahrscheinlichkeiten $P(q_i, q_j)$ (definiert durch die Transitionsmatrix A) und Generierungswahrscheinlichkeiten $P(x_t, q_i)$ (definiert über die Generierungsmatrix B) werden in MEMMs durch die Funktion

$$P(y_t = q_i | y_{t-1} = q_j, x_t)$$

ersetzt, welche die Wahrscheinlichkeit eines Zustandes $q_i \in Q$ gegeben seines Vorgängerzustandes q_j und der Beobachtung x_t definiert - anders als bei HMMs ist die Beobachtung nicht nur vom aktuellen Zustand abhängig, sondern von der Transition zwischen den Zuständen aus Q.

Im Folgenden sind $|Q|$ Funktionen P_{q_j} definiert als

$$P_{q_j}(q_i | x_t) = P(y_t = q_i | y_{t-1} = q_j, x_t)$$

Diese Erweiterung der Parametrisierbarkeit ermöglicht es, Transitionen mit mehreren, stochastisch voneinander abhängigen Merkmalen zu modellieren.

Maximum Entropy Prinzip

Bei einem MEMM wird die Annahme getroffen, dass das beste Modell für die zu untersuchenden Daten jenes ist, welches konsistent zu den in Trainingsdaten gefundene Zusammenhänge ist, aber sonst keinerlei Annahmen macht: alle nicht trainierte Kombinationen aus Vorgängerzustand, aktueller Zustand und Observation sollen aus diesem Grund untereinander gleichverteilte Wahrscheinlichkeiten wiedergeben.

Die Zusammenhänge werden als finite Anzahl binär- oder reellwertige Merkmalsfunktionen eingeführt, die zusätzliche Aussagen zu jeder Observation machen können.

Sei x_t eine Beobachtung, q_t ein möglicher Nachfolgezustand, $b : x \in X \rightarrow \{0, 1\}$ eine binärwertige Funktion und s ein Zielzustand, so definieren wir die Merkmalsfunktion durch

$$f_{b,s}(x_t, y_t) = \begin{cases} 1 & \text{falls } b(x_t) = 1 \text{ und } s = y_t \\ 0 & \text{sonst} \end{cases}$$

Nach dem Maximum Entropy Prinzip wird vorausgesetzt, dass der Wert einer Merkmalsfunktion der so genannten *Maximum Entropy* Bedingung gehorcht. Für jeden Vorgängerzustand q' und jedes Merkmal $a = (b, s)$ muss die bedingte Wahrscheinlichkeit $P_{\tilde{q}}(s|X)$ folgende Gleichung erfüllen:

$$\frac{1}{m_{q'}} \sum_{k=1}^{m_{q'}} f_{b,s}(x_{t_k}, y_{t_k}) = \frac{1}{m_{q'}} \sum_{k=1}^{m_{q'}} \sum_{q \in Y} P(q|q', x_{t_k}) f_{b,s}(x_{t_k}, q)$$

Die Folge $t_1, \ldots, t_{m'_q}$ entspricht jenen Positionen in den Sequenzen, in denen y_{t_k} den Wert q' annimmt und die Transitionsfunktion $P(q|q', x_{t_k})$ aktivieren.

So hat nun mit jede Transition folgende Wahrscheinlichkeit:

$$P(q_i|q_i, x_t) = \frac{1}{Z(X_t, q_j)} \exp\left(\sum_a \lambda_a f_a(x_t, q_i)\right)$$

Man beachte, dass die Wahrscheinlichkeiten zum Folgezustand lokal vom Vorgängerzustand über Z normalisiert wird.

Man erhält so pro Zustandsübergang ein *exponentielles Modell.*

Label Bias

Jedes MEMM leidet wie viele andere Verfahren auch an dem Problem, unter bestimmten Voraussetzungen für bestimmte Observationen immer die gleichen Zustände anzunehmen (siehe [Lafferty u. a. (2001)]).

Ist der Vorzustand y_i gegeben, parametrisiert ein MEMM die Verteilung über den nächsten Zustand als ein *Maximum-Entropy* Modell. Die Summe der Transitionswahrscheinlichkeiten eines jeden Zustands muss 1 ergeben. Aus der Formel für die bedingte Wahrscheinlichkeit einer Zustandssequenz $\vec{Y}$ bei gegebener Beobachtungssequenz

$$P(\vec{Y}|\vec{X}) = P(y_1|x_1) \cdot \prod_{i=2}^{n} P(y_i|y_{i-1}, x_i) \tag{2.1}$$

erklärt sich das so genannte *Label Bias* Problem: selbst wenn bei gegebener Beobachtungssequenz die Wahrscheinlichkeit einer Zustandssequenz $\vec{Y}_1$ lokal betrachtet höher ist, könnte das Modell eine andere Sequenz $\vec{Y}_2$ bevorzugen, wenn die Zustände in $\vec{Y}_2$ durchschnittlich weniger mögliche Transitionen in andere Zustände bietet. Beim MEMM werden die Transitionswahrscheinlichkeiten einer zur Beobachtungssequenz gehörigen Zustandssequenz jeweils

vom aktuellen Zustand (lokal) in der Sequenz auf 1 normalisiert, d.h. an der Position y_i der Zustandssequenz werden alle möglichen Folgezustände in der Sequenz unter Auswertung der Observation mit Wahrscheinlichkeiten versehen, die in der Summe 1 ergeben müssen.

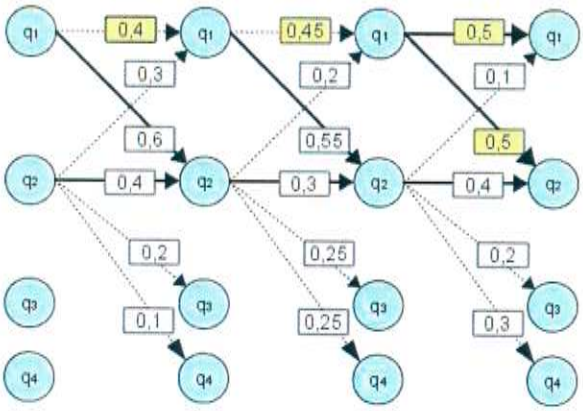

Abbildung 2.3: Beispiel für Label Bias: Ausschnitt der Transitionswahrscheinlichkeiten zwischen vier Zuständen über vier Observationen

In der Abbildung 2.3 werden exemplarisch Transitionswahrscheinlichkeiten zwischen vier Zuständen $q_1, \ldots, q_4$ über vier Observationen (im Bild nicht dargestellt) gezeigt. Die durchgezogenen Linien zeigen jeweils die wahrscheinlichste Transition von einem Zustand zu seinem Nachfolger. Werden die Transitionswahrscheinlichkeiten lokal betrachtet, würde die wahrscheinlichste Zustandssequenz von q_1 sofort auf q_2 wechseln und dort bleiben. Berechnet man jedoch den wahrscheinlichsten Pfad nach der Formel (2.1) stellt sich heraus, dass die Zustandssequenzen $\{q_1, q_1, q_1, q_1\}$ und $\{q_1, q_1, q_1, q_2\}$ wahrscheinlicher sind. Da q_2 mehr Nachfolger als q_1 definiert, und MEMMs die Transitionswahrscheinlichkeiten lokal normieren, wird nicht in den Zustand q_2 gewechselt.

2.1.4 Markov Random Field

Diese Vorform des Conditional Random Field ist definiert als ein ungerichteter, azyklischer Graph $G = (V, E)$ mit der Knotenmenge V und Kantenmenge E. Jeder Knoten im Graph entspricht genau einem Zustand aus dem Zustandsalphabet Q, und die über die Kanten gegebenen Nachbarschaften zwischen den Knoten entsprechen der stochastischen Abhängigkeit zwischen den Zuständen. Dabei gilt die Markov-Eigenschaft: seien die Elemente in $Q_q \subset Q$ diejenigen Zustände, deren korrespondierende Knoten im Graphen

mit einem zu $q \in Q$ korrespondierenden Knoten über eine Kante verbunden sind, so gilt für das Modell die bedingte Wahrscheinlichkeit

$$P(q|Q_q) = P(y|Q)$$

Im Gegensatz zu den zuvor vorgestellten Modellen können Zustände nicht nur von vorherigen Zuständen stochastisch abhängig sein, sondern auch von nachfolgenden, so wie sie im ungerichteten Graphen verknüpft sind. Aufgrund der Markov-Eigenschaft sind allerdings nur direkte Nachbarschaften voneinander abhängig.

Zur Berechnung der bedingten Wahrscheinlichkeiten werden eine Menge Ω positiv reellwertige *Potentialfunktionen* eingeführt, die auf Cliquen[2] definiert sind:

$$\Omega_{C_i} : A_{C_i} \subset V \rightarrow \mathbb{R}^+, A_{C_i} \text{ ist die i-te Clique im Graph}$$

Normalisiert mit dem Faktor

$$Z = \sum_{Y^T} \left(\prod_{\Omega} \Omega_{C_i}(y_{i_1}, \ldots, y_{i_{|C_i|}}) \right)$$

ergibt sich die Wahrscheinlichkeit für eine Labelsequenz mit:

$$P(Y) = \frac{1}{Z} \cdot \prod_{\Omega} \Omega_{C_i}(y_{i_1}, \ldots, y_{i_{|C_i|}})$$

2.2 Linear Chain Conditional Random Field

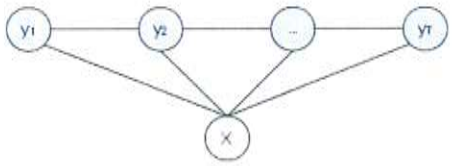

Abbildung 2.4: Das graphische Modell eines Conditional Random Fields

[2]Cliquen sind definiert als eine Teilmenge von Knoten eines Graphen, die paarweise miteinander - also vollständig - verknüpft sind

2.2.1 Begriffe und Definitionen

Das Conditional Random Field entspricht den MRF (siehe Abschnitt 2.1.4), ist aber wie die MEMM (siehe Abschnitt 2.1.3) kein generierendes, sondern konditionelles Modell, und betrachtet die bedingte Wahrscheinlichkeit einer Zustandssequenz $\vec{Y}$ bei gegebener Beobachtungssequenz $\vec{X}$.

Wie beim MEMM kann jeder Zustand aus der Sequenz $\vec{Y}$ von der gesamten Beobachtungssequenz abhängig sein, und kann wie das MRF sowohl von vorherigen wie auch nachfolgenden Zuständen abhängig sein.

Potentialfunktionen

Die Potentialfunktionen sind denen der MRF-Modelle sehr ähnlich und erscheinen in der Form

$$\Omega(y_{i-1}, y_i, \vec{X}, i) = \begin{cases} b(\vec{X}, i) & \text{falls } \Omega \text{ aktiv für } y_{i-1}, y_i \\ 0 & \text{sonst} \end{cases}$$

Wie man aus den Funktionsdefinition ersehen kann, ist es möglich, Potentialfunktionen zu entwerfen, die nur bei bestimmten Transitionen einen positiven Wert zurück liefern. Gleichzeitig ist es aber auch möglich, den Vorgängerzustand zu ignorieren, man spricht in diesem Fall von so genannten *State Features*, ansonsten von einem *Transitional Feature*. Es ist auch möglich, die Observation zu ignorieren, um die in *MP* zuerst erwähnten Transitionswahrscheinlichkeiten in der Klassifikation mit einzubeziehen. In komplexeren CRF Modell ist es auch möglich, auf mehr als einen Vorgängerzustand zuzugreifen.

Die Potentialfunktionen können mit binären oder reellwertige Merkmalsfunktionen parametrisiert werden, die sich meist dadurch auszeichnen, ohne weitere Kenntnis der Menge möglicher Observationen Charakteristiken der Observation zu beschreiben. Dies schließt jedoch nicht aus, Merkmalsfunktionen zu definieren, die auf Datenbanken basieren.

Beispiel
Sei $\Omega_k(y_{i-1}, y_i, \vec{X}, i)$ eine Potentialfunktion, die mit einer binären Merkmalsfunktion $b(\vec{X}, i)$ parametrisiert ist. Es gelte:

$$b(\vec{X}, i) = \begin{cases} 1 & \text{getFontsize}(x_i) \neq \text{getFontsize}(x_{i-1}) \\ 0 & sonst \end{cases}$$

und

$$\Omega_k(y_{i-1}, y_i, \vec{X}, i) = \begin{cases} b(\vec{X}, i) & y_{i-1} = \text{TITLE}, y_i \neq y_{i-1} \\ 0 & sonst \end{cases}$$

so gilt $\Omega_k = 1$ genau dann, wenn es einen Zustandswechsel von TITLE zu einem anderen Zustand aus Q gab, und sich die Schriftgröße geändert hat.

Falls eine Potentialfunktion für eine Transition y_{i-1}, y_i definiert und zu berechnen ist, so wird das Resultat dieser Merkmalsfunktion zurückgegeben. Das Verhalten eines CRF wird durch jeweils eine Gewichtung λ_k pro Potentialfunktionen Ω_k festgelegt.

2.2.2 Bedingte Wahrscheinlichkeit

Wie bereits erwähnt handelt es sich bei einem CRF um ein konditionelles Modell, und die bedingte Wahrscheinlichkeit ist wie beim Markov Random Field

$$P(\vec{Y}|\vec{X}) = \frac{1}{Z(\vec{X})} \prod_k \sum_{i=1}^{t} \Omega_k(y_{i-1}, y_i, \vec{X}, i)$$

Man beachte, dass hier nicht lokal Normalisiert werden muss: im Gegensatz zum MEMM leidet das CRF also nicht am Label Bias. Zum trainieren der Potentialfunktionen wird der Vektor $\vec{\lambda}$ eingeführt, der diese einzeln gewichtet. Nach [Hammersley und Clifford (1971)] lässt sich diese Formel auch noch in seine Exponentialform umschreiben, und somit ergibt sich die bedingte Wahrscheinlichkeit

$$P(\vec{Y}|\vec{X}, \vec{\lambda}) = \frac{1}{Z(\vec{X}, \vec{\lambda})} \exp\left(\prod_k \sum_{i=1}^{t} \lambda_k \Omega_k(y_{i-1}, y_i, \vec{X}, i)\right)$$

Die Normalisierungsfunktion Z ist dabei definiert durch

$$Z(\vec{X}, \vec{\lambda}) = \sum_{\vec{Y} T} \exp\left(\sum_k \sum_{i=1}^{t} \lambda_k \Omega_k(y_{i-1}, y_i, \vec{X}, i)\right)$$

Wie man sieht, wird für die Berechnung eines CRF nur ein exponentielles Modell benötigt.

Trainieren eines Conditional Random Fields

Das Trainieren eines Conditional Random Fields passt den Gewichtungsvektor λ auf die Trainingsdaten so an, so dass die Wahrscheinlichkeit für gegebene Observationssequenzen X und Zustandssequenzen Y maximal wird (Maximum-Likelihood). Es wird der Logarithmus der oben beschriebenen bedingten Wahrscheinlichkeit gebildet und maximiert. Sei H eine Menge von Trainingsbeispielen $(\vec{X}_h, \vec{Y}_h)$, dann ist die zu maximierende Formel gegeben durch:

$$L(\vec{\lambda}) = \sum_{h \in H} \log P_{\vec{\lambda}}(\vec{Y}_h | \vec{X}_h)$$

Einfachheitshalber werden die Potentialfunktionen im Folgenden bereits aufsummiert:

$$F_k(\vec{Y}|\vec{X}) = \sum_{i=1}^{T} (\Omega_k(y_{i-1}, y_i, \vec{X}, i))$$

Die Log-Likelihood-Funktion kann nun einfach aufgelöst werden nach

$$\begin{aligned} L(\vec{\lambda}) &= \sum_{h \in H} \left(\sum_k \lambda_k F_k(\vec{Y}_h, \vec{X}_h) + \log \frac{1}{Z(\vec{X}_h)} \right) \\ &= \sum_{h \in H} \left(\sum_k \lambda_k F_k(\vec{Y}_h, \vec{X}_h) - \log Z(\vec{X}_h) \right) \end{aligned}$$

Da diese Funktion konvex ist, wird nicht in die Gefahr gelaufen, in einem lokalen Minima hängen zu bleiben. Man setze nun einfach die Ableitung nach einem λ_k auf 0 und löst die Gleichung auf, um die Erwartungswerte der Funktion über die Trainingsverteilung zu berechnen. Der Erwartungswert über die Modellverteilung

$$\sum_{y \in \vec{Y} T} P(y|\vec{X}_h, \vec{\lambda}) F_k(y|\vec{X}_h)$$

muss jedoch mit iterativen oder Gradient basierenden Methoden berechnet werden, wie sie in [Lafferty u. a. (2001)] beschrieben werden.

Kapitel 3

Verwandte Arbeiten

3.1 Informationsextraktion mit Conditional Random Fields

Zur Informationsextraktion mithilfe von Conditional Random Fields gibt es bereits eine Reihe von Arbeiten, da sich das Modell sehr in den zur Anwendung in der *Named Entity Recognition* und der *Computerlinguistik* eignet. Bereits bei seiner Einführung in [Lafferty u. a. (2001)] wurde es im Bereich des *Part-of-speech Taggings* mit seinen Vorfahren - den HMMs und MEMMs - verglichen.

Ein weitaus fortschrittlicheres Verfahren nach [Zhu u. a. (2005)] nutzt layoutspezifische Merkmale zur Informationsextraktion mithilfe einer modernen Variante der CRF. Mit einem auf Internetseiten angepassten zweidimensionalen Conditional Random Field werden zwei verschiedene Arten von Internetseiten analysiert: Internetseiten mit zum Verkauf angebotenen Produkten mit vier Klassifikationen (der Produktname, ein Bild, der Preis und die Beschreibung), und Forscherseiten zur Klassifizierung des Namens, der Telefonnummer, der Emailadresse und Anschrift. Bei ihrem Vergleich mit Linear-Chain Conditional Random Fields konnten die Autoren die Effektivität ihres Ansatzes auf Trainingsdaten mit multidimensionalen Abhängigkeiten beweisen.

In einer dieser sehr ähnlichen Arbeit zeigen Andrew McCallum und Fuchun Peng in [Peng und McCallum (2004)] die Möglichkeit, mithilfe wohldefinierter Featurevektoren und Conditional Random Fields ebenfalls publikationsspezifische Informationen zu extrahieren. Dabei verglichen sie die Ergeb-

nisse der Klassifizierung durch ein CRF mit anderen Modellen wie HMMs[1] und SVMs[2].

Der Trainings- und Testkorpus für ihre Untersuchung besteht aus bereinigten Texten, und die genutzten Merkmalsfunktionen umfassen sowohl typographische als auch textinhaltliche Merkmale. Dabei sind die Texte bereits aus ihrem Vektorformat extrahiert und bereinigt, d.h. es wird nur der Text bis zur Einleitung oder - falls diese nicht auf der ersten Seite existiert - bis zum Ende der Seite betrachtet, der restliche Inhalt darf verworfen werden.

In ihrer Arbeit stellen sie drei unterschiedliche Merkmalsklassen vor, die sie zum trainieren und testen des CRF verwenden: Lokale Merkmale, Layoutmerkmale und Externe Merkmale, und betrachtet werden die extrahierten Wörter in Lesereihenfolge als Beobachtungssequenz.

Unter den oben genannten Einschränkungen und mit den in Tabelle 3.1 beschriebenen Merkmalen klassifizieren die Autoren mit einem HMM 93,1%, einer SVM 92,9%, und mit einem CRF 98,3% der Wörter richtig.

Diese Ergebnissen müssen allerdings im Rahmen der Problemeinschränkungen diskutiert werden. Es wird eine unfehlbare Bestimmung der Lesereihenfolge, der Worttrennung und der Zeilenzuweisung vorausgesetzt, doch diese Informationen sind in Vektorformaten wie PDF verloren und müssen rekonstruiert werden (siehe Kapitel 4 auf Seite 25). Die Merkmalsfunktionen PHONEORZIP, ACRO, NOTES und AFFILIATION setzen Expertenwissen voraus, und sind im Rahmen ihrer Arbeit auf den englischen Sprachraum beschränkt. Außerdem ist es notwendig, für die Merkmale BIBTEX_DATE, BIBTEX_AUTHOR, NOTES und AFFILIATION ein Lexikon zu führen und regelmäßig zu aktualisieren.

3.2 Segmentierung von PDF Dokumenten

Die Schwierigkeit, Seitenstrukturen eines PDF Dokumentes selbst mit Adobe-eigenen Anwendungen wie *PDFEdit* aus der Adobe SDK wiederherzustellen, wurde bereits in [Chao u. a. (2001)] bemerkt. Ohne segmentierende Algorithmen und geometrische Positionsanalyse ist eine Rekonstruktion nicht möglich.

Da der Informationsaustausch über Dokumente im PDF immer mehr an Beliebtheit zunimmt, gibt es eine ganze Reihe von Forschungsteams, die mit den unterschiedlichsten Verfahren Lösungen zu diesem Problem vorstellen.

[1] Hidden Markov Modell
[2] Support Vector Machine

In [Yuan u. a. (2006)] stellen die Autoren ein regelbasiertes System zur Rekonstruktion der Seitenstruktur von Dokumenten im PDF vor. Dabei wird der extrahierte Text mit wohldefinierten Markierungen versehen (*tag injection*) und mit einem mustervergleichenden Algorithmus ausgewertet.

Das AIDAS-System nach [Anjewierden (2001)] verfolgt hybride Algorithmen (Bottom-Up Rekonstruktion und Top-Down Expertensystem), bei dem logische Strukturen der Seiten wie Absätze, Listen oder Überschriften ganzer Dokumente wiederhergestellt werden können.

Ein anderer Ansatz, der visuelle Merkmale zur Informationsextraktion aus Dokumenten im PDF ausnutzt, stellen die Autoren in ihrem Artikel *Visual Information Extraction* (siehe [Aumann u. a. (2006)]) vor: mithilfe von Dokumententemplates wird in einem Top-Down-Verfahren die verloren gegangene Seitenstruktur wiederhergestellt.

Die Verfahren unterscheiden sich vor allem dadurch, wie speziell bzw. allgemein gültig sich ihr Ansatz darstellt, und welche Klassen von Dokumenten sie rekonstruieren können. Noch mehr Möglichkeiten bietet das Forschungsgebiet der optischen Texterkennung (OCR), da hier ebenfalls eine Segmentierung des Textes vor seiner Interpretation stattzufinden hat. Da sich diese mit bekannten Vertretern wie z.B. der *Rekursive XY-Cut* auch durch ihre Sprachunabhängigkeit auszeichnen und auf rein visuellen Informationen operieren, eignen sie sich hervorragend zur Weiterverwendung im Rahmen dieser Arbeit.

Name	Beschreibung
	Lokale Merkmale
INITCAP	Groß geschriebenes Wort
ALLCAPS	Kapitalisiertes Wort
CONTAINSDIGITS	Enthält Ziffern
ALLDIGITS	Enthält ausschließlich Ziffern
PHONEORZIP	Erkannt als Telefonnummer bzw. Postleitzahl
CONTAINSDOTS	Ein Punkt innerhalb des Wortes
CONTAINSDASH	Ein Minuszeichen innerhalb des Wortes
ACRO	Erkannt als Abkürzung
LONELYINITIAL	Buchstabe gefolgt von Punkt: Initialien
SINGLECHAR	Ein einzelner Buchstabe
CAPLETTER	Ein einzelner groß geschriebener Buchstabe
PUNC	Satzzeichen
URL	Erkannt als URL über reguläre Ausdrücke
EMAIL	Erkannt als Email über reguläre Ausdrücke
WORD	Als Wort erkannt
	Layoutmerkmale
LINE_START	Befindet sich am Zeilenanfang
LINE_IN	In der Mitte der Zeile
LINE_END	Am Ende der Zeile
	Externe Merkmale
BIBTEX_AUTHOR	Name in Autorenlexikon gefunden
BIBTEX_DATE	Erkannt als Datumsangabe
NOTES	Ausdrücke zur Publikationsart (*appeared, submitted*)
AFFILIATION	Ausdrücke zur Zugehörigkeit (*institution, Lab*)

Tabelle 3.1: In [Peng und McCallum (2004)] genutzte Potentialfunktionen zur Verwendung im Klassifizierer

Kapitel 4

Ansatz zur Informationsextraktion

4.1 Implementierte Verfahren und Algorithmen

4.1.1 Extraktion der Glyphen aus PDF Dokumenten

Verschiedene Open Source Entwicklungen bieten alternativ zu kommerziellen Produkten Analysemöglichkeiten der im Vektorformat gespeicherten Glyphen. Im Rahmen dieser Arbeit ist das von der *Apache Foundation*[1] betreute Projekt *PDFBox*[2] mit dessen Unterprojekt *FontBox* in der Version 0.8 verwendet worden. Diese Projekte sind freie Java-Implementierung der PDF Spezifikationen unter der *Apache Lizenz v2.0*[3]

Die Extraktion der Glyphen aus einem Dokument im PDF stellt diese Projekte vor eine große Herausforderung. Die Spezifikationen sehen nicht nur eine Vielzahl verschiedener Schriftartenfamilien (wie z.B. TrueType, AFM, BitMapFonts) vor, sondern auch unterschiedliche Möglichkeiten, Schriftartbeschreibungen in einem Dokument zu hinterlegen.

Glyphen im PDF

Ein Glyph verbindet im PDF spatiale Information - seine Position, Breite und Höhe - mit typographischen Merkmalen: der verwendete Font mit

[1] http://www.apache.org
[2] http://pdfbox.apache.org
[3] http://www.apache.org/licenses/LICENSE-2.0

seinen Spezifikationen, dessen Name, Familie, Größe und Skalierung. Zusätzlich steht je nach Implementierung der Text, den ein spezifisches Glyph repräsentiert, im Glyph zur Verfügung. Ein Glyph kann einige Buchstaben oder ganze Sätze darstellen, aber auch Steuerzeichen, die verschiedene PDF-Applikationen z.B. zur Steuerung eigener Funktionen verwenden.

Das Format gibt Entwicklern von dokumentenerzeugenden Anwendungen im PDF vielerlei Freiheiten. Leerzeichen können innerhalb eines Glyphs oder indirekt über die Abstände zwischen zwei Glyphobjekten spezifiziert werden. Das Druckbild eines Glyphen kann durch verschiedene Kombination aus horizontaler bzw. vertikaler Skalierung, der angegebenen Fontgröße sowie frei spezifizierten Transformationsmatrizen angepasst werden, und die zusätzlich vorhandenen Breiten- und Höhenangaben werden oftmals fehlerhaft oder gar nicht belegt, da das Druckbild von diesen Metainformationen unabhängig ist.

Die Problematik macht sich im verwendeten Parser dadurch bemerkbar, dass die Glyphen einiger Dokumente bzw. einige Glyphen innerhalb eines Dokumentes nur unzureichend weitergegeben werden. Ohne eine Plausibilitätskontrolle bzw. Korrektur sind die extrahierten Glyphen nicht im System verwendbar.

Normalisierung der Glyphen

Zwei Verfahren werden angewendet, um die beschriebenen Schwachstellen in einer automatischen Glyphanalyse zu bereinigen. Im ersten Verfahren wird die Tatsache ausgenutzt, dass das Druckbild den genauen Zustand eines Glyphen darstellt, wie es von einem Anzeigeprogramm oder Druckertreiber verstanden wird. Geraten die ausmultiplizierten Größenangaben eines Glyphen unter einen experimentell bestimmten Schwellwert, so wird das Glyph auf ein Bitmap gedruckt und die Höhe und Breite pixelgenau bestimmt. Die so gewonnenen Höhenangaben werden pro spezifizierten Font gesammelt und das Maximum bestimmt. Am Ende der Analyse des Dokumentes wird allen Glyphen mit der gleichen Fontspezifikation diese Höhe zugewiesen. Ist im Glyph der beschriebene Text hinterlegt, so werden die Breitenangaben über die Anzahl der Zeichen gemittelt, um eine Annäherung an die Zeichenbreite auf gleiche Weise zu ermitteln.

Das zweite Verfahren dient der Gruppierung von Glyphen in Zeilen, und behebt Probleme in der Leerzeichenberechnung, der Erkennung von Aufzählungs- und anderen Sonderzeichen sowie von hoch- bzw. tiefgestellten Text, und erleichtert die Bestimmung der Lesereihenfolge. Der verwendete Algorithmus ist eine Abwandlung des in [Breuel (2002a)] vorgestellten Verfahren namens

Robust Least Square Baseline Finding inklusiver der Erweiterung mit Nebenbedingungen nach [Breuel (2002b)] und entstammt dem Gebiet der optischen Zeichenerkennung.

4.1.2 Zweidimensionale Datenstruktur

Zur Bewältigung der Fragestellung ist eine effiziente, zweidimensionale Datenstruktur notwendig. Die Tagger, sämtliche Segmentierungsalgorithmen sowie die Trainings- und Testroutinen benötigen nicht nur ständigen Zugriff auf einzelne Glyphen, sondern sollten auch in der Lage sein, Textregionen im Ganzen auszulesen und zu annotieren. Insbesondere besteht die Notwendigkeit, Textregionen zu vereinigen oder ihre Schnitt- bzw. Differenzmenge effizient zu bestimmen. Da das Training des CRF auf einen Korpus bereits annotierter Publikationen angewiesen ist, ist die Persistenz dieser Datenstruktur eine weitere wichtige Anforderung.

In [Ousterhout (1982)] stellt John Ousterhout eine zweidimensionale Datenstruktur für Layout-Applikationen in der Schaltkreisherstellung im Bereich der *VLSI*[4] vor. Die *Corner Stitching Datenstruktur* ermöglicht es, Bereiche in einem zweidimensionalen Feld effizient zu speichern, zu markieren und auszulesen.

Corner Stitching Datenstruktur

Corner Stitching ist eine Datenstruktur, die Objekte in zwei Dimensionen sortiert. Besondere Bedeutung wird hierbei der effizienten Nachbarschaftssuche beigemessen, und unterscheidet sich insbesondere von anderen mehrdimensionalen Datenstrukturen wie z.B. multidimensionale Binärbäume dadurch, dass der leere Raum zwischen den zu speichernden Objekten explizit in der Datenstruktur spezifiziert ist, und dass das gesamte zweidimensionale Feld zu jedem Zeitpunkt wohldefiniert ist.

Alle Objekte sowie der leere Raum zwischen ihnen werden als nicht überlappende, rechteckige *Tiles* repräsentiert. Jedes Tile ist definiert über seine Attribute, die die Koordinaten der linken oberen sowie rechten unteren Ecke des Rechtecks angeben, und vier Zeiger, die sog. *corner stitches* (siehe Abb. 4.1). Diese speziellen Zeiger verbinden ein Tile derart mit seinen Nachbarn, dass Rückschlüsse auf die relative Lage der Nachbarn gezogen werden können:

[4]Very Large Scale Integration: Integrationsgrad, kennzeichnend für Schaltkreise mit über 100.000 Transistoren

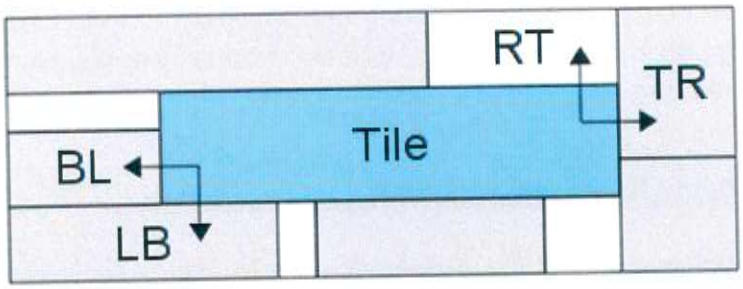

Abbildung 4.1: Beispiel eines Tiles mit den vier *Stitches*

- der *Rightest-Top*-Zeiger (RT) weist auf den Nachbarn, dessen untere Kante einen gemeinsamen Punkt mit der oberen rechten Ecke des Tiles besitzt.

- der *Top-Right*-Zeiger (TR) weist entsprechend auf den Nachbarn, dessen linke Kante einen gemeinsamen Punkt mit der oberen rechten Ecke des Tiles besitzt.

- der *Leftest-Bottom*-Zeiger (LB) entspricht punktsymmetrisch dem Rightest-Top-Zeiger. Er weist auf den Nachbarn, dessen obere Kante einen Punkt mit der unteren linken Ecke des Tiles aufweist.

- der *Bottom-Left*-Zeiger (BL) entspricht punktsymmetrisch dem Top-Right-Zeiger und weist auf den Nachbarn, dessen rechte Kante sich einen Punkt mit der unteren linken Ecke des Tiles teilt.

Mithilfe dieser Datenstruktur können so die Glyphen eines PDF Dokumentes räumlich gespeichert werden, und erfüllen die o.g. Anforderungen zur weiteren Verarbeitung.

Da sich Glyphen in einem PDF Dokument gegenseitig überlappen können, ist bei der Implementierung dieser Datenstruktur darauf geachtet worden, schnell Kollisionen der Tiles zu erkennen. Bei Kollision wird eine neue Corner Stitching Schicht erstellt, und im weiteren Verlauf werden die Glyphen auf alle erzeugten Schichten gleichverteilt. Dies stellte sich im Rahmen dieser Anwendung sogar als effizienter als die traditionelle Implementierung nach [Ousterhout (1982)] heraus, da sich die Anzahl der Tiles in den Schichten um mehrere Größenordnungen verringerte, und nur in Extremfällen mehr als drei Schichten vonnöten sind.

Operation	Liste	K-d Baum	Quadtree	Corner Stitching
Punktsuche	N	$\log N$	$T + \log N$	$\sqrt{N}$
Nachbarsuche	N	$\log N$	$T + \log N$	1
Bereichsuche	N	n	$T + n$	n
Objektsuche	nN	$n \log N$	$n(T + \log N)$	n
Objekt hinzufügen	1	$\log N$	$\log N$	$\sqrt{N} + n$

Tabelle 4.1: Laufzeitklassen einiger Datenoperationen nach [Marple u. a. (1990)].
N: Gesamtzahl der Objekte in der Datenstruktur.
n: Anzahl Objekte pro Knoten bzw. Anzahl Objekte im gesuchten Bereich
T: Anzahl Objekte pro Knoten im Quadtree

4.1.3 Rekursiver XY-Cut

Der *Rekursive XY-Cut* ist ein baumbasierter Segmentierungsalgorithmus, bei dem ein Dokument rekursiv der Länge nach horizontal und vertikal geschnitten wird, bis ein Endkriterium erreicht wird (nach [Nagy u. a. (1992)]).

In seiner naivsten Form such der Algorithmus in jedem Schritt nach der größtmöglichen horizontal bzw. vertikal ausgerichteten leeren Fläche, teilt an dieser Stelle das Dokument in zwei Hälften und ruft sich rekursiv mit den neuen Flächen parametrisiert auf, wobei er bei jedem Abstieg (also bei jedem rekursiven Aufruf) die gesuchte Ausrichtung der Weißfläche wechselt. Ein mögliches Abbruchkriterium kann beispielsweise die Größe der Fläche sein, auf der gesucht wird.

Je nach Fragestellung variieren die Implementationen des Algorithmus. Stören L-förmige Weißflächen die Suche nach einem guten Schnitt, so kann statt nach durchgehenden horizontalen bzw. vertikalen Weißflächen zu suchen nach T-förmigen Weißflächen gesucht werden.

Im Rahmen dieser Arbeit stand jedoch vielmehr die Frage im Vordergrund, auf welche Art der bestmögliche Schnitt gewählt werden kann, und wie das Abbruchkriterium zu definieren ist. In Experimenten hat sich schnell herausgestellt, dass der beste Schnitt in längst nicht allen Fällen derjenige mit der größten Weißfläche ist: vor allem die Namen der Autoren und ihre Emailadressen zeigten Teilweise so geringen Abstand, dass sie von diesem Algorithmus niemals getrennt werden konnten. Es stellt sich jedoch heraus, dass man sich nicht nur auf eine einzige Bewertungsfunktion zur Wahl des

Schnitts beschränken muss.

Mithilfe der auf Segmentbasis operierenden Merkmalsfunktionen, die eigentlich zur Analyse der Segmente im CRF verwendet werden, kann die Güte eines Schnittes berechnet werden. Analysieren Merkmalsfunktionen Bereiche, in denen mehrere, unterschiedliche Textformatierungen enthalten sind, weisen sie diesen nur eine geringe Konfidenz zu. Da semantisch zusammengehörende Elemente in einer Publikation meist ein ähnliches Schriftbild aufweisen, ist somit klar, dass das betrachtete Segment so nicht ausreichend zur Weiterverarbeitung ist.

Je mehr Formatierungen gefunden werden, desto höher ist die Wahrscheinlichkeit, an einer semantisch sinnvollen Position im Dokument segmentiert zu haben. So bleibt nur noch, den Rekursiven XY-Algorithmus derart anzupassen, dass er die Summe aktiver Merkmalsfunktionen als Bewertungsfunktion verwendet.

Der für das System entwickelte Algorithmus führt über alle möglichen Schnitte Buch, und wählt genau die Kombination an Schnitten, die über alle Segmente die meisten aktiven Merkmalsfunktionen vorzuweisen hat.

4.1.4 Pyramidensegmenter

Diese Segmentierungsart entstammt der Idee, ein Dokument aus verschiedenen Entfernungen zu betrachten: je weiter entfernt der Betrachter sich zum Text befindet, desto verschwommener sind die Details und das Auge nimmt nur noch zusammengehörende Strukturen wahr. In [Tan und Zhang (2000)] zeigen die Autoren die Übertragung dieser Idee in einen Algorithmus.

Initialisierung

Die zu konstruierende Pyramidenstruktur besteht aus einer Sequenz $P_0, \ldots, P_L$ quadratischer Binärfelder P_i mit den Ausmaßen $2^{L-i} \times 2^{L-i}$, wobei $L \in \mathbb{N}^+$ die maximale Höhe der Pyramide beschreibt und angemessen zu bestimmen ist. Jedes Binärfeld P_i wird rekursiv aus seinem Vorgänger P_{i-1} konstruiert. Sei $P_i(x, y) = 1 =$ wahr wenn das Bit an der Position $x, y \in [0, 2^{L-1} - 1]$ im Binärfeld $P - i$ gesetzt ist, dann gilt:

$$P_i(x,y) = \begin{cases} 1 & P_{i-1}(x,y) \vee P_{i-1}(x+1,y) \vee P_{i-1}(x,y+1) \vee P_{i-1}(x+1,y+1) \\ 0 & sonst \end{cases} \tag{4.1}$$

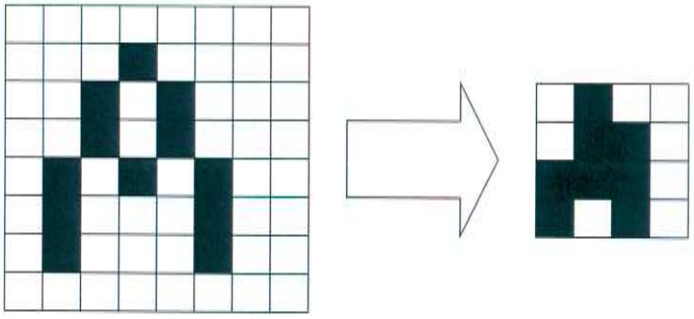

Abbildung 4.2: Konstruktion eines Binärfeldes aus seinem Vorgänger

Sei $h \in \mathbb{R}$ die Höhe des zu segmentierenden PDF Dokumentes, so bestimmen wir das Levelmaximum (die Höhe der Pyramide) mit

$$L = \lfloor \log_2 h \rfloor$$

Zur Initialisierung des *Basislevels* P_0 können nun alle Glyphen passend skaliert auf das $2^L \times 2^L$ große Binärfeld gedruckt werden (siehe dazu auch den Abschnitt 4.1.1 auf Seite 26), und alle weiteren Binärfelder $P_1, \ldots, P_L$ entsprechend der Formel 4.1 initialisiert werden. In der praktischen Anwendung sind die letzten drei Binärfelder nicht von Bedeutung und können meist ausgelassen werden.

Zwei gesetzte Bits im Bitfeld gelten als *benachbart*, falls sie sich im Feld berühren oder direkt diagonal zueinander liegen. Formal gilt also: $P_i(x, y)$ und $P_i(\tilde{x}, \tilde{y})$ mit $(x, y) \neq (\tilde{x}, \tilde{y})$ sind genau dann benachbart, wenn $P_i(x, y) = P_i(\tilde{x}, \tilde{y}) = 1$ und $|x+y-\tilde{x}-\tilde{y})| \leq 2$. Eine Menge gesetzter Bits im Bitfeld gilt genau dann als *zusammenhängende Komponente*, wenn ein Pfad *benachbarter* Bits zwischen allen möglichen Bitpaaren der Menge gefunden werden kann. Zwei benachbarte Bits gehören also immer der gleichen zusammenhängenden Komponente an.

Algorithmus Der Algorithmus sucht ausgehend von einem algorithmisch gefundenen Startlevel *zusammenhängende Komponenten*, die - in mehreren Schritten optimiert - auf das zu segmentierende Dokument zurück übertragen werden und dort die gefundenen Segmente definieren:

1. Bestimme das Startlevel anhand der *relativen Dichte* der Bitfelder. Die *Dichte* eines Bitfeldes P_i ist der Quotient aus der Anzahl gesetzter Bits und allen Bits im Feld. Die *relative* Dichte eines Bitfeldes $P_i, i > 0$

ist definiert als der Quotient aus der Dichte des Bitfeldes P_i und der Dichte des Basislevels P_0. Das Startlevel $S = i$ bezieht sich dann auf das größte Bitfeld P_i mit der relativen Dichte ≥ 3.4.

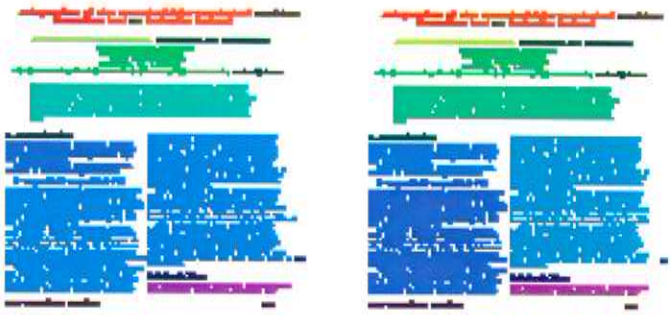

Abbildung 4.3: Links: Bitmap eines PDF im Startlevel, verbundene Komponenten mit gleicher Farbe markiert (Schritt 1). Rechts: unterbesetzte Spalten in großen Komponenten gelöscht (Schritt 2)

2. Suche im Bitfeld P_S nach unterbesetzten Spalten in *großen zusammenhängenden Komponenten*. Eine zusammenhängende Komponente wird als *große zusammenhängende Komponente* bezeichnet, wenn die Anzahl gesetzter Bits der Komponente größer ist als $2 \cdot 2^{L-S}/100$. Ist dies nicht der Fall, werden sie als *kleine zusammenhängende Komponente* definiert.

3. Betrachte pro zusammenhängender Komponente alle Bits innerhalb ihres Fangrahmens. Der Fangrahmen einer Komponente ist das kleinstmögliche Rechteck, welche alle Bits der Komponente enthält. Zeilenweise von links nach rechts: Setze alle Bits bis zum ersten gesetzten Bit der Komponente in gleicher Reihe auf 1. Dieser Schritt füllt sämtliche Einrückungen auf der linken Seite auf, und sie gehören so zur zusammenhängenden Komponente.

4. Sämtliche *kleine zusammenhängende Komponente*, die sich innerhalb des Fangrahmens einer großen zusammenhängende Komponente befinden, werden mit dieser zusammengeführt.

5. Alle übrig gebliebenen, kleinen Komponenten werden anhand der Newton-Metrik $n = \frac{m_1 \cdot m_2}{r^2}$ (siehe [Tan und Zhang (2000)]) paarweise miteinander verbunden, falls $n > 1$. Dabei ist m_i die Anzahl gesetzter Bits der kleinen Komponente i, und r der euklidische Abstand der Mittelpunkte zwischen den beiden betrachteten Komponenten.

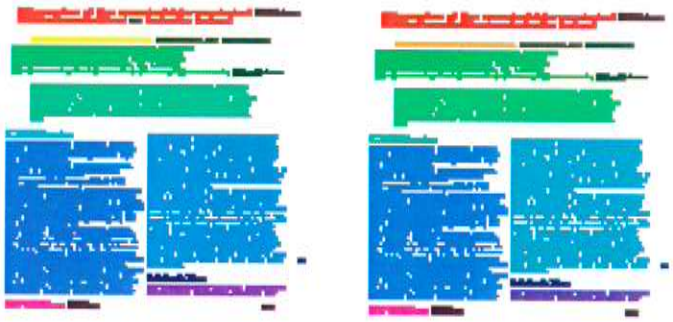

Abbildung 4.4: Links: Linkseinrückungen sind aufgefüllt (Schritt 3). Rechts: kleine Komponenten innerhalb großer Komponenten werden mit diesen zusammengeführt (Schritt 4)

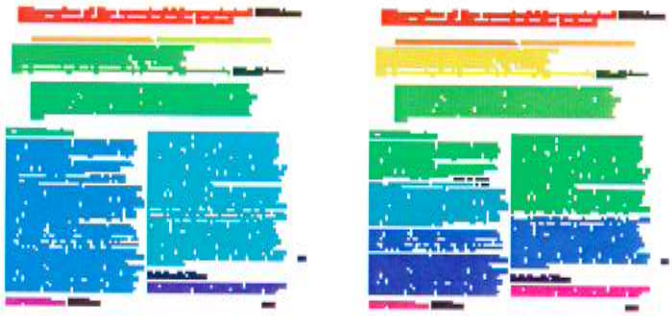

Abbildung 4.5: Links: kleine Komponente nach Newton-Metrik verbunden (Schritt 5). Rechts: Paragraphen sind anhand der rechtsseitigen Einrückung gespalten (Schritt 6).

6. Betrachte zeilenweise die rechtsseitige Einrückung der großen Komponenten in Bezug auf ihren Fangrahmen: ist diese größer als die durchschnittliche rechtsseitige Einrückung innerhalb der Komponente, spalte die Komponente am Ende der Einrückung in ihren oberen und unteren Teil. Dieser Schritt spaltet Paragraphen, die sonst als ein großes Segment erkannt werden würden.

7. Übertrage die Fangrahmen der verbundenen Komponenten auf die Koordinaten des ursprünglichen Dokumentes: sämtliche Glyphen, die sich innerhalb einer rücktransformierten Komponente befinden, gelten als Segment.

Aufgrund der hervorragend Ergebnisse, die der Algorithmus für das Conditional Random Field liefert (siehe dazu die Abschnitte 5.2.1 und 5.3.1), ist dieser Algorithmus der einzige noch aktive Segmentierer im System.

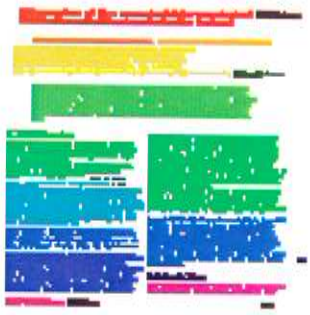

Abbildung 4.6: Bitmap eines PDF im Startlevel, verbundene Komponenten mit gleicher Farbe markiert

4.2 Arbeitsweise der Komponenten

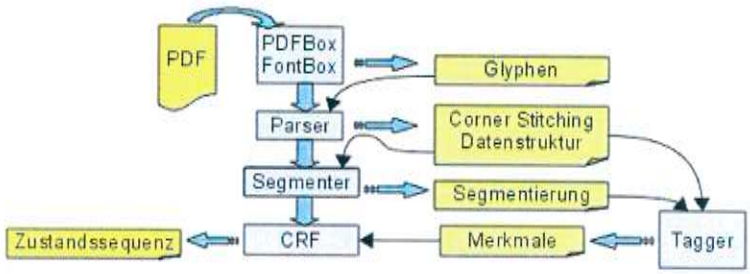

Abbildung 4.7: Das System im Überblick

Neben den Kernkomponenten sind eine Vielzahl weiterer Komponenten daran beteiligt, die Analyse zu ermöglichen. Im Folgenden wird die Analyse eines PDFs exemplarisch beschrieben, um einen Überblick über die Arbeitsweise des implementierten Systems vermitteln zu können.

Wie in der Abbildung 4.7 angedeutet, sind fünf Hauptkomponenten an der Analyse einer Publikation beteiligt. Die Toolkits PDFBox bzw. Fontbox bilden dabei eine Ausnahme, da sie im System nur als Schnittstelle zu den PDF-Dateien genutzt werden und seien hier nur der Vollständigkeit halber erwähnt.

Sind die Glyphen generiert, werden sie dem Parser zur Verfügung gestellt. Dieser führt eine Reihe von Filterfunktionen aus, bei denen fehlerhafte Glyphen entweder repariert oder verworfen werden. In den meisten Fällen handelt es sich dabei um Steuersymbole, die von manchen PDF-Erzeugern im Format versteckt werden (siehe dazu auch Abschnitt 4.1.1).

Sind die fehlerhaften Glyphen korrigiert, werden sie in die mehrlagige Cor-

ner Stitching Datenstruktur übertragen (Abschnitt 4.1.2). An diesem Punkt fängt der Tagger an, Informationen zu sammeln. Die Namen und Skalierungen der Fonts werden in Histogramme gespeichert, und es wird ein minimaler Rahmen berechnet, in den alle Glyphen hineinpassen. Dies ist notwendig, um die in Tabelle 5.3 erwähnten Merkmale *commonFont*, *smallesFont*, *largestFont*, *top*, *bottom*,*topThird*, *middle* und *bottomThird* zu berechnen.

Im Trainingsfall lädt der Tagger auch die vordefinierten Label und testet die geladenen Daten auf Integrität: sind Glyphen gar nicht oder mit mehr als einem Label versehen, wird je nach Konfiguration abgebrochen, oder es wird die graphische Oberfläche zum Labeln gestartet.

Der Segmenter wird mit dem minimalen Rahmen und den Glyphen in der 2D-Datenstruktur parametrisiert und berechnet die Segmente (siehe Abschnitte 4.1.4 und 4.1.3). Der Tagger lädt diese Segmente und berechnet aus ihnen die Merkmale *leftAlign*, *rightAlign*, *centered*, *block*, *smallestLS* und *largestLS*. Hierzu startet der Tagger verschiedene Algorithmen zur Berechnung der Zeilen und führt einen topologischen Sortieralgorithmus durch, um die Lesereihenfolge der Segmente zu bestimmen.

Die sortierten Segmente und die vom Tagger gesammelten Informationen werden nun dem CRF-Modul übergeben, welches daraus die wahrscheinlichste Labelsequenz berechnet. Diese Informationen können nun über eine Anwendungsschnittstelle erfragt werden.

Kapitel 5

Experimente und Ergebnisse

5.1 Erstellung des Publikationskorpus

Zum Trainieren der Conditional Random Fields sowie zur Auswertung verschiedener Segmentierungsalgorithmen und der Güte der Informationsextraktion wurde im Verlauf der Diplomarbeit ein Datenkorpus aus 350 Publikationen erstellt.

Die Publikationen im Datenkorpus erstrecken sich über eine breite Auswahl an Themengebieten, Sprachen und Formatierungen, um sicherzustellen, nicht einen speziellen Typus zu bevorzugen und damit die Effektivität des Systems unnötig einzuschränken. Es hat sich gezeigt, dass unterschiedliche Forschungsgebiete oft ihren eigenen Publikationsstil entwickeln, der sich deutlich in der Typographie der betrachteten Dokumente widerspiegelt: während beispielsweise im mathematisch/technischen Bereich zweispaltige Texte dominieren, deren Erscheinung der Einfluss bekannter Textsatzprogramme wie Latex deutlich anzusehen ist, wird in anderen Gebieten eher einspaltiger Text mit doppeltem Zeilenabstand bevorzugt. Ähnliche Beobachtungen können auch bei Dokumenten aus unterschiedlichen Sprachräumen angestellt werden. Allgemein kann festgestellt werden, dass je weiter sich publizierende Entitäten voneinander entfernen, desto eher evolvieren sie einen zueinander unterschiedlichen Stil.

Google Scholar[1] bietet eine leistungsstarke Schnittstelle zur Suche nach Publikationsdokumenten, und bietet ebenfalls die Möglichkeit, die Ergebnisse auf Dokumente im PDF zu beschränken. Damit fällt es leicht, eine zufällige

[1] http://scholar.google.de

Menge unterschiedlicher Publikationen zu sammeln, ohne sich unbewusst auf bestimmte Themengebiete oder Sprachen festzulegen.

Für jede im Korpus vorhandene Publikation sind Bereiche definiert, die die semantische Struktur des Textes beschreiben. Jedem Glyph auf den relevanten Seiten ist also genau ein *Tag* zugewiesen (siehe Tabelle 5.1). Um die Markierung von Bereichen mit Tags im Korpus zu erleichtern ist im Rahmen dieser Diplomarbeit ein Programm entstanden, welches es ermöglicht, diese Bereiche eindeutig für ein PDF-Dokument zu definieren und persistent zu speichern.

Abbildung 5.1: Glyphendarstellung in der Tag-Anwendung. Bereiche mit gefärbten Hintergrund sind entsprechend ihrer Farbe mit einem der links angeführten möglichen Tags markiert, die in grün gefärbten Glyphen werden gerade selektiert.

Im sogenannten Batchmodus durchsucht das Programm einen Ordner nach PDF Dateien, über die keine oder unvollständige Tag-Informationen

vorliegen, und ermöglicht dem Benutzer, mithilfe einer grafischen Benutzeroberfläche diese Bereiche mit Tags zu versehen. So kann jederzeit der Korpus erweitert werden, indem einfach PDF-Dateien in den Korpusordner kopiert werden.

Name des Tags	Semantische Bedeutung
abstract	Die Einleitung der Publikation
affil	Institut, Anschrift u.ä.
author	Die Namen der Autoren
email	Emailadressen
heading	Kapitel- oder Sektionsüberschriften
keyword	Angegebene Schlüsselwörter zur Publikation
published	Informationen zum Verlag, Konferenz u.ä.
tel	Telefon/Fax/Handy
text	Der Publikationstext
title	Überschrift der Publikation
web	Internetadressen
other	Zu keinem Tag passende Glyphen

Tabelle 5.1: Tags und ihre Bedeutung

Aus dem Korpus lässt sich nun für jede Publikation jederzeit das gewünschte Ergebnis der Informationsextraktion auslesen, da die dort gesuchten Informationen als eigenständige Bereiche definiert sind. Da die gesuchten Informationen nicht in einen auf Klartext basierten Korpus spezifiziert sind, sondern über Bereiche mit direktem Bezug zum korrespondierenden Dokument, können diese Bereiche auch zur Evaluierung der Segmentierer verwendet werden, worauf im folgenden Kapitel noch genauer eingegangen wird. Allerdings darf sich so nichts mehr an der geometrischen Interpretation der Glyphen ändern, da sonst die Bereiche die Glyphen nicht mehr akkurat annotieren (siehe dazu Abschnitt 4.1.1 auf Seite 26).

5.2 Metriken

5.2.1 Metrik zur Evaluierung der Segmentierungsalgorithmen

Wie bereits erwähnt soll der Korpus auch zur Evaluierung der Segmentierer verwendet werden. Dazu muss zunächst betrachtet werden, welche Segmentierung im Zusammenhang dieser Arbeit optimal ist.

Sei $\vec{X} = x_1, \ldots, x_T$ die Beobachtungssequenz, anhand derer das CRF die gesuchte Zustandskette $\vec{Y} = y_1, \ldots, y_T$, $y_i \in Q$ aus dem endlichen Zustandsalphabet Q berechnet. Wie in Kapitel 4.2 beschrieben, wird die Beobachtungssequenz $\vec{X}$ aus den topologisch sortierten Textsegmenten gebildet, die der Segmentierungsalgorithmus aus der aktuell betrachteten Publikation berechnet hat. Ist die Publikation im Korpus aufgenommen worden, so ist auch bekannt, welche Teile des Textes zu welchem Zustand aus Q gehören. Diese Textteile stellen im Prinzip die Informationen dar, die das System extrahieren können soll. Optimal wäre also ein Algorithmus, der den Text genau in $|Q|$ Teile segmentiert, so dass in jedem Segment nur Text vorkommt, der genau einem Zustand in Q zugeordnet ist. Dieser Optimalzustand ist praktisch nicht zu erreichen, da die Publikationen sich so nicht darstellen.

Die nächstbeste Segmentierung, die praktisch zu erreichen und messbar ist, unterteilt den Text aus der Sicht der gesuchten Information *akkurat*, d.h. jede gesuchte Information lässt sich aus einer Untermenge aller Segmente akkurat rekonstruieren. Formal kann dies so ausgedrückt werden: sei

$$S = \{S_1, \ldots, S_m\} : S_i \subset T, \bigcup_i S_i = T$$

eine Segmentierung, jedes Segment $S_i \in S$ eine Menge Glyphen aus der betrachteten Glyphmenge T, und

$$\Gamma = \{\Gamma_1, \ldots, \Gamma_{|Q|}\}, \ \forall\, i, j : \Gamma_i \subset T, \ \Gamma_i \cap \Gamma_j = \emptyset \Leftrightarrow i \neq j$$

die Menge der Textbereiche, deren Elementen anhand der Trainingsdaten immer genau ein Zustand aus Q zugeordnet ist.

Aus der Sicht der gesuchten Informationen Γ ist eine Segmentierung also genau dann *akkurat*, wenn jedes ihrer Elemente Γ_i aus einer Untermenge der gefundenen Segmente gebildet werden kann, also wenn gilt:

$$\Gamma_i \subset S \ \forall\, i \in [1, |Q|] \tag{5.1}$$

Es ist leicht zu erkennen, dass ein trivialer Segmentierungsalgorithmus, der beispielsweise jeden Buchstaben eines Textes in ein eigenes Segment abspaltet, auf jeden Fall dieses Kriterium erfüllt.

Allerdings wird in unserem Fall nach einem effektiven Segmentierungsalgorithmus gesucht, der das CRF optimal in seiner Aufgabe der Informationsextraktion unterstützt. Wird das Kriterium (5.1) verletzt, analysiert das CRF Segmente, die es gar nicht richtig klassifizieren kann, da sich mehr als

eine der zu entschlüsselnden Informationen im gleichen Segment befinden. Das Kriterium eignet sich also gut dafür, bekannte Segmentierungsalgorithmen auf ihre Effektivität in diesem speziellen Anwendungsfall zu prüfen und miteinander zu vergleichen

Sei wieder eine Segmentierung $S = \{S_1, \ldots, S_m\}$ und die gesuchten Informationen Γ gegeben. Zunächst wird für jedes Segment S_i bestimmt, mit welchem $\Gamma_j \in \Gamma$ es sich die größte Schnittmenge teilt. Dieses Γ_j bezeichnen wir im folgenden als die *Hauptkomponente* des Segments S_i. Sei $S_{\Gamma_j} \subset S$ definiert als die Vereinigungsmenge aller Segmente, deren Hauptkomponente Γ_j ist, und $S^{\complement}_{\Gamma_j}$ seine Komplementärmenge. Dann können wir für Γ_j

$$\begin{aligned} \text{TruePositive}(\Gamma_j) &= ||S_{\Gamma_j} \cap \Gamma_j|| \\ \text{FalsePositive}(\Gamma_j) &= ||S_{\Gamma_j} \setminus \Gamma_j|| \\ \text{TrueNegative}(\Gamma_j) &= ||S^{\complement}_{\Gamma_j} \setminus \Gamma_j|| \\ \text{FalseNegative}(\Gamma_j) &= ||\Gamma_j \setminus S_{\Gamma_j}|| \end{aligned} \tag{5.2}$$

mit einer beliebigen Norm $||\cdot||$ berechnen (die Norm, die allen Ergebnissen in Kapitel 5.3 zugrunde liegt, ist definiert als die Summe der Buchstaben aller Glyphen im Textsegment).

In der folgenden Evaluation werden die Maße *Precision*,*Recall* und *Accuracy* verwendet, die wie folgt definiert sind:

$$\begin{aligned} \textit{Precision} &= \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}} \\ \textit{Recall} &= \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}} \\ \textit{Accuracy} &= \frac{\text{TruePos.} + \text{TrueNeg.}}{\text{TruePos.} + \text{TrueNeg.} + \text{FalsePos.} + \text{FalseNeg.}} \end{aligned} \tag{5.3}$$

5.2.2 Metriken zur Evaluierung des CRF

Die Informationen, die das CRF zu extrahieren hat, stehen bereits markiert im Korpus zur Verfügung, und können so direkt zur Evaluierung herangezogen werden.

Anders als im Falle des Segmenters muss keine Hauptkomponente mehr für Segmente berechnet werden, da diese einen Zustand aus Q schon zugewiesen bekommen haben.

Sei Γ analog zu Abschnitt 5.2.1 definiert, und $\tilde{S}_{\Gamma_j} \subset S$ die Vereinigungsmenge aller Segmente, denen das CRF den Zustand Γ_j zugewiesen hat. Dann können wir genau wie in (5.2) für alle Γ_j mit einer beliebigen Norm $||\cdot||$ folgende Kennwerte berechnen:

$$\begin{aligned} \text{TruePositive}(\Gamma_j) &= ||\tilde{S}_{\Gamma_j} \cap \Gamma_j|| \\ \text{FalsePositive}(\Gamma_j) &= ||\tilde{S}_{\Gamma_j} \setminus \Gamma_j|| \\ \text{TrueNegative}(\Gamma_j) &= ||\tilde{S}^{\complement}_{\Gamma_j} \setminus \Gamma_j|| \\ \text{FalseNegative}(\Gamma_j) &= ||\Gamma_j \setminus \tilde{S}_{\Gamma_j}|| \end{aligned} \tag{5.4}$$

Die zweite Metrik, die zur Evaluation verwendet wird, ist die *Konfusionsmatrix*. In dieser quadratische Matrix werden die vom CRF ermittelten Label (also die gesuchten Informationen) mit den tatsächlichen Labeln kombiniert und ihre Auftrittshäufigkeit gezählt. Dabei korrespondieren die Spalten mit den tatsächlichen, gesuchten Zuständen aus Q, und die Zeilen mit den ermittelten Ergebnissen. Die richtigen Klassifikationen sind also auf der Hauptdiagonalen zu finden.

5.3 Evaluation

5.3.1 Evaluation der Segmenter

Im Verlauf der Ausarbeitung haben sich zwei verschiedene Ansätze zur Segmentierung der Publikationsdokumente durch ihre Effektivität im Hinblick auf die weitere Auswertung durch den Klassifizierer hervorgetan:

- XY-Cutline Segmenter
- Pyramidensegmenter

Wie bereits in Kapitel 5.1 beschrieben kann der gleiche Trainings- und Testkorpus verwendet werden, der zur Evaluation des Conditional Random Fields erstellt wurde. Da die Segmentieralgorithmen kein Training benötigen, sind alle 350 Publikationen segmentiert und anhand der im Abschnitt 5.2.1 vorgestellten Metrik miteinander verglichen worden:

Da OTHER meist nur einige wenige Glyphen beinhaltet, ist der Rekursive XY-Cut nicht in der Lage, diese vernünftig auszuwerten und wurde daher weggelassen. Es ist deutlich zu erkennen, dass das Pyramidenverfahren besser

	Rekursiver XY-Cut		Pyramidenverfahren	
	Precision	Recall	Precision	Recall
ABSTRACT	0,71	0,60	0,98	0,96
AFFIL	0,88	0,87	0,97	1
KEYWORD	0,42	0,32	0,84	0,71
OTHER	-	-	0,99	0,84
PUBLISHED	0,76	0,41	0,99	0,87
TEXT	0,95	0,98	0,99	1
TITLE	0,92	0,91	0,99	0,97

Tabelle 5.2: Evaluierung und Vergleich der Segmenter

als der Rekursive XY-Cut in der Lage ist, dem CRF akkuratere Segmente zu übermitteln.

Andere getestete Verfahren wie beispielsweise die Growing Bounding Box haben sich früh aus nicht robust oder leistungsstark genug herausgestellt: entweder mussten sie derart sensitiv parametrisiert werden, dass sich die Anzahl der Segmente als zu hoch zur Weiterverwendung im CRF herausstellten, oder verlangten Expertenwissen und waren so mit der Idee einer auf visuellen Merkmalen basierten Informationsextraktion nicht mehr vereinbar.

5.3.2 Evaluation des CRF

Experiment

In mehreren Läufen sind immer jeweils 300 Publikationen zum Trainieren des CRF und 50 zum Testen zufällig ausgewählt und ihre Ergebnisse gemittelt worden. Die verwendeten Merkmalsfunktionen für die Potentialfunktionen des CRF sind in Tabelle 5.3 zu finden.

Die vom CRF zu erzeugende Zustandssequenz - im folgenden auch als Labelsequenz bezeichnet - setzt sich aus folgender Zustandsmenge zusammen:

- ABSTRACT Die Einleitung der Publikation. Entspricht dem Tag *abstract*.
- AFFIL Die Namen, Anschriften und Emailadressen der Autoren. Entsprechen den Tags *affil*, *author*, *email* und *tel*.
- PUBLISHED Das Publikationsdatum, der Herausgeber oder die Konferenzbezeichnung und vergleichbare Informationen. Entsprechen den Tags *published* und *web*.

Name	Beschreibung
Layoutmerkmale - Zustand	
leftAlign	Text ist links ausgerichtet
rightAlign	Text ist rechts ausgerichtet
centered	Text zentriert
block	Text im Blocksatz
commonFont	Meistgenutzer Font im Dokument
largestFont	Maximale Skalierung der Schrift im Dokument
smallestFont	Minimale Skalierung der Schrift im Dokument
smallComponent	Segment füllt weniger als 10% der Seitenfläche
largeComponent	Segment füllt mehr als 10% der Seitenfläche
smallestLS	Geringster Zeilenabstand aller Segmente
largestLS	Größter Zeilenabstand aller Segmente
top	Segment befindet sich am oberen Rand
bottom	Segment befindet sich am unteren Rand
topThird	Segment im oberen Drittel der Seite
middle	Segment von anderen Segmenten umgeben
bottomThird	Segment im unteren Drittel der Seite
Layoutmerkmale - Transition	
fontSwitch	Fontwechsel zwischen Segmenten
scaleSwitch	Skalierungswechsel zwischen Segmenten
columnSwitch	Spaltenwechsel zwischen Segmenten

Tabelle 5.3: Genutzte Potentialfunktionen zur Verwendung im Klassifizierer

- TITLE Der Titel der Publikation. Entspricht dem Tag *title*.
- TEXT Der Inhalt der Publikation. Entspricht dem Tag *heading* und *text*.
- KEYWORD Die Liste der Themengebiete, die behandelt werden. Entspricht dem Tag *keyword*.
- OTHER Symbole, Randnotizen der Verleger bzw. juristische Notizen zum Copyright und andere sehr selten vorkommende Informationen. Entspricht dem Tag *other*.

In der Tabelle 5.4 sind die nach der in Abschnitt 5.2.2 definierten Metrik berechneten Ergebnisse zusammengefasst.

	Precision	Recall	Accuracy
ABSTRACT	0,64	0,64	0,82
AFFIL	0,70	0,87	0,98
KEYWORD	0,99	0,19	0,97
OTHER	0,41	0,10	0,92
PUBLISHED	0,26	0,19	0,97
TEXT	0,86	0,88	0,84
TITLE	0,71	0,70	0,99

Tabelle 5.4: Evaluierung des Conditional Random Fields

Ergebnis

Die Ergebnisse sind abhängig von der Güte der Segmentierer. Obwohl sich der Pyramidensegmentierer als sehr robust herausgestellt hat, ist nicht zu erwarten, dass er gänzlich fehlerfrei laufen kann. Ist die Segmentierung fehlerhaft, überdecken sich die Segmente nicht genau mit den Segmenten im Korpus, und das CRF ist gar nicht in der Lage, bessere Ergebnisse zu erzielen.

5.3.3 Fehleranalyse

Die mehrfach angesprochene Abhängigkeit des CRF vom Pyramidensegmentierer ist in der Evaluierung gut zu erkennen. Dort, wo der Segmentierer gute Ergebnisse liefert (TEXT, TITLE, AFFIL und ABSTRACT), kann auch das Conditional Random Field gute Arbeit leisten.

Betrachten wir nun die vergleichsweise geringeren *Recall*-Werte von KEYWORD, PUBLISHED und OTHER in der Segmenterbewertung. Keywords stehen in Publikationen häufig direkt unterhalb der Einleitung und direkt oberhalb des Textes, und es hat sich gezeigt, dass die Keywords dann zusammen mit dem Abstract, Text oder beiden in ein Segment gruppiert werden. Das bestätigt sich ebenfalls in der Konfusionsmatrix, die zeigt, dass nur selten richtig klassifiziert wurde: von 122 Segmenten sind 114 Segmente fälschlicherweise als ABSTRACT bzw. TEXT klassifiziert.

Es liegt in der Natur der zur Segmenterbewertung verwandten Metrik (siehe Abschnitt 5.2.1), größere Textfelder bei der Zuweisung der Hauptkomponente den kleineren vorzuziehen, also wird das CRF in den meisten Fällen mit dem Label ABSTRACT oder TEXT trainiert. Die geringe Bereitschaft des CRF, in den Zustand KEYWORDS zu springen (siehe Zeile KEYW in Konfusionsmatrix, Tabelle 5.5) verstärkt diese Vermutung. Eine manuelle

Analyse der Ergebnisse des Segmentieralgorithmus hat die Vermutung bestätigt. Der sehr hohe *Precision*-Wert von Keywords zeigt allerdings, dass das CRF sehr wohl in der Lage ist, richtig zu klassifizieren, falls akkurat segmentiert wurde.

Die mit PUBLISHED markierten Informationen stehen meist in gleicher Schriftart wie der Text der Publikation, und werden häufig einfach nur mit einem horizontalen Strich von diesem getrennt. Leider war es mit PDFBox nicht möglich, diesen Strich auszulesen, sonst könnte eine Merkmalsfunktion eingeführt werden, die bessere Ergebnisse liefern könnte.

OTHER bezeichnet im Gegensatz zu PUBLISHED und KEYWORD keine spezifischen Informationen, sondern dient als Sammelbezeichnung für Segmente, die anders nicht zugeordnet werden können. Es müsste eine Vielzahl neuer Label eingeführt werden, um diese Segmente vernünftig auswerten zu können, doch hat sich experimentell gezeigt, dass der Korpus hierfür zum trainieren noch zu klein ist: die Komplexität des Conditional Random Fields steigt pro neuem Segment exponentiell an, worunter die Klassifizierung leidet, und die zur Zeit verwendeten Label stellen mit OTHER einen Kompromiss zwischen Genauigkeit und Berechenbarkeit dar.

	ABSTR	AFF	PUBL	TITLE	TEXT	OTHER	KEYW
ABSTR	903	106	36	41	440	196	83
AFFIL	72	716	33	324	58	20	0
PUBL	0	0	12	0	9	4	0
TITLE	11	119	23	296	14	10	0
TEXT	398	97	78	22	3360	544	31
OTHER	0	0	0	0	0	3	0
KEYW	1	0	0	0	0	0	8

Tabelle 5.5: Konfusionsmatrix Conditional Random Field

5.4 Diskussion

Es hat sich die Vermutung bestätigt, dass selbst kleine Fehler im Segmentierer große Probleme in der Klassifizierung ergeben. Dies betrifft vor allem jene Label, die oft nur aus einigen wenigen Glyphen bestehen, und die sich visuell wenig vom restlichen Inhalt einer Publikationsseite abheben, wie es bei KEYWORDS der Fall ist.

Eine Lösung des Problems wäre, die Einführung von nicht-visuellen Merkmalen zuzulassen. Merkmale, die beispielsweise Wörterbücher benötigen, könnten auf bestimmte Schlüsselwörter wie *Keyword* oder *Abstract* reagieren und so die Leistungsfähigkeit der Informationsextraktion steigern.

Allerdings würde damit erstens die Sprachunabhängigkeit verloren gehen, und zweitens müssten die Glyphen direkt und nicht nur in Gruppen als Segment betrachtet werden. Glyphen können aus Buchstaben, Teilwörtern oder ganzen Sätzen bestehen (siehe Abschnitt 4.1.1), also müsste der Inhalt der Glyphen einer Zeile zunächst fusioniert und daraufhin in Wörter unterteilt werden. Um dies überhaupt zu ermöglichen, müssen die Glyphen zunächst in Zeilen segmentiert und die Leerzeichen berechnet werden.

Leider sind die Positions- und Größenangaben der Glyphen nicht immer korrekt, und wenn die Wortextraktion fehlschlägt, ist auch ein auf Wörterbücher basierter Ansatz nicht effektiv. Solange also der verwendete Parser diese Probleme nicht behebt, sollte von diesem Ansatz abgesehen werden.

Eine andere Lösung wäre die Erweiterung des Systems um Expertenmodule, die sich auf die Analyse spezifisch gelabelter Segmente konzentrieren Dies ist der zur Zeit wohl praktikabelste Ansatz, mit dem das System verbessert werden kann.

Kapitel 6

Zusammenfassung

6.1 Ergebnisse der Arbeit

In dieser Arbeit wurden die Grundsätze der Informationsextraktion anhand visueller Eigenschaften erläutert. Es ist ein System vorgestellt worden, welches ein Dokument im PDF in eine effiziente, zweidimensionale Struktur umwandelt, segmentiert, und welches mithilfe eines Conditional Random Fields semantische Informationen zurückgewinnen kann, die bei der Dokumenterzeugung verloren gegangen sind.

Von besonderer Bedeutung war es, den Einfluss nötiger Expertensysteme so weit es geht zu reduzieren, um eine thematisch und sprachlich breit gefächerte Auswahl an Publikationen verarbeiten zu können.

Es wurde gezeigt, dass ein aufeinander aufbauendes, hierarchisches System von der Robustheit und Effektivität jeder einzelnen Komponente abhängig ist, und es wurden Lösungen präsentiert, die genau diese Voraussetzungen erfüllen.

6.2 Ausblick

Das in dieser Arbeit vorgestellte System bietet in jedem Teilbereich Raum für Verbesserungen und Weiterentwicklungen.

Die Extraktion der Glyphen aus einer im PDF erstellen Datei war bis Abschluss der Diplomarbeit immer noch ein großes Problem. Obwohl Adobe das Portable Document Format unter eine freie Lizenz gestellt hat, ist

die Extraktion des Inhalts immer noch fehlerbehaftet. Solange das Extraktionsmodul nicht robust und zuverlässig auf alle Dokumente anzuwenden ist, leiden alle davon abhängigen Module unter der Fehlerpropagation.

Viele Gruppen in der Open Source Gemeinde haben sich dem Problem schon angenommen, doch ist bis heute noch kein konkurrenzfähige Lösung präsentiert worden. Jede Verbesserung am Parser würde die Zuverlässigkeit des Systems immens erhöhen.

In dieser Arbeit wurden zur Informationsextraktion ausschließlich Merkmale verwendet, die in der Publikation zu finden waren, und lässt viele Informationsquellen unberührt. Wie sich gezeigt hat, ist die Extraktion des Titels einer Publikation einer der leichtesten Aufgaben für das System. Man könnte die Titelextraktion vom restlichen System entkoppeln und dazu verwenden, im Internet bekannte Datenbanken wissenschaftlicher Publikationen wie *CiteSeer*[1] oder *SpringerLink*[2] abzufragen. Damit könnten Merkmalsfunktionen eingeführt werden, die den Suchraum bereits stark einschränken.

Völlig unangetastet ist auch die Möglichkeit, das gleiche System auf die Analyse von Literaturverzeichnissen anzuwenden, wie sie in jeder Publikation zu finden sind.

Vor allem aber sollte an einer Fusion dieser Ansätze gearbeitet werden. Es könnte eine Ontologie geschaffen werden, die unsicheres Wissen aus verschiedenen Quellen akzeptiert und miteinander in Relation bringen kann. Das Ziel sollte sein, die verschiedenen Ansätze, die nach dem gleichen Wissen streben, in Konkurrenz laufen zu lassen, und ihre Ergebnisse zu korrelieren. Jedem System könnte ein Konfidenzgrad erteilt werden, und finden verschiedene Ansätze die gleiche Information, steigt die Konfidenz der Einträge an entsprechender Stelle. Dies würde aus einer Menge unsicherer, probabilistischer Verfahren ein mächtiges Werkzeug zur Informationsgewinnung schaffen.

[1] http://citeseer.ist.psu.edu
[2] http://www.springerlink.de

Literaturverzeichnis

[Anjewierden 2001] ANJEWIERDEN, Anjo: AIDAS: Incremental Logical Structure Discovery in PDF Documents. In: *In 6th International Conference on Document Analysis and Recognition (ICDAR*, 2001, S. 374–378

[Aumann u. a. 2006] AUMANN, Yonatan ; FELDMAN, Ronen ; LIBERZON, Yair ; ROSENFELD, Benjamin ; SCHLER, Jonathan: Visual information extraction. In: *Knowl. Inf. Syst.* 10 (2006), Nr. 1, S. 1–15. – ISSN 0219-1377

[Breuel 2002a] BREUEL, Thomas M.: Robust Least Square Baseline Finding using a Branch and Bound Algorithm. In: *in Document Recognition and Retrieval VIII, SPIE*, 2002, S. in Druck

[Breuel 2002b] BREUEL, Thomas M.: Two Geometric Algorithms for Layout Analysis. In: *In Workshop on Document Analysis Systems*, Springer-Verlag, 2002, S. 188–199

[Chao u. a. 2001] CHAO, Hui ; BERETTA, Giordano ; SANG, Henry: PDF Document Layout Study with Page Elements and Bounding Boxes. In: *Workshop on Document Layout Interpretation and its Applications* (2001)

[Hammersley und Clifford 1971] HAMMERSLEY, John ; CLIFFORD, Peter: *Markov fields on finite graphs and lattices.* 1971. – Unpublished

[Händel 2008] HÄNDEL, Ronny: *PAT-Trees zur Automatischen Informationsextraktion aus semi-strukturierten Webdaten*, Universität Karlsruhe, Institut für Theoretische Informatik (ITI), Diplomarbeit, 2008

[Lafferty u. a. 2001] LAFFERTY, John ; MCCALLUM, Andrew ; PEREIRA, Fernando: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proc. 18th International Conf. on Machine Learning* (2001), S. 282–289

[Marple u. a. 1990] MARPLE, David ; SMULDERS, Michiel ; HEGEN, Henk: Tailor: a layout system based on trapezoidal corner stitching. In: *IEEE Trans. on CAD of Integrated Circuits and Systems* 9 (1990), Nr. 1, S. 66–90

[McCallum u. a. 2000] MCCALLUM, Andrew ; FREITAG, Dayne ; PEREIRA, Fernando: Maximum Entropy Markov Models for Information Extraction and Segmentation. In: *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2000, S. 591–598. – ISBN 1-55860-707-2

[Nagy u. a. 1992] NAGY, George ; SETH, Sharad ; VISWANATHAN, Mahesh: A Prototype Document Image Analysis System for Technical Journals. In: *Computer* 25 (1992), Nr. 7, S. 10–22. – ISSN 0018-9162

[Ousterhout 1982] OUSTERHOUT, John K.: Corner Stitching: a Data Structuring Technique for VLSI Layout Tools / EECS Department, University of California, Berkeley. University of California at Berkeley, Dec 1982 (UCB/CSD-83-114). – Forschungsbericht. – URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/1982/6352.html`

[Peng und McCallum 2004] PENG, Fuchun ; MCCALLUM, Andrew: Accurate information extraction from research papers using conditional random fields. In: *HLT-NAACL04*, 2004, S. 329–336

[Sutton und Mccallum 2006] SUTTON, Charles ; MCCALLUM, Andrew: *Introduction to Conditional Random Fields for Relational Learning*. In: GETOOR, Lise (Hrsg.) ; TASKAR, Ben (Hrsg.): *Introduction to Statistical Relational Learning*, MIT Press, 2006

[Tan und Zhang 2000] TAN, Chew L. ; ZHANG, Zheng.: Text block segmentation using pyramid structure. In: P. B. KANTOR, D. P. LOPRESTI, & J. ZHOU (Hrsg.): *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* Bd. 4307, Dezember 2000, S. 297–306

[Yuan u. a. 2006] YUAN, Fang ; LIU, BO ; YU, Ge: *A Study on Information Extraction from PDF Files*. Bd. 3930/2006. S. 258–267, Springer-Verlag Berlin / Heidelberg, 2006, ISSN 1611-3349

[Zhu u. a. 2005] ZHU, Jun ; NIE, Zaiqing ; WEN, Ji-Rong ; ZHANG, Bo ; MA, Wei-Ying: 2D Conditional Random Fields for Web information extraction. In: *ICML '05: Proceedings of the 22nd international conference on Machine learning*. New York, NY, USA : ACM, 2005, S. 1044–1051. – ISBN 1-59593-180-5

Tabellenverzeichnis

Abbildungsverzeichnis