Universität Kaiserslautern
Fachbereich Informatik
AG Datenverwaltungssysteme
Prof. Dr. Theo Härder

Einsatz von Text-Mining-Algorithmen bei der Realisierung eines semi-automatischen Erschließungswerkzeuges im Projekt MetaAkad

Projektarbeit

von

Mohamed Amine CHATTI

Betreuer:

Dipl.-Inform. Marcus Flehmig

Dezember 2002

Hiermit erkläre ich an Eides statt, daß ich die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur angefertigt habe.

Kaiserslautern, den 08.12.2002

_____

Mohamed Amine Chatti

Inhaltsverzeichnis

Kapitel 1 Einführung

In diesem Kapitel wird zuerst das Ziel dieser Projektarbeit vorgestellt und dann werden die Schwierigkeiten aufgelistet, mit denen man konfrontiert werden kann, wenn man die Problematik, die das im Rahmen dieser Projektarbeit entwickelte Werkzeug betrachten soll, mit anderen bekannten Methoden lösen will.

1.1 Ziel

Das META-AKAD (Metadatenzugang für akademisches Lehr- und Lernmaterial)-Projekt hat das Ziel, dem Nutzer einen einheitlichen Zugang zu einem umfassenden Angebot von im Internet verfügbaren Lehr- und Lernmaterial zu verschaffen. Zur Realisierung dieses Dienstes soll Online-Lehr- bzw. Lernmaterial auf kooperativer Basis gesammelt, durch standardisierte und materialspezifische Metadaten erschlossen, nach inhaltlichen und didaktischen Kritierien bewertet und in einer einheitlichen Nutzeroberfläche zugänglich gemacht werden[1]. Diese Projektarbeit soll die Betrachtung verschiedener Techniken und Konzepte zur Analyse von semi-strukturierten Dokumenten (hauptsächlich HTML) sein, um bei der Metadaten-Erschließung im Rahmen des Projektes META-AKAD ein semi-automatisches Erschließungswerkzeug erstellen zu können. Dieses Werkzeug soll sich dadurch auszeichnen, dass es nicht nur die einzelnen zur Beschreibung notwendigen Attribute eines Dokuments auszufüllen ermöglicht, sondern auch dadurch, dass es sinnvolle Vorschläge macht, d.h. eine Liste mit konkreten Vorschlägen zu den verschiedenen Attributen dem Benutzer zur Verfügung stellt.
Zu sinnvolle Vorschläge soll dabei durch die Verwendung von Text-Mining-Techniken gelangt werden. Der Einsatz solcher Methoden soll untersucht werden, indem beispielhaft für Namen (Autor, E-Mail, ...) konkrete Algorithmen ausgewählt und in ein möglichst modulares System für die Erschließung integriert werden.
Bei der Informationsextraktion und dem Text-Mining geht es um das Aufspüren und Strukturieren relevanter Informationseinheiten aus einer Menge von unstrukturierten oder semi-strukturierten Texten[2]. Ein wichtiger Teilbereich hierbei ist u.a. die sogenannte „Named-Entity-Erkennung“. In den letzten Jahren wird in diesem Teilbereich verstärkt, maschinelle Lernverfahren entwickelt. Viele Text-Mining-Algorithmen wurden entwickelt, um Namen in unstrukturierten Dokumenten zu finden und zu klassifiziern. Unter „Named-Entities“ [STE 2001] sind auch Personennamen zu finden, die u.a. den Autor in einem HTML-Dokument darstellen können.

1. https://rzblx1.uni-regensburg.de/metaakad
2. http://www.dfki.de/~neumann/ml-seminar.html

Das Ziel dieser Projektarbeit ist ein Werkzeug zu entwickeln, dass Vorschläge für das Attribut „Autor“ geben kann. Es soll dabei möglich sein Personennamen zu finden, die von einem Honorar eingeführt werden wie z.B. „Prof. Dr. Theo Härder“, d.h. man beschränkt sich in diesem Fall nicht nur auf Theo Härder als Personennamenvorschlag.

1.2 Probleme

Personennamen in einem Dokument zu erkennen ist eine schwierige Aufgabe. Normalerweise kann man, wenn man die interne Struktur des Satzes oder den Kontext genau betrachtet, entscheiden, ob ein Satzteil ein Personenenname ist. Allerdings, sowohl syntaktisch als auch semantisch, kann man Personennamen von anderen Namen oder Substantiven nicht unterscheiden, wenn man keine zusätzlichen Informationen benutzt. Deshalb setzt man häufig eine Datenbank ein, um diese Aufgabe zu lösen. Diese Idee ist zwar möglich aber das hat den Nachteil, dass der Aufwand ein Datenbanksystem zu erstellen und zu verwalten u.U. sehr hoch ist.
Wenn man also ohne Hilfe eines Datenbanksystems arbeiten will, dann muss man unbedingt Wörterbücher benutzen und sich mehrere Regeln und Heuristiken überlegen, die es erlauben richtige Personennamen zu finden.
Hierbei stößt man auf zwei Probleme: Zum einen dass, die Wörterbücher nie vollständig sein können, da die Anzahl der Personennamen nicht beschränkt ist und diese Namen in verschiedenen Formen vorkommen. Zum anderen, dass die Regeln und Heuristiken, die verwendet werden müssen, nicht alle Fälle abdecken können.
Es gibt noch ein entscheidendes Problem, das bestimmt der Grund war, warum nicht viele Namenerkennungs- bzw. Namenklassifikationsalgorithmen entwickelt werden, die mit einem deutschsprachigen Text zu tun haben. Das liegt nämlich hauptsächlich daran, dass in der deutschen Sprache keinen Unterschied zwischen Substantiven und Namen gibt, da beide großgeschrieben werden.

Kapitel 2 soll eine Motivation sein. In diesem Kapitel wird über die bekanntesten Namenerkennungssysteme gesprochen, nachdem die bei diesen Systemen verwendeten Ansätze vorgestellt werden. Das Konzept der Namenerkennung und die Beschreibung des allgemeinen Vorgehens werden der Inhalt des dritten Kapitels sein. Im Kapitel 4 wird der Einsatz des Namenerkennungswerkzeuges im Projekt META-AKAD erläutert. Kapitel 5 ist eine Zusammenfassung und ein Ausblick.

Kapitel 2 Motivation

In diesem Kapitel werden die bekanntesten Namenerkennungssysteme und die dabei verwendeten Ansätze vorgestellt.

2.1 Einführung

Ähnlich wie Data-Mining die Analyse strukturierter numerischer Daten bezeichnet, beschreibt der Begriff des Text-Mining eine Menge von Methoden zur (halb-) automatischen Auswertung großer Mengen natürlichsprachlicher Texte. Das Gebiet des Text Mining umfaßt vielfältige Methoden zur Extraktion von Informationen aus natürlichsprachlichen Texten. In diesem Gebiet bzw. in dem Forschungsgebiet der Namenserkennung und Namensklassifikation werden viele Algorithmen vorgestellt und dokumentiert. Bei diesen Namenerkennungssystemen, die Teile der sogenannten „Information-Extraction-Systems“ sind, werden hauptsächlich zwei Ansätze verwendet (Abbildung 1) und zwar der sogenannte „**Knowledge-Engineering-Approach**“ und „**Automatic-Training-Approach**“ [AI1999].

Abbildung 1 Zwei Verfahren Bei den Namenerkennungssystemen

Knowledge-Engineering-Approach:

```
- Die Grammatiken und Regeln sind von Hand erstellt,
- der Einsatzbereich der Regeln wird von einem Experten durch
  Selbstprüfung und Untersuchung von Sammlungen entdeckt,
- manuelle Nachbearbeitung (Test und Debug) notwendig.
```

Automatic-Training-Approach:

```
- Statistische Methoden falls möglich benutzen,
- Regeln aus den Sammlungen erlernen,
- Regeln aus Dialogen mit dem Benutzer erlernen.
```

❑

Bei dem Knowledge-Engineering-Approach wird, mit Hilfe einer Sammlung aus bekannten Namen, eine Menge von Regeln und Heuristiken entwickelt, um bei der Namenserkennung z.B. Namen zu extrahieren und in verschiedenen Kategorien zu klassifizieren.

Dieses Verfahren braucht aber eine qualifizierte Person mit genauen Kenntnissen über das Information-Extraction-System (IE-System) und auch eine manuelle Nachbearbeitung durch Personen, die Test und Debug Aufgaben wahrnehmen. Ein System mit hohen Leistung zu bilden, ist meistens ein iterativer Prozess, in dem der Entwickler eine Menge von Regeln verfasst, die dann getestet werden. Die durchgeführten Tests haben wiederum Einfluss auf die ursprünglichen Regeln d.h. die Testergebnisse bestimmen, ob die Regeln (Heuristiken) behalten oder geändert werden sollen.
Bei dem Automatic-Training-Approach verhält es sich anders. Es ist dabei nicht nötig, eine Person zu haben, die viel Erfahrung mit dem IE-System hat, sondern eine Person, die fähig ist, ein Tainingspaket aus mehreren Texten zu bilden. Auf diesem Trainingspaket läuft dann ein Trainingsalgorithmus, der auf mathematischen und statistischen Funktionen basiert. [AI1999]

Beide Verfahren haben sowohl Vorteile als auch Nachteile (Abbildung 2 und 3). Der Knowledge-Engineering-Approach hat den Vorteil, dass heutzutage die besten Information-Extraction-Systeme dieses Verfahren benutzen, obwohl beim [MUC 7] die Systeme, die mehr Leistung gebracht haben, solche sind, die den Automatic-Training-Approach benutzt haben. Allerdings braucht der Knowledge-Engineering-Approach einen aufwändigen Test-und-Debug Zyklus. Dieses Verfahren braucht auch externe Ressourcen und qualifizierte Personen, die fähig sind, Regeln zu erstellen. Außerdem hängt die Anwendbarkeit der erstellten Regeln stark von der Benutzten Sprache ab, d.h. die Auswirkung der Regeln auf Texte in English sind anders als deren Auswirkung auf Texte in Deutsch. Wie im Fall der Substantiven und Namen: Namen werden im Englischen großgeschrieben, dagegen werden im Deutschen sowohl Namen als auch Substantive großgeschrieben.

Abbildung 2 Knowledge-Engineering-Approach

Vorteile:

- Mit Geschicklichkeit und Erfahrung sind gute Systeme, die dieses Verfahren nutzen, nicht schwer zu entwickeln.
- Die Systeme, die mehr Leistung gebracht haben, sind solche, die dieses Verfahren benutzt haben.

Nachteile:

- Sehr mühsamer Entwicklungsprozess.
- Änderungen bei den Anforderungen können schwer angepasst werden.
- Das erforderliche Expertenwissen muss vorhanden sein.

❑

Die Vor- und Nachteile beim Automatic-Training-Approach sind zu denen des Knowledge-Engineering-Approach komplementär (Abbildung 3). Bei dem zuerst genannten Verfahren konzentriert man sich nicht darauf Regeln zu erstellen, sondern auf die Produktion von Trainingsdaten. Die Trainingsalgorithmen laufen dann automatisch auf diesen Trainingsdaten und brauchen daher keine Intervention eines Entwicklers.

Dieses Verfahren ist zudem leicht auf verschiedene Sprachen zu übertragen. Beim Automatic-Training-Approach besteht allerdings das Problem, dass nicht genug Trainingsdaten verfügbar, sie zu teuer oder schwer zu haben sind.

Abbildung 3 Automatic-Training-Approach

Vorteile:

- Die Wiederverwendung ist möglich (z.B. Übertragung auf andere Sprachen).
- System-Expertise ist nicht für die Anpassung erforderlich.
- Die “Data-driven“ Regeln garantieren die Überdeckung aller Fälle.

Nachteile:

- Es sind nicht genug Trainingsdaten vorhanden. Sie sind zudem zu teuer oder schwer zu haben.
- Es muss eine große Menge von Trainingsdaten vorhanden sein.
- Änderungen bei den Anforderungen führen dazu, eine große Menge von Trainingsdaten wiederzuschreiben.

❑

2.2 Der „Knowledge-Engineering-Approach“

Den Knowledge-Engineering-Approach nennt man auch den „Handcrafted-Approach“, weil dieses Verfahren auf manuell definierten Regeln basiert. Diese Regeln hängen stark von der Intuition der Entwickler ab. Sie sind nicht immer richtig, da sie von vielen Parametern abhängen, wie z.B. der Sprache. Deshalb treten bei diesem Verfahren Konsistenz- und Wiederverwendbarkeitsprobleme auf. In Beispiel 1 sind Regeln, die im Rahmen eines Knowledge-Engineering-Verfahren erstellt worden sein könnten.

Beispiel 1

- Akdemischer Titel + “Capitalized_Word” ---> Title + Personenname

  z.B: Prof. Härder

- Anrede + “Capitalized_Word” ---> Anrede+ Personenname

  z.B: Mr. Jones

- Tätigkeit + “Capitalized_Word” ---> Tätigkeit + Personenname

  z.B: Kanzler Schröder

❑

Bei diesem Vefahren kann also eine Heuristik so definiert werden: Ein Nachname ist ein Wort, das nach einem Akademischer Titel, einer Anrede oder einer Tätigkeit vorkommt.

2.3 Der „Automatic-Training-Approach“

Zwei automatische Methoden, die bei Namenerkennungssystemen eingesetzt werden können werden in diesem Abschnitt vorgestellt.

2.3.1 Decision-Trees

Zitat

“A decision-tree asks questions about an event, where the particular question asked depends on the answers to previous questions, and where each question helps to reduce the uncertainty of what the correct choice or action is.“ [MAG1994] d.h. man versucht in jedem Schritt Fragen zu stellen. Diese Fragen hängen von der Antworten der vorherigen Fragen stark ab. Auf diese Fragen versucht man stets Antworten zu geben und Wahrscheinlichkeiten zu bestimmen. Diese Wahrscheinlichkeiten können dann dabei helfen, die richtige Lösung zu treffen.

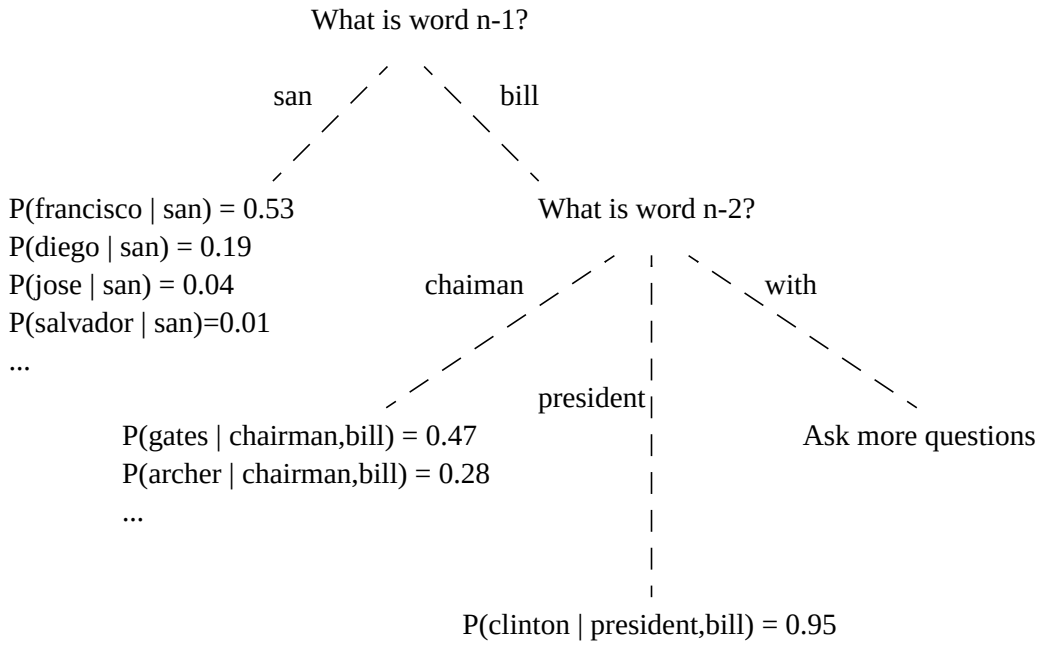
Elemente eines “Decision-Tree”

Ein „Decision-Tree“ (Beispiel 2) besteht aus den folgenden drei Elementen:

- Zukunft: Die möglichen Ausgaben des Decision-Tree-Modells.
  Eine Zukunft weist auf ein Element aus einer Auswahlliste hin, die der “Decision-Tree” anbieten kann. Diese Auswahlliste ist das Vokabular der Zukunft. Jedes Blatt des Baumes ist mit einem Vorschlag aus der Auswahlliste verbunden.
- Geschichte: Die verfügbaren Informationen zum Modell. Diese Informationen können aus dem vorherigen, aktuellen und nachfolgenden Knoten abgeleitet werden. Zur Geschichte des Baumes gehören die syntaktischen und semantischen Informationen des Baumes, dessen Struktur, die Anzahl der Knoten im Baum, das Auftreten zweier Knoten des Baumes in einer Beziehung zueinander etc.
- Fragen: Das Decision-Tree-Modell basiert sich auf Fragen. Jede gegebene Antwort erzeugt einen neuen Knoten im Baum. Das Ziel eines Decision-Tree-Algorithmus ist, die beste Fragenfolge über die Vergangenheit zu finden, um die Zukunft zu bestimmen. [BOR1999]

  Sobald ein Decision-Tree aufgebaut ist, dann ist seine Nutzung sehr einfach. Wir stellen uns jetzt ein Modell vor, das versucht das nachfolgende Wort in einem Text mit einer Geschichte, die aus den zwei vorherigen Wörter besteht, zu bestimmen.
  Eine mögliche Decision-Tree-Implementierung kann wie in Beispiel 2 aussehen. Man erhält am Ende die Wahrscheinlichkeiten, die in den Blättern des Baumes dargestellt werden. So ist im Beispiel *P(clinton | president,bill) = 0.95* die Wahrscheinlichkeit, mit der der Begriff *“clinton”* nach dem Begriff *“president bill”* vorkommt.

Beispiel 2 **Decision Tree Beispiel**

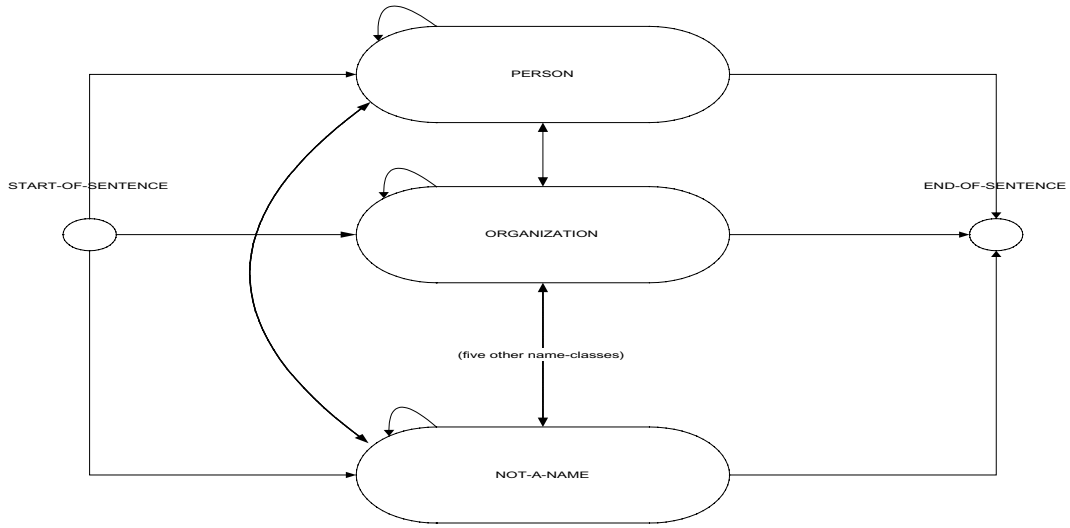


❑

2.3.2 Hidden Markov Models

Im Gebiet Namenerkennung verwenden viele Algorithmen auch das Hidden-Markov-Modell (HMM). Bei dem HMM hängt der aktuelle Zustand eines Systems nur vom vorherigen Zustand ab und jedes Wort ist im Modell durch einen einzigen Zustand repräsentiert, und es existiert eine Wahrscheinlichkeit, die mit dem Übergang vom aktuellen Wort zum nächsten assoziiert wird. Der Einsatz dieses Modells wird an Hand eines Beispiels aus dem IdentiFinder System, das von BBN Systems & Technologies entwickelt wurde [BSW1997], vorgestellt. Dieses Modell erlaubt aber nicht nur die Extraktion von Personennamen, sondern auch andere Namenklassen wie Organisation und Datum (Abbildung 4). Das IdentiFinder-System sagt die nächste Namenskategorie voraus, gestützt auf das vorherige Wort und die vorherige Namenskategorie.

Abbildung 4 Das HMM-Modell beim Identifinder-System [BSW1997]



❑

Eine Version des Identifinder-Systems basiert sich auf die zwei Gleichungen in Abbildung 5

Abbildung 5 Die wichtigsten Gleichungen beim Identifinder-System [BOR1999]

$$P(NC \mid NC\text{-}1, w\text{-}1) = c(NC, NC\text{-}1, w\text{-}1) / c(NC\text{-}1, w\text{-}1)$$

$$P(\langle w, f\rangle \mid \langle w, f\rangle\text{-}1, NC) = c(\langle w, f\rangle, \langle w, f\rangle\text{-}1, NC) / c(\langle w, f\rangle\text{-}1, NC)$$

Dabei werden folgende Definitionen verwendet:

NC = Aktuelle Namenklasse
NC-x = Namenklasse des Wortes, das als x-tes Wort vor dem aktuellen Wort auftritt
c(W) = Anzahl des Auftretens des Wortes W in der Training-Sammlung (Trainingsdaten)
w = Ein Wort
f = Ein Feature

❑

In Beispiel 3 werden “capitalized“ als Feature und “organization_name“ bzw. “person_name“ als Namenklassen benutzt. Man berechnet mit Hilfe der in Abbildung 5 eingeführten Gleichungen und einer Sammlung von Trainingsdaten die Wahrscheinlichkeiten, dass das Wort “ANDERSON” ein Organisationname bzw. Personenname sein kann, falls “ANDERSON” nach dem Wort “ARTHUR” vorkommt. Hier kann man “ANDERSON” eher als Organisationname betrachten und nicht als Personenname.

Beispiel 3

```
P(<anderson,capitalized>|<arthur,capitalized>-1,organization_name)
> (größer als)
P(<anderson,capitalized>|<arthur,capitalized>-1,person_name)
```

❑

2.4 Vorstellung der Namenerkennungssysteme

In dem Forschungsgebiet „Information Extraction“ werden viele Systeme vorgeschlagen, die sich nicht nur auf Personennamen beschränken, sondern auch andere Namen wie Organisationnamen und Firmennamen berücksichtigen.
Einige dieser Systeme sind auch auf dem Markt als konkrete Produkte zu finden. Diese benutzen meistens den „Knowledge-Engineering-Approach“, wie SRI’s FASTUS[1] [FASTUS] und TextPro[2]. Dieses Verfahren wird auch von „Nominator“ [RW1997] verwendet. Nominator ist ein Modul, das für die Extraktion der Namen aus einem englischen Text entwickelt und in ein IBM Produkt integriert wurde. Nominator findet Namen in einem Text und entscheidet dann zu welcher Namenklasse sie gehören. Dabei werden folgende Schritte durchgeführt: Zuerst wird eine Liste mit Namenskandidaten aus dem ganzen Dokument gesammelt. Dann werden die Namenkandidaten miteinander verglichen und die Liste wird verfeinert. Danach wird die Liste in Gruppen aus äquivalenten Namen aufgeteilt und jeder Gruppe wird ein Namenstyp zugeordnet wie Person, Organisation etc. Nominator wird auch vom Fachbereich Computer-Science an der Universität Pennsylvania benutzt [PEN] und wird in ein System integriert, das in [MUC 6] vorgestellt wird.
In [MUC 7] werden auch andere Systeme vorgestellt. Die meisten davon benutzen das Hidden-Markov-Modell benutzt wie IdentiFinder von BBN Systems & Technologies [BSW1997] und das MENE System, das an der Universität New York entwickelt wurde [BSAG1998]. Es gibt aber in [MUC 7] auch Systeme, die den Knowledge-Engineering-Approach verwenden wie FACILE [BRM1998] und IsoQuest [KH1998].

1. http://www.ai.sri.com/~appelt/fastus.html
2. http://www.ai.sri.com/%7Eappelt/TextPro/

Alle Systeme, die oben vorgestellt wurden funktionieren nur auf englische Texte, deshalb wird die Aufgabe später in MET (Multilingual Entity Task) [MUC 7] verallgemeinert. Es werden aber nur Japanisch, Chinesisch und Spanisch betrachtet. Auf englische Texte arbeiten diese Systeme sehr gut, aber sie versagen bei anderen Sprachen wie auch der deutschen Sprache, da alle Namen und Substantive großgeschrieben werden. Das ist im Englischen nicht der Fall.

In diesem Zusammenhang kommt unser Namenerkennungswerkzeug, das mit Hilfe vonWörterbüchern das Ziel hat, Personennamen sowohl aus einem englischen Text als auch aus einem deutschen Text zu extrahieren. Dieses Werkzeug folgt auch den Knowledge-Engineering-Approach, da es auf einer Menge von Regeln und Heuristiken basiert. In Kapitel 3 wird das Konzept der Namenerkennung näher betrachtet.

Kapitel 3

Beschreibung des Namenerkennungs-werkzeuges

In diesem Abschnitt wird das Konzept (Algorithmus) des implementierten Namenerkennungswerkzeuges beschrieben. Bei diesem Werkzeug handelt es sich um ein System, das die Aufgabe hat, Personennamen aus einem unstrukturierten bzw. semi-strukturierten Dokument zu extrahieren. Dieses System basiert auf eine Menge von Heuristiken, die auf eine Menge von Zeichenketten (Tokens) angewendet werden. Dabei wird kein Datenbankensystem verwendet, da dieses Werkzeug unabhängig von diesen Mitteln arbeiten soll. Aber es werden Wörterbücher verwendet, die allerdings nichts anderes als eine Liste von Textdateien sind, die Vornamen, Nachnamen, Honorare (z.B. Dr., Prof.) enthalten.

3.1 Einführung

In dem Forschungsgebiet, das sich mit der Namenerkennung und Namenklassifikation beschäftigt, werden viele Algorithmen vorgestellt und dokumentiert, um englische Texte zu verarbeiten. Allerdings lassen sich diese Algorithmen nicht unverändert auf deutschsprachige Texte anwenden. Das liegt hauptsächlich daran, dass in der deutschen Sprache keine Unterscheidung zwischen Substantiven und Namen gemacht wird, da beide großgeschrieben werden.

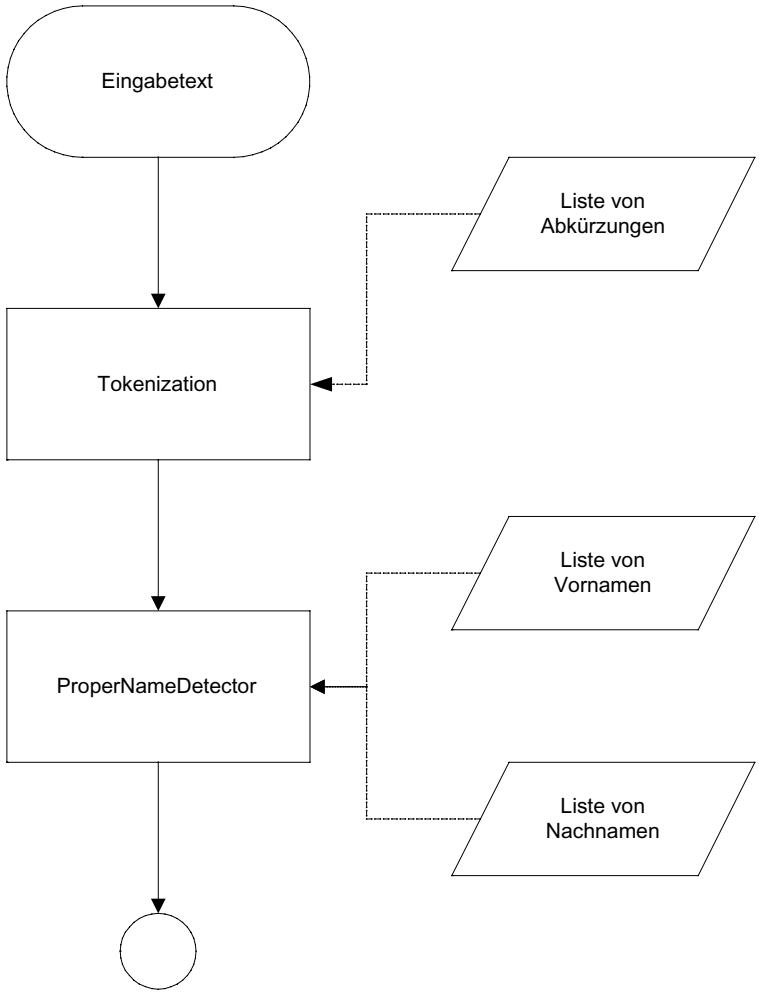
Das hier vorgestellte Namenerkennungswerkzeug basiert auf der Beobachtung, dass in einem Text ein Nachname entweder nach dem eigenen Vornamen steht oder mit einem akademischen Titel wie Dr., Prof. etc... eingeführt wird. Es kommt aber auch vor, dass ein Nachname nach einem Wort zu finden ist, das eine Tätigkeit repräsentiert wie Kanzler, Personalleiter, Minister etc..
Dieses Werkzeug ist also eine Kombination von Vornamen-Listen, Nachnamen-Listen, Honorar-Listen und selbst definierten Heuristiken (Regeln), wie z.B. „Ein nach einem Vorname, großgeschriebenes Wort ist ein Nachnamekandidat“ oder „Ein Sonderzeichen vor einem Nachnamekandidat hebt diese Hypothese auf“. Auch Matching-Regeln auf Vornamen und Nachnamen werden verwendet. Mit Matching ist die Untersuchung einer vorhandenen Datei nach Präsenz eines Eintrags, der dem aktuellen Token ähnlich ist, gemeint.

Im folgenden Abschnitt wird das allgemeine Vorgehen erläutert. Nach einer Übersicht werden die einzelnen Schritte des Systems genauer betrachtet.

3.2 Beschreibung des allgemeinen Vorgehens

Die Erkennung der Personennamen aus einem unstrukturierten bzw. semi-strukturierten Dokument folgt mittels einer Komponente namens ProperNameDetector. Diese Komponente bekommt als Eingabe eine Liste von Tokens und verwendet eine vorgegebene Liste mit Vornamen und eine andere mit Nachnamen (Abbildung 6).

Abbildung 6 Überblick



❑

Die Abbildung 6 ist ein Überblick über die Funktionalität des Namenerkennungswerkzeuges. Dieses Werkzeug versucht zuerst den Eingabetext in eine Folge von Token umzuwandeln. Dies wird durch die Tokenization-Komponente realisiert (3.2.1). Dann wird der ProperNameDetector eingesetzt, um Personennamen zu extrahieren (3.2.2).

3.2.1 Die Tokenization-Komponente

Vorbereitung

Bei der sogenannten „Tokenization“ handelt es sich um die Übersetzung eines gegebenen Textes in Tokens, d.h. die Zerlegung des Textes in elementare Textteilen. Ein Token kann ein Wort oder ein Sonderzeichen sein. Allerdings im Fall des Punktes wird zwischen Punkt und Pünktchen unterschieden. Darüber wird im nächsten Abschnitt (Tokenization) gesprochen. Der Eingabetext kann entweder ein Fließtext, oder ein semi-strukturiertes Dokument wie eine HTML- oder XML-Datei sein. In beiden Fällen wird die Eingabe als eine einzige Zeichenkette betrachtet und daraus werden Tokens extrahiert. Im Fall eines Fließtextes wird nichts zusätzlich gemacht, da dieser Text selbst nichts anders als eine Zeichenkette ist. Aber im Fall eines semi-strukturierten Dokuments wird nur der Text also keine Tag-Namen oder Tag-Attibute extrahiert. Die Vorbereitungen, die dazu führen, aus einem semi-strukturierten Dokuments einen Text zu gewinnen, werden in Kapitel 4 näher betrachtet. Erst nach dieser Umwandlung werden die Tokenization-Regeln eingesetzt. Hier wird zunächst auf die Tokenization selbst konzentriert.

Als Beispiel wird der HTML-Code in Beispiel 4 verwendet.

Beispiel 4

```
<TR><TD><A HREF="/staff/Haerder"
CLASS="subtopic">Prof.Dr.Theo Härder</A></TD></TR>
<TR HEIGHT="5"><TD HEIGHT="5"></TD></TR>
<TR><TD><A HREF="/staff/Ritter/"
CLASS="subtopic">Dr. Norbert Ritter</A></TD></TR>
<TR HEIGHT="5"><TD HEIGHT="5"></TD></TR>
<TR><TD><A HREF="/staff/Flehmig/"
CLASS="subtopic">Marcus Flehmig</A></TD></TR>
```

❑

Der Code wird zu folgender Zeichenkette umgewandelt:

```
Prof. Dr. Theo Härder. Dr. Norbert Ritter. Marcus Flehmig.
```

Tokenization

Bei der Tokenization selbst muss man beachten, dass die Wörter von der Zeichensetzung getrennt werden müssen. An dieser Stelle merkt man aber, dass in einem Text ein Punkt einmal als eine Abschließung eines Satzes oder auch nach einer Abkürzung (wie z.B. Dr. oder Prof.) vorkommen kann. Wir benutzen daher eine Liste mit möglichst vielen Abkürzungen der deutschen Sprache, um den Unterschied zwischem einem Punkt (englisch „full stop“) und einem Pünktchen (englisch „dot“) zu identifizieren. In dem zuletzt genannten Fall muss das Pünktchen als ein Teil des Wortes und nicht als ein neues Token betrachtet werden. Also muss das Ergebnis, wenn man zum Beispiel den Satz „*Prof. Dr. Theo Härder. Dr. Norbert Ritter. Marcus Flehmig.*“ in Tokens übersetzen will, so aussehen:

Prof.
Dr.
Theo
Härder
.
Dr.
Norbert

Ritter
.
Marcus
Flehmig
.

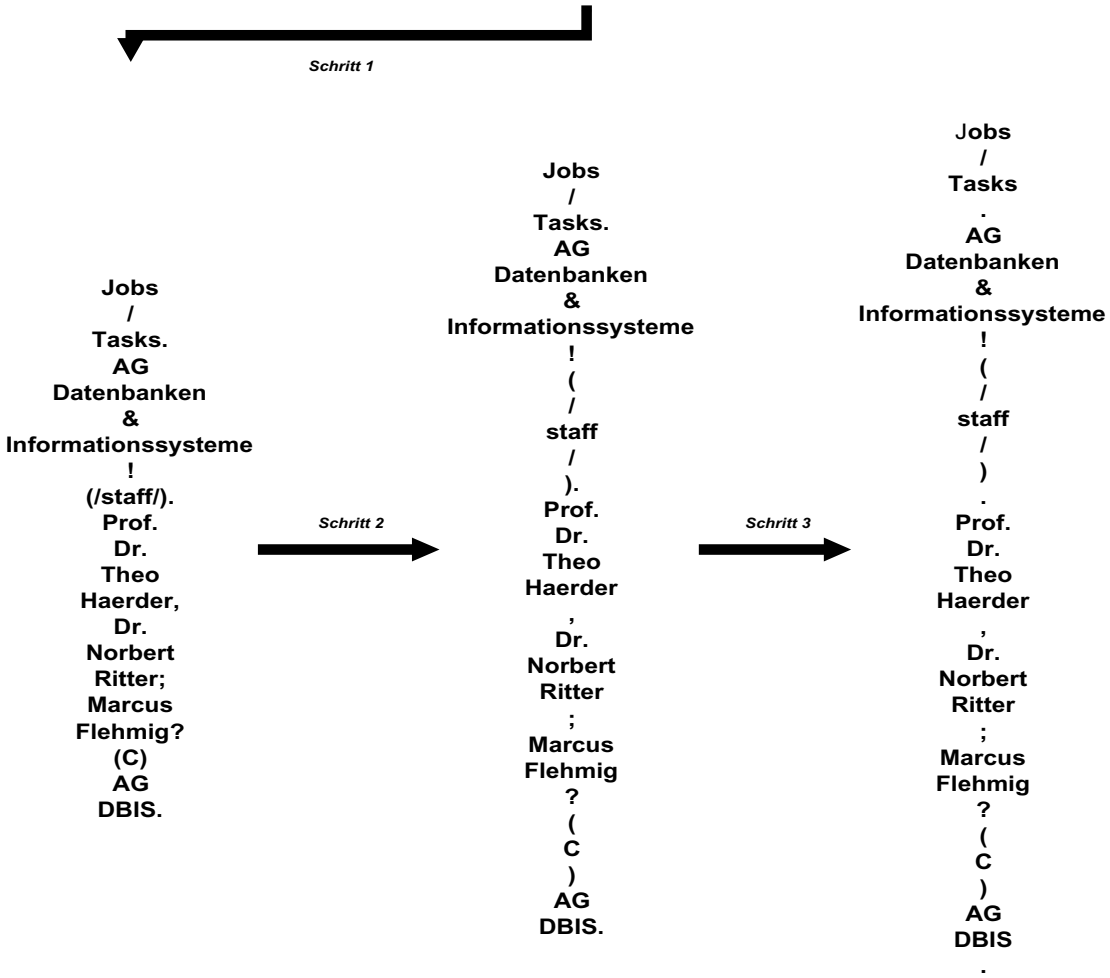
Bei der Tokenization sind die folgenden Schritten durchzuführen:

- Schritt 1: Das Leerzeichen ist das Trennsymbol zwischen den verschiedenen Tokenkandidaten. Nach diesem ersten Schritt erhält man dann eine Menge von Zeichenketten, die sowohl aus Buchstaben als auch aus Sonderzeichen bestehen. Diese werden als Tokenkandidaten betrachtet und weiter bearbeitet.
- Schritt 2: Dann werden die Tokenkandidaten auf Präsenz von Sonderzeichen außer dem Punkt geprüft. Wenn das der Fall ist, dann werden diese Sonderzeichen vom Wortkörper getrennt und als neue Tokens betrachtet. Die Punkte werden in diesem Schritt nicht als Token gesehen, es ist nämlich zuerst die Unterscheidung zwischen einem Punkt und einem Pünktchen zu machen.
- Schritt3: Die so erhaltenen Tokenkandidaten werden nun auf Präsenz des Punktes geprüft. Hier kommt die Liste der Abkürzungen zum Einsatz. Wenn der Punkt nach einer Abkürzung vorkommt, dann wird er als Teil vom Token gesehen. Ansonsten wird dieser Punkt als ein neues Token betrachtet und wird daher vom Wort getrennt.

In Beispiel 5 sind die Schritte der Tokenization zu sehen. Die Zeichenkette, die in diesem Beispiel verwendet wird, wurde selbst konstruiert und nicht aus einem Html-Dokument extrahiert. Sie dient nur zur Erläuterung der Schritte, die bei der Tokenization durchgeführt werden müssen. Nach dem ersten Schritt bestehen die Tokenkandidaten aus Buchstaben und Sonderzeichen. Nach dem zweiten Schritt werden die Sonderzeichen bis auf dem Punkt als eigenständige Tokens betrachtet. Nach dem dritten Schritt werden die Punkte, die nicht nach Abkürzungen vorkommen, auch als Tokens gesehen.

Beispiel 5 **Schritte der Tokenization**

Aus "Jobs / Tasks. AG Datenbanken & Informationssysteme! (/staff/). Prof. Dr. Theo Haerder, Dr. Norbert Ritter; Marcus Flehmig? (C) AG DBIS."
erhält man dann:



❑

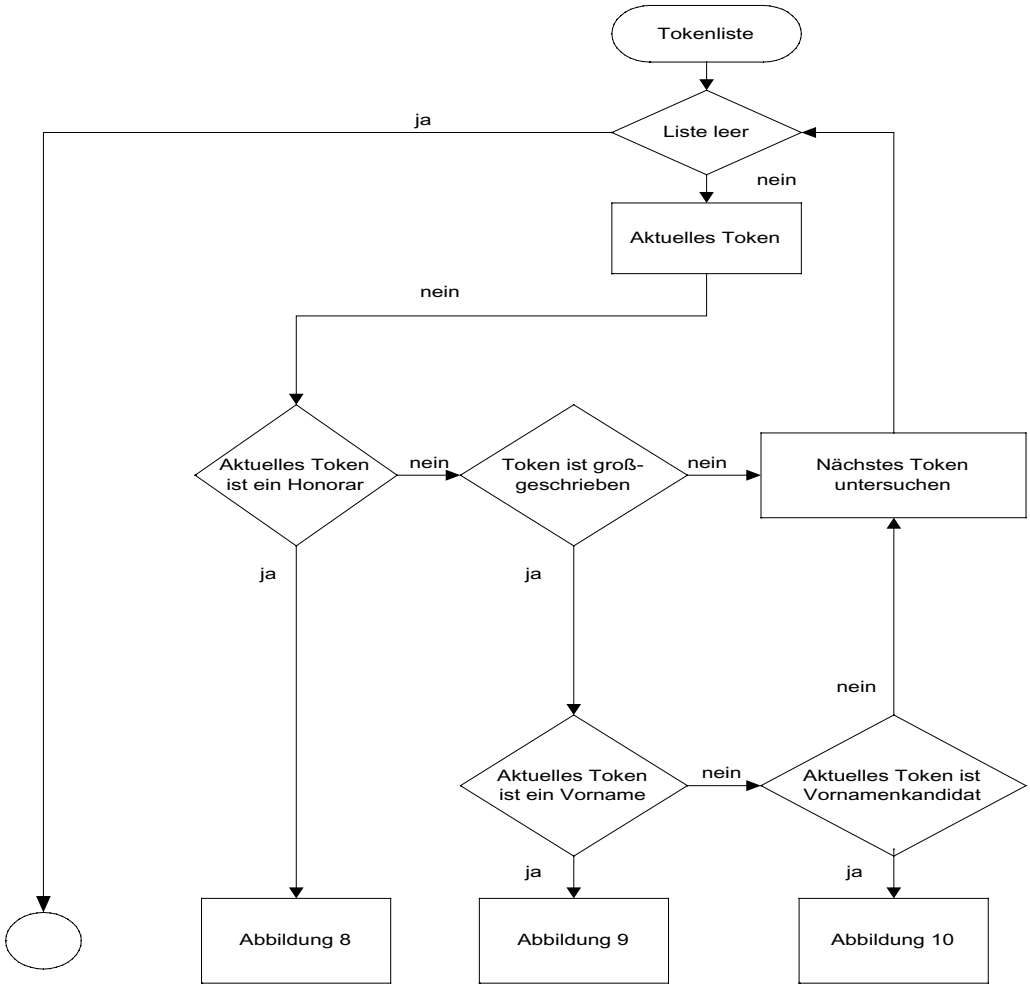
3.2.2 Die ProperNameDetector-Komponente

Bei dem sogenannten „ProperNameDetector“ handelt es sich um ein Verfahren zur automatischen Extraktion von Personennnamen aus einer gegebenen Tokenliste.
In der deutschen Sprache ist die Aufgabe, Personennamen aus einem Textdokument zu erkennen und zu extrahieren nicht einfach, da sowohl Namen als auch Substantive großgeschrieben werden. Zudem treten noch zwei andere Probleme auf: Zum einen fällt die Entscheidung schwer, ob ein großgeschriebes Wort tatsächlich ein Vorname oder ein Nachname ist, da nie eine Datenbank mit Vornamen bzw. Nachnamen vollständig sein kann. Das zweite Problem besteht darin zu erkennen, ob zwei großgeschriebene Wörter, die in einem Text nacheinander auftreten, Bestandteile eines Personennamen sind. Um diese Schwierigkeiten zu umgehen braucht man daher unbedingt Regeln, die es leichter machen, solche Entscheidungen zu treffen. Diese Regeln sollen es erlauben, Vornamen bzw. Vornamenkandidaten und Nachnamen bzw. Nachnamenkandidaten zu erkennen und Personennamen bestehend aus einem Vornamen und einem Nachnamen zu identifizieren.
ProperNameDetector ist also ein regelbasierendes Verfahren, bei dem Wörterbücher und Heuristiken gebraucht werden. Da dieses Verfahren auf Regeln beruht, braucht man einige Beobachtungen, um die notwendigen Regeln zusammenzustellen. Unter diesen Beobachtungen ist z.B., dass ein Nachname ein großgeschriebenes Wort ist, das entweder nach dem eigenen Vornamen oder nach einem akademischen Titel wie Prof., Dr. oder auch nach einem den Beruf beschreibenden Begriff (wie Leiter) vorkommt. Diese Beobachtungen erlauben es u.a., einige Heuristiken abzufassen, um Personennamen zu identifizieren. Eine dieser Heuristiken könnte sein, dass z.B. ein großgeschriebenes, nach einem Vornamen auftredendes Wort ein Nachnamenkandidat ist. Bei der Erstellung solcher Heuristiken braucht man allerdings Wörterbücher, da sonst die Aufgabe der Personennamenserkennung wegen den oben und in Kapitel 1 beschriebenen Problemen sehr schwer wird . Deswegen haben wir in dieser Arbeit mehrere Wörterbücher eingesetzt. Diese werden mit Java-Programmen und Perl-Skripten nachbearbeitet und in kleineren Liste untergeteilt, um die Suche effektiver und schneller zu machen.
Im folgenden Abschnitt werden an Hand von Beispielen diese Regeln und die Bedingungen beim Einführen dieser Regeln genauer betrachtet. Wie schon erwähnt wurde, kommt der „ProperNameDetector“ zum Einsatz, nachdem man das Eingabe-Dokument in Tokens übersetzt hat. Deshalb ist eine gute Tokenization eine Basis für ein gutes Ergebnis. Am Anfang des ProperNameDetector-Prozesses hat man also eine Liste von Tokens, die auf Präsenz von Personennamen getestet werden soll. Diese Tokenliste muss nicht nur die großgeschriebenen Wörter enthalten sondern auch die Sonderzeichen und die kleingeschriebenen Wörter. Durch die erstellten Regeln wird nämlich betrachtet, was nach dem aktuellen Token vorkommt und erst dann kann entschieden werden, ob ein Token, als Teil eines Personennamens zu sehen ist. Die Tokenliste wird durchlaufen. Falls das aktuelle Token ein Sonderzeichen oder ein kleingeschriebenes Wort ist, dann macht man weiter mit dem nachfolgenden Token. Beim Betrachten des aktuellen Tokens kann man in eine der folgenden drei Fällen geraten:

- Aktuelles Token ist ein Honorar,
- aktuelles Token ist ein Vorname oder
- aktuelles Token ist ein Vornamenkandidat.

Die Abbildung 7 beschreibt den generellen Algorithmusablauf.

Abbildung 7 genereller Algorithmusablauf

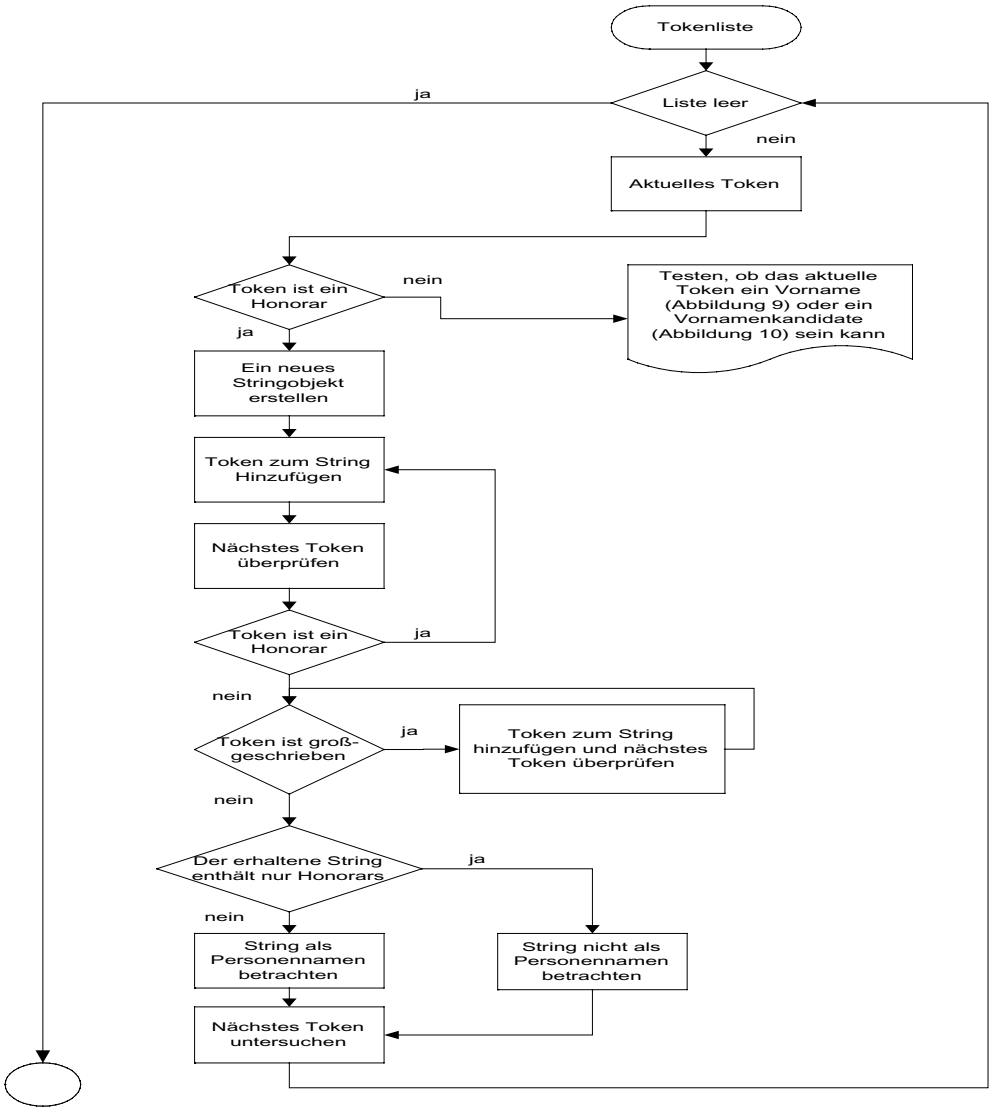


❑

Aktuelles Token ist ein Honorar

Ein Eigenname wird sehr oft mit einem Honorar wie Dr. oder Prof. eingeführt. Deshalb ist ein großgeschriebenes Wort nach einem Honorar sehr wahrscheinlich ein Teil eines Personennamens. Diese Beobachtung wird hier verwendet. Dazu braucht man ein Honorar-Wörterbuch d.h. eine Liste mit Wörtern, die einen akademischen Titel oder eine Tätigkeit beschreiben können. Diese Liste wird selbst erstellt und enthält eine Menge solcher einführenden Wörter wie Dr., Prof., Dr.-Ing., Dipl.-Inform., President, Manager, Hiwi. etc.

Abbildung 8 Algorithmusablauf im Fall, dass das aktuelle Token ein Honorar ist



❑

Wenn also das aktuelle Token ein Honorar ist, dann werden mit Hilfe des Honorar-Wörterbuches folgende Schritte durchgeführt (Abbildung 8):

1. Es wird eine Liste erstellt und das aktuelle Honorar-Wort wird zu dieser Liste hinzugefügt.
2. Das in der Token-Liste folgende Token wird getestet. Falls es auch ein Honorar ist, wird es auch zu der Liste hinzugefügt.
3. Falls das nachfolgende Token ein großgeschriebenes Wort oder eine Buchstabe gefolgt von einem Pünktchen ist, wird es als Teil des Personennamens betrachtet und zu der Liste hinzugefügt.
4. Es werden höchstens drei solcher Token betrachtet, da angenommen wird, dass ein Personenname höchstens aus drei Wörtern besteht.
5. Die Suche wird bei Auftreten eines Sonderzeichens oder eines kleingeschriebenen Wortes unterbrochen.

   So findet man Personennamen, die mit einem Honorar eingeführt werden wie es das Beispiel 6 zeigt.

Beispiel 6

```
Prof. Dr. Theo Härder
Dr. Norbert Ritter
Prof. Dr. K. Madlener
Dipl. Inform. Marcus Flehmig
```

❑

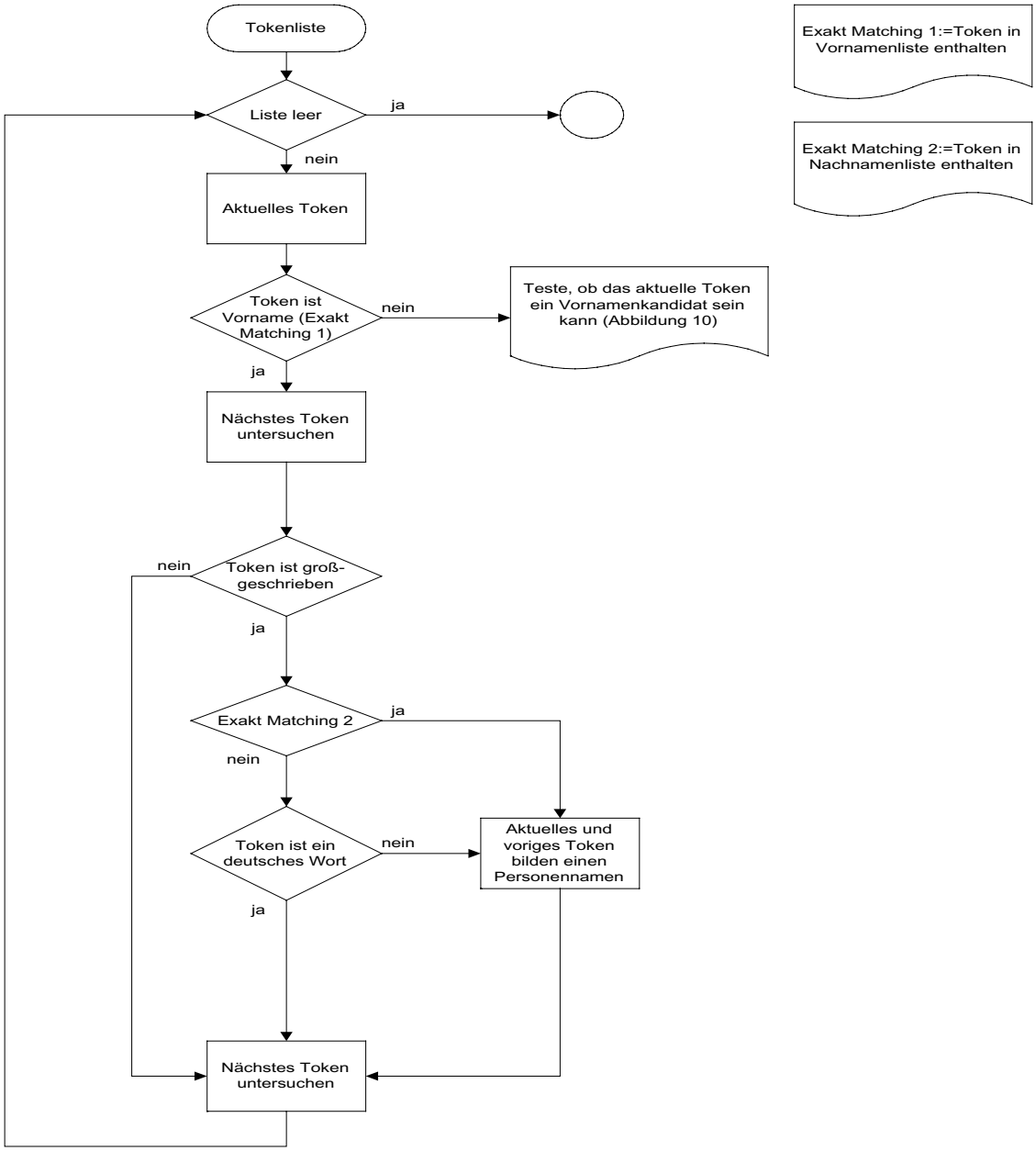
Aktuelles Token ist ein Vorname

Man kann bemerken, dass in einem Dokument ein Nachname immer nach dem zugehörigen Vornamen vorkommt, es sei denn dieser Nachname wurde irgendwo in demselben Dokument mit einem Vornamen oder einem Honorar schon erwähnt. Mit dieser Beobachtung kann man sich ein Verfahren überlegen, das zuerst entscheidet, ob ein Token ein Vorname sein kann und dann untersucht, ob das nachfolgende Token als Nachname zu sehen ist. Die Anzahl der Vornamen ist aber sehr gross und man ist nicht in der Lage sie alle in einem Wörterbuch zusammenzufassen. Eine Verwendung eines solchen Hilfsmittels ist aber nötig, da man sonst nicht entscheiden kann, ob ein großgeschriebenes Wort ein Vorname ist. Diese Aufgabe ist zudem bei Sprachen viel schwieriger, die syntaktisch keinen Unterschied machen zwischen Namen und Substantiven wie im Fall der deutschen Sprache. Es wird also ein Wörterbuch mit Vornamen an dieser Stelle benutzt. Die Qualität des Ergebnisses hängt stark vom benutzten Wörterbuch ab. Je umfangreicher ein Wörterbuch ist, desto besser sind die Ergebnisse. Das hier verwendete Wörterbuch wird an der Universität von New Mexico (Computing Research Laboratory) erstellt und enthält eine Liste von Vornamen, die aus verschiedenen Sprachen stammen. Diese Liste mit vielen anderen Ressourcen für verschiedene Sprachen findet man unter[1] .

1. http://crl.nmsu.edu/Resources/resource.htm

Die Vornamenliste wird aufbereitet d.h. in kleineren Listen unterteilt und sortiert, um das Verfahren zu beschleunigen. Man gelangt auch zu diesem Ziel, indem man an einem bestimmten Punkt die Suche nach einem Vornamen im Wörterbuch unterbricht, so braucht man nicht die ganze Liste zu untersuchen und spart damit Zeit.

Abbildung 9 Algorithmusablauf im Fall, dass das aktuelle Token ein Vorname ist



❑

Das Beispiel 7 dient der Erläuterung der Heuristiken, die im Fall, dass das aktuelle Token ein Vorname ist, gebraucht werden.

Beispiel 7

```
1....weil Martin Software entwickelt.
2....in der Gottlieb Straße.
3.Amin Chatti...
4.Norbert Ritter...
5.Hans-Peter Steiert
```

❑

Wenn das aktuelle Token mehr als zwei Buchstaben enthält (Das wird benötigt, um den Fall zu vermeiden, dass ein Token eine großgeschriebene Präposition wie In, Zu, To etc. ist) und ein Vorname ist, d.h. es ist ein großgeschriebenes Wort und es kommt in der Vornamenliste vor, dann wird getestet, ob das folgende Token ein Nachname sein kann. Dieser Test wird auch durchgeführt, wenn das aktuelle Token ein zusammengesetztes Wort (wie Hans-Peter im Beispiel 7) ist, dessen erste oder zweite Teil ein Vorname ist.
Man kann an dieser Stelle eine Heuristik erstellen und sagen, dass wenn ein großgeschriebenes Wort in einem Text direkt nach einem Vornamen vorkommt, dann ist es ein Nachname. Das klappt leider nicht immer. Als Beispiel seien dazu die zwei ersten Fälle im oberen Beispiel genannt. In beiden Fällen sind Martin bzw. Gottlieb sichere Vornamen (sie kommen in der Vornamenliste vor), allerdings sind Software und Straße keine Nachnamen, obwohl sie direkt nach Vornamen vorkommen und beide großgeschrieben sind. Also muss man hier sich andere Heuristiken überlegen, um diesem Problem auszuweichen. Für diesen Zweck werden zwei weitere Listen gebraucht: Eine Liste mit Nachnamen und eine andere mit vielen deutschen Substantiven d.h. die großgeschriebenen Wörter, die im Deutschen keine Namen sind. Diese beiden Listen werden auch an der Universität von New Mexico erstellt und obwohl die Nachnamenliste nicht sehr umfangreich ist, kann man sie als Ausgangspunkt verwenden, um mit Existenztest-, Ähnlichkeitstest- und Matching-Algorithmen gute Ergebnisse zu erzielen. Dasselbe gilt auch für die Vornamenliste, da nicht alle Vornamen erwähnt werden. Diese Algorithmen werden später genauer betrachtet, weil zuerst die Idee der Heuristik vorgestellt werden soll (Abbildung 9).
Die Idee basiert auf der Beobachtung, dass es sehr wahrscheinlich ist, dass nach einem sicheren Vornamen (d.h. ein Vorname, der in der Vornamenliste zu finden ist) ein Nachname vorkommt. Daher werden die Akzeptanztests (d.h. ein Token als Teil eines Personenenamens zu sehen) nicht so durchgeführt wie im Fall, dass das aktuelle Token nur ein Vornamenkandidat ist (Ein Token ist ein Vornamekandidat, wenn es nicht in der Vornamenliste existiert, aber es wird nach weiteren Ähnlichkeits- oder Matchingprozeduren als Vorname betrachtet).

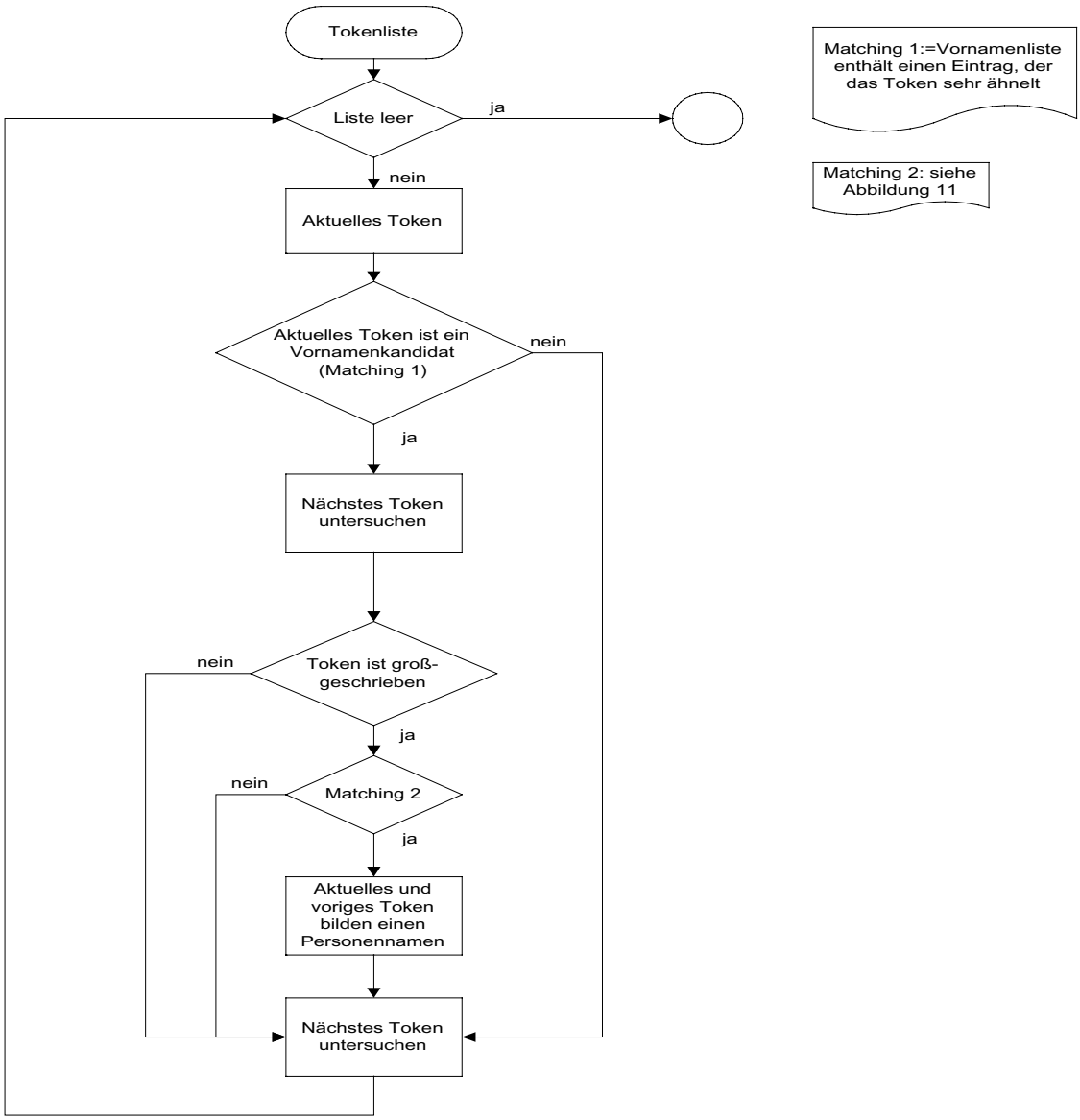
Man muss nur den Fall betrachten, dass der Nachnamenkandidat eigentlich kein Nachname ist, sondern er ist nur ein großgeschriebenes deutsches Wort, das aufgrund von grammatikalischen Regeln (wie Fall 1 im Beispiel 7 durch die Präsenz von „weil“) oder als Substantiv (wie Fall 2) nach einem Vornamen vorkommt. Auf den letzten Fall trifft man sehr oft besonders in der deutschen Sprache, in der alle Namen großgeschrieben werden. Deswegen haben wir noch ein Wörterbuch benutzt mit vielen Substantiven, die in der deutschen Sprache zu finden sind. Mit Hilfe dieses Wörterbuches bestand die anfängliche Idee, dass ein Nachnamenkandidat als Nachname betrachtet wird, wenn er nicht in der Liste ist, die die deutschen Substantiven enthält und zudem nach einem sicheren Vornamen vorkommt. Ein sicherer Vorname ist ein Vorname, der in der Vornamenliste existiert. Ein Nachnamenkandidat ist ein großgeschriebenes, keine Zahl enthaltendes Wort, das aus mehr als 2 Buchstaben besteht. So werden Software im Fall „...weil Martin **Software** entwickelt.“ und Straße im Fall „...in der Gottlieb **Straße**“ nicht als Nachnamen gesehen, obwohl die beiden nach sicheren Vornamen vorkommen.
Das Problem ist aber, wenn man diesen Ansatz verfolgt, dass Substantive, die in der Liste mit deutschen Substantiven existieren und gleichzeitig Nachnamen sein können (wie Ritter im Fall „Norbert **Ritter**...“ im Beispiel 7) nie ausgewählt werden. Und dieser Fall ist sehr häufig in der deutschen Sprache, in der viele Nachnamen sich aus deutschen Substantiven ableiten z.B. Ritter, Zimmermann, Schneider etc.
Damit auch diese Fälle in Betracht gezogen werden, braucht man an dieser Stelle die Nachnamenliste, die helfen kann, zu entscheiden, ob ein Nachnamenkandidat tatsächlich ein Nachname ist oder nur ein Substantiv. So wird zuerst getestet, ob Ritter, in der Nachnamenliste existiert. Ist das der Fall, dann wird Ritter als Nachname angenommen, obwohl dieses Wort in der Liste der deutschen Substantiven auch existiert.
Der noch verbleibende Fall ist der, dass ein Nachnamenkandidat weder in der Nachnamenliste noch in der Liste der deutschen Substantiven existiert (wie Chatti in „Amin **Chatti**...“ im Beispiel 7). In diesem Fall wird er automatisch als Nachname angenommen, weil er direkt nach einem sicheren Vornamen ist und es entsteht keine Gefahr, dass er als ein deutsches Substantiv vorkommt. So werden alle Fälle behandelt, bei denen ein Nachnamenkandidat direkt nach einem sicheren Vornamen vorkommt. Die dabei notwentigen Schritte werden hier zusammengefasst:

1. Testen, ob das aktuelle Token in der Vornamenliste existiert, also ein sicherer Vorname ist.
2. Wenn dies der Fall ist, dann wird das in der Tokenliste nachfolgende Token auf einem Nachnamenkandidat getestet. Wenn dieses Token direkt nach dem Vornamen vorkommt d.h. es existieren keine Sonderzeichen, die beide Tokens trennen, es außerdem großgeschrieben ist, aus mehr als 2 Buchstaben besteht und keine Zahl enthält, dann wird das Token als Nachnamenkandidat betrachtet.
3. Es wird zuerst mit den Existenz- und Ähnlichkeitsprozeduren getestet, ob der Nachnamenkandidat in der Nachnamenliste ist.
4. Ansonsten wird getestet, ob der Nachnamenkandidat nicht in der Liste der deutschen Substantiven ist.
5. Der Nachnamenkandidat wird als Nachname betrachtet, wenn er entweder den Test in 3. oder den in 4. besteht.

Aktuelles Token ist ein Vornamenkandidat

Die Abbildung 10 beschreibt den Fall, dass das Aktuelle Token kein sicherer Vorname ist d.h. es kommt nicht in der Vornamenliste vor, sondern es ist ein Vornamenkandidat d.h. es ähnelt einem Eintrag aus der Vornamenliste.

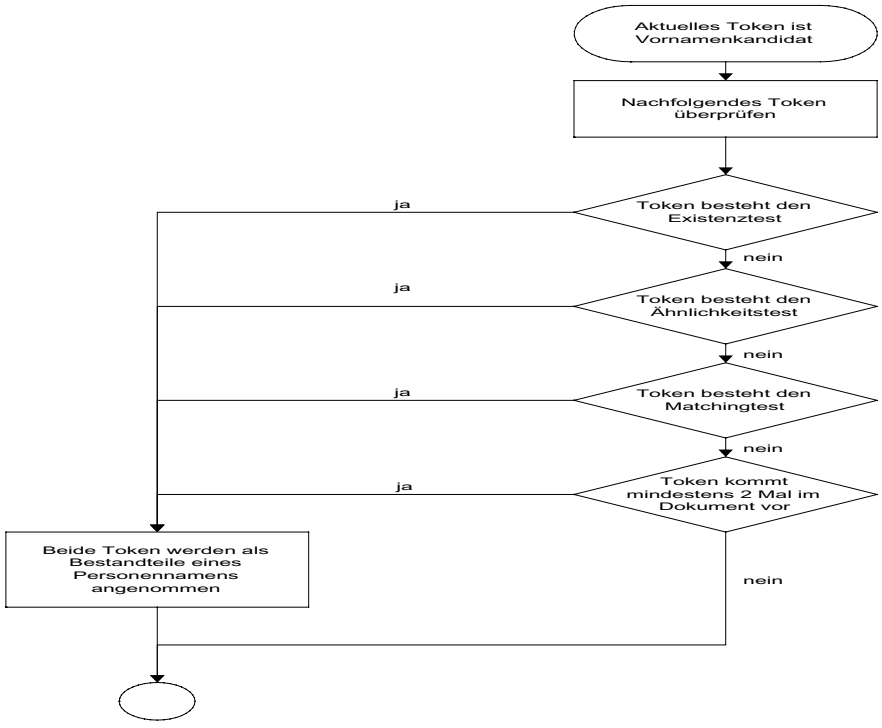
Abbildung 10 Algorithmusablauf im Fall, dass das aktuelle Token ein Vornamenkandidat ist



❑

Wie schon erwähnt wurde, ist das hier gebrauchte Vornamen-Wörterbuch nicht sehr umfangreich. Es existiert außerdem keines, das alle Vornamen enthalten kann. Das liegt hauptsächtlich daran, dass die Anzahl der Vornamen weltweit nicht beschränkt ist. Die Vornamen kommen in verschiedenen Formen vor (ein Nachname wie Mayer z.B. kann auch als Meyer, Maier oder Meier vorkommen) und sie können auf vielen Arten kombiniert werden wie Hans-Peter und Hans-Jürgen. Und das allein macht es sehr schwer, alle Vornamen in einem Wörterbuch zusammenzufassen. Dieses Problem gibt es auch bei den Nachnamen, weil sie wie die Vornamen kombiniert werden können. Daher ist der Existenztest (d.h. das einfache Prüfen, ob ein Token in der Vornamenliste vorkommt), das in diesem System durchgeführt wird, nicht genügend für eine optimale Lokalisierung aller Namen in einem Dokument, weil die benutzten Namenressourcen nicht vollständig sein können. Um möglichst viele Namen in einem Dokument zu finden, sind neben der Existenztest-Prozedur andere Prozeduren zu entwickeln, damit man die Vornamen bzw. Nachnamen, die nicht in der Vornamen- bzw. Nachnamenliste zu finden sind, auch identifizieren kann. Die Ähnlichkeitstestprozeduren und die Matchingprozeduren spielen also eine große Rolle, um die Lücken zu decken, die wegen der Unvollständigkeit der vorhandenen Vornamen- bzw. Nachnamenlisten entstanden sind. Mit diesen zuletzt genannten Prozeduren ist die Wahrscheinlichkeit der Vornamen- und Nachnamenlokalisierung größer geworden. Diese Prozeduren werden später näher betrachtet, aber hier wird zuerst deren Einsatz im System anhand von Beispielen erläutert.

Abbildung 11 Fälle der Matching 2 in Abbildung 10



❑

Die Abbildung 11 beschreibt den Ablauf, wenn das aktuelle Token als Vornamenkandidat betrachtet werden kann (entspricht Matching 2 in Abbildung 10).
Wie schon erwähnt wurde, wird ein Personenname nur angenommen, wenn er aus einem Vornamen und einem Nachnamen besteht. Das liegt daran, dass in einem Dokument ein Nachname immer nach dem eigenen Vornamen vorkommt. Und falls ein Nachname allein in einem Satz zu finden ist, dann ist er vorher sehr wahrscheinlich nach einem Vornamen oder einem Honorar aufgetreten. Deshalb werden die Tests immer auf zwei Token durchgeführt.
Der Fall, in dem das erste Token ein sicherer Vorname ist, wurde im letzten Abschnitt schon behandelt. Die Tests, die in diesem Fall auf dem nachfolgenden Token durchgeführt werden, werden nicht so streng durchgeführt, weil nach einem sicheren Vornamen sehr wahrscheinlich ein Nachname folgt (Streng meint hier, dass z.B. ein großgeschriebenes, nicht-deutsches Wort nach einem sicheren Vornamen direkt als Nachname gesehen wird, dagegen werden weitere Matching-Tests auf dieses Wort durchgeführt, falls es nach einem Vornamenkandidaten und nicht nach einem sicheren Vornamen vorkommt).

Der Fall, der in diesem Abschnitt behandelt wird, betrifft Token, die als Vornamenkandidat gesehen werden. Ein Vornamenkandidat ist ein großgeschriebenes Token, das nicht in der Vornamenliste vorkommt, aber es besteht den Ähnlichkeitstest, d.h. der Unterschied zu einem sicheren Vornamen ist sehr gering wie z.B. **Malik** und **Malek**. Malik ist ein Vorname, allerdings kommt er in der Vornamenliste nicht vor. Aber weil er Malek sehr ähnelt, das in der Vornamenliste zu finden ist, wird er angenommen und als Vornamenkandidat betrachtet.
Dieser Fall wird anders angegangen als der erste Fall, weil die Wahrscheinlichkeit hoch ist, dass der Vornamenkandidat tatsächlich kein richtiger Vorname ist. Daher werden die Tests strenger, die auf dem nachfolgenden Token durchgeführt werden.

Wenn also das aktuelle Token den Ähnlichkeitstest besteht wie im Fall **Brahim**, das **Braham** ähnelt, dann wird getestet, ob das nachfolgende Token ein Nachname sein kann. Eine dieser Möglichkeiten können auftreten:

1. Das nachfolgende Token ist ein Sonderzeichen oder es wird klein geschrieben. In diesem Fall sucht man nach dem nächsten möglichen Vornamenkandidat oder Honorar.

2. Das nachfolgende Token besteht den Existenztest, d.h. es kommt in der Nachnamenliste vor wie **Schmidt**; **Haustein** etc.

3. Das nachfolgende Token besteht den Ähnlichkeitstest, d.h. es ähnelt einem sicheren Nachnamen, der in der Nachnamenliste vorkommt.
   Ein Ähnlichkeitstest ist in einem der folgenden 3 Fällen positiv:

a. Das aktuelle Token ist ein Teilstring eines Nachnamens, der in der Nachnamenliste vorkommt. z.B. **Amin/Amine**; **Helmut/Helmuth** oder **Graf/Graff.**

b. Das aktuelle Token existiert in der Nachnamenliste, allerdings ohne z.b. ein anschließendes „s“. Dieser Fall kommt sehr oft vor z.B. **Adam/Adams**; **Bruno/Brunos**; **Watt/Watts**; **Hinrich/Hinrichs**; **Braham/Brahams** etc. Das gilt auch für andere Buchstaben.

c. Das aktuelle Token unterscheidet sich von einem Nachnamen, der in der Nachnamenliste existiert nur mit einer Buchstabe im Wort. z.B. **Schmitz/Schmitt**; **Deitmer/Dettmer**; **Redmann/Reimann** etc.

4. Das nachfolgende Token besteht den Matchingstest, d.h. der Unterschied zu einem Nachnamen aus der Nachnamenliste ist gering, so dass dieses Token als ein Nachnamenkandidat betrachtet werden kann. Bei dem Matchingstest werden die Anzahl der Buchstaben gezählt, die an den gleichen Stelle vorkommen. Der Unterschied darf dabei eine bestimmte Grenze nicht überschreiten.
   Der Matchingstest ist also eine andere Stufe des Ähnlichkeitstests und erlaubt, Tokens als Nachnamenkandidat zu akzeptieren, obwohl sie sich an mehr als eine Stelle von einem Nachnamen aus der Nachnamenliste unterscheiden.z.B. **Hertzsch/Hertzog**; **Ayari/Ayala**; **Selcuk/Seniuk**; **Rücker/Richer** etc.

Kapitel 4

Einsatz des Namenerkennungs-werkzeuges im Projekt MetaAkad

4.1 Über MetaAkad

Über das Internet werden von Hochschulen, Bildungseinrichtungen und Verlagen im In- und Ausland Lehr- und Lernmaterialien zur Verfügung gestellt. Sie sind entweder kostenfrei für jeden Internetbenutzer zugänglich, sind teilweise aber auch kostenpflichtig. Diese Angebote werden jedoch meist nur von einem begrenzten Personenkreis genutzt. Der Bekanntheitsgrad dieser Produkte beschränkt sich z.B. bei Hochschulangeboten zumeist auf einzelne Lehrstühle der anbietenden Hochschule. Die fehlende Erschließung dieser elektronischen Dokumente macht das Auffinden geeigneter Produkte für Lehrzwecke schwierig.

Das Projekt META-AKAD hat das Ziel, dem Nutzer einen einheitlichen Zugang zu einem umfassenden Angebot von im Internet verfügbaren Lehr- und Lernmaterial zu verschaffen. Zu den Nutzern des Angebots zählen dabei sowohl Lehrende, die das Material im Rahmen ihrer Lehrveranstaltungen einsetzen können, als auch Lernende, für deren Selbst-studium das Material zur Verfügung gestellt wird. Das Projekt schafft einen innovativen Pilot-Service durch den Einsatz und die Entwicklung geeigneter technischer Mittel und den Aufbau funktionierender Organisationsstrukturen, um den effizienten Einsatz der elektronischen Materialien für Lehrzwecke und einen schnellen, möglichst umfassenden und bedarfsgerechten Zugriff auf die im Internet verfügbaren Dokumente zu ermöglichen. Dieser neue Nutzer-Service wird exemplarisch für die Fächer Mathematik, Physik, Germanistik, Psychologie und Biologie aufgebaut, soll aber so ausgelegt werden, dass er auf alle Disziplinen erweitert werden kann.

Zur Realisierung dieses Dienstes soll Online-Lehr- bzw. Lernmaterial auf kooperativer Basis gesammelt, durch standardisierte und materialspezifische Meta-Daten erschlossen, nach inhaltlichen und didaktischen Kriterien bewertet und in einer einheitlichen Nutzeroberfläche zugänglich gemacht werden. Dies beinhaltet die Erschließung mit Hilfe von Verfahren für die (semi-) automatische Klassifikation und Vergabe von Metadaten. Zur Verbesserung der Qualität der Erschließung werden Verfahren zur intellektuellen Nachbearbeitung integriert. Das Sammeln der Metadaten zu den Dokumenten soll sowohl automatisch erfolgen als auch auf kooperativer Basis manuell nachgearbeitet und ergänzt werden. Schließlich sollen die Materialien von Fachleuten bewertet werden. Durch die Integration einer solchen Qualitätskontrolle wird die Verwertbarkeit der Materialien für Lehr- und Lernzwecke deutlich erhöht und damit ein wesentlicher Mehrwert des Dienstes geschaffen, da der Nutzer in META-AKAD nicht nur recherchieren kann, welche Dokumente verfügbar sind, sondern auch detaillierte Informationen über deren Qualität und Nutzbarkeit erhält. Der Zugang zu den Materialien wird durch ein benutzerfreundliches Interface ermöglicht, das auf der Grundlage von Nutzerstudien evaluiert und laufend optimiert wird.[1]

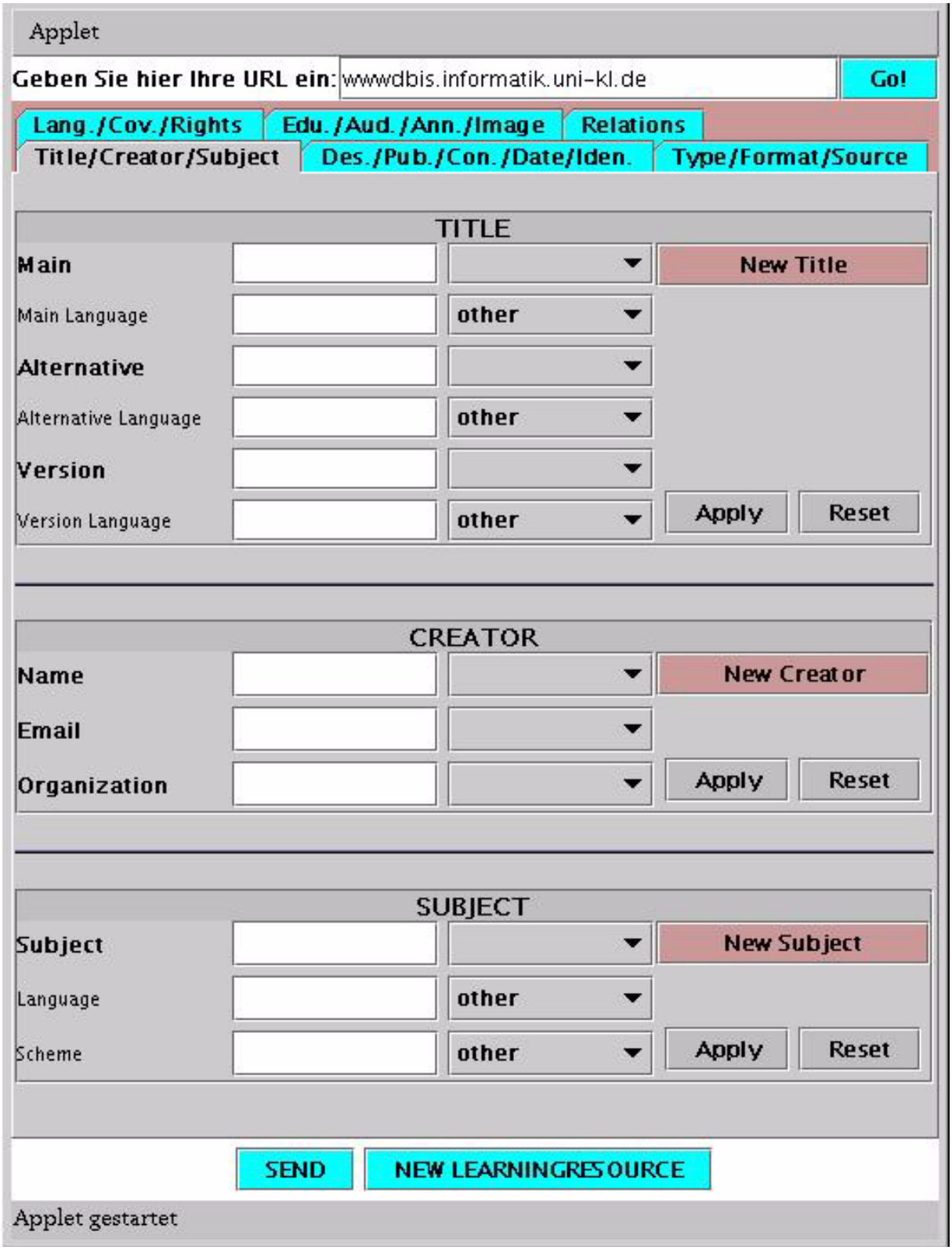
4.2 Überblick

In diesem Abschnitt wird beschrieben, wie das Namenerkennungswerkzeug im Rahmen des Projekts MetaAkad eingesetzt wird. Im Projekt MetaAkad wird ein semi-automatisches Erschließungswerkzeug erstellt, um bei der Erschließung sinnvolle Vorschläge für die Metadaten, die in einem semi-strukturierten Dokument (hauptsächlich HTML) gesucht werden, machen zu können.
Mit Hilfe des GUI-Werkzeugs (Abbildung 12), das im Projekt MetaAkad eingesetzt werden soll, werden Metadaten aus semi-strukturierten Dokumenten extrahiert, die im Internet verfügbar sind wie z.B. Titel, Creator, Publisher, Type, Format etc., aber es werden dabei nicht nur die Metadateneinträge(Attribute) auf der Oberfläche ausgefüllt, sondern auch sinnvolle Vorschläge zu diesen Attributen gegeben. Verschiedene Algorithmen werden bei der Erschließung integriert, die als Ziel haben, dem Benutzer konkrete und sinnvolle Vorschläge zu geben. Dies macht aus diesem Tool ein semi-automatisches Erschließungstool. Nachdem man die Attribute des Dokuments bestimmt hat, wird aus diesen Attributen ein XML-Dokument erzeugt, das zum XML-Server weitergeschickt wird. Das Namenerkennungswerkzeug kann dabei helfen, konkrete Vorschläge für Autoren, Herausgeber u.a. zu geben. Dabei wird das bereits im vorherigen Kapitel vorgestellte Werkzeug eingesetzt, um konkrete Vorschläge für die jeweilige Attribute zu geben. Das Werkzeug gibt also eine Liste mit den möglichen Personennamen zurück, die im HTML-Dokument zu finden sind. Der Benutzer kann dann aus dieser Liste einen Namen wählen, der beispielsweise das Attribut Creator im XML-Dokument darstellen wird.

In der folgenden Abbildung (Abbildung 12) ist ein Beispiel des Applets, das im Projekt MetaAkad eingesetzt wird mit einem Überblick auf einigen Attribute, die aus dem HTML-Dokument extrahiert werden sollen.

1. https://rzblx1.uni-regensburg.de/metaakad

Abbildung 12 Das MetaAkad Applet



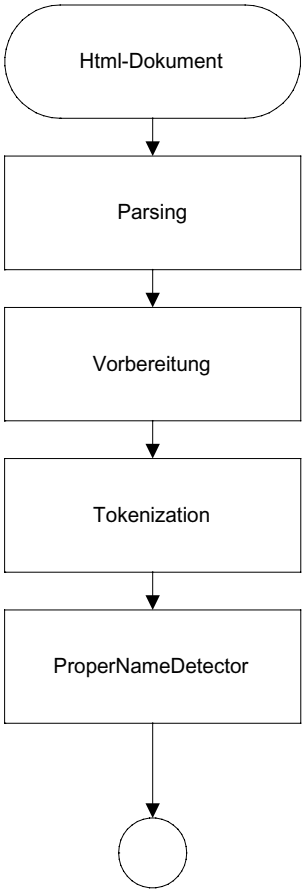
❑

4.3 Einsatz des Namenerkennungswerkzeuges

4.3.1 Motivation

Das Namenerkennungswerkzeug extrahiert aus einer gegebenen Zeichenkette (String) Tokens und wendet dann Heuristiken an, um aus den extrahierten Tokens solche zu bestimmen, die Teile eines Personennamens sein können. Daher muss vor dem Einsatz des Werkzeuges eine Zeichenkette aus dem HTML-Dokument erzeugt werden. Das HTML-Dokument muss daher zuerst geparst werden. Dann wird die Ausgabe des Parsens weiter verfeinert, indem man die nicht relevanten Informationen vernachlässigt und Punkte hinzufügt, wenn es bei der Tokenization helfen kann. Die Schritte wie man aus einem HTML-Dokument Personennamen finden kann, werden in Abbildung 13 skizziert. Im Folgenden werden die Regeln, die bei der „Vorbereitung“ eingesetzt werden näher betrachtet..

Abbildung 13 Prozess zur Personennamenerkennung in einem HTML-Dokument



❑

4.3.2 Beschreibung

Das Namenerkennungswerkzeug wird also eingesetzt, um konkrete Vorschläge für das Attribut Creator zu geben. Es braucht eine Zeichenkette als Eingabe. Diese Zeichenkette wird in Token transformiert und dann werden verschiedene Heuristiken auf diese Tokenmenge angesetzt, um Personennamen zu extrahieren. Ein HTML-Dokument besteht aus einer Menge von TAGs wie Title, Meta, Script, Style, Table, Td, H1, A etc. und eine Zeichenkette muss aus dem Inhalt dieser TAGs gewonnen werden (Beispiel 8). Es ist hier aber hilfreich den Text (die Zeichenkette, die aus dem Html-Dokument konstruiert wird), so aufzubauen, dass er einerseits minimal und andererseits gut strukturiert ist, d.h. Punkte und andere Sonderzeichen, die bei der Tokenization nötig sind, sind im Text an der richtigen Stellen zu finden.

Beispiel 8 **Text aus einem HTML-Dokument erzeugen**

Das folgende HTML-Code:

```
<HTML>
<HEAD>
<TITLE>Amin CHATTI Homepage - TUNISIA - Uni kaiserslautern -
tunesien - tunisie </TITLE>
</HEAD>
</HTML>
```

wird zu einer Zeichenkette umgewandelt mit dem Inhalt:

```
Amin CHATTI Homepage - TUNISIA - Uni kaiserslautern -
tunesien - tunisie
```

❑

- Text minimalisieren

  Wenn der Text, der aus einem HTML-Dokument gewonnen wird lang ist, ist der Zeitaufwand beim Einsatz des Namenerkennungswerkzeuges hoch, da jedes Wort aus dem Text getestet wird, ob es ein Teil eines Personennamens sein kann. Darum ist die Vernachlässigung der nicht relevanten Informationen von Bedeutung. Zu der nicht relevanten Informationen gehört z.B. der Inhalt einiger Html-TAGs. Das ist der Fall bei einem Script oder auch Style-TAG (Beispiel 9). Generell kommt viel Text vor, aber dieser Text enthält keine Personennamen. So wird der gewonnene Text kürzer und der Zeitaufwand niedriger, wenn man diese störenden Teile vernachlässigt.

Beispiel 9 **Nicht relevante Teile in einem Html-Dokument**

```
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function NeuFenster(name)
{
window.open(name,"_blank","width=500,height=500,scrollbars,res
izable=yes,location,menubar");
}
</SCRIPT>
</HEAD>
<BODY background=" backg.gif" bgcolor="#FFFFFF" link="#FF9966"
vlink="#FF9900" alink="#006666">
<STYLE type="text/css">
.gross   {color:   white;font:bold   18pt   Times;text-decora-
tion:none;}
.klein {color: red;font:bold 12pt Times;}
</STYLE>
</BODY>
</HTML>
```

❑

- Text strukturieren

  Wie schon erwähnt wurde, gehört zur *Vorbereitung* (Abbildung 13), eine Zeichenkette aus dem Inhalt der in einem HTML-Dokument sich befindenden TAGs zu bilden. Allerdings ist ein Text ohne Sonderzeichen schwerer zu behandeln gegenüber einem Text, der Punkte und andere Sonderzeichen enthält. Das liegt daran, dass es schwierig wird Teile eines Personennamens zu finden, die eigentlich zusammen gehören, in einem Text, der keine Sonderzeichen enthält. Das Namenerkennungswerkzeug enthält Funktionen, die den Einsatz eines Punktes genau beachten und entscheiden können, ob in einem Text ein Punkt als eine Abschließung eines Satzes (Punkt) oder aber nach einer Abkürzung wie z.B. Dr., Prof. (Pünktchen) vorkommt. Den Vorteil kann man nutzen, indem man Punkte zum initialen Text hinzufügt, die dabei helfen können, richtige Personennamen zu finden. Die Idee ist also einen Punkt hinzufügen, wenn ein Text in einem TAG zu Ende ist (Beispiel 10). Dies macht die Struktur des Textes übersichtlicher und das Ergebnis der Personennamenserkennung besser.

Beispiel 10 Punkte zum initialen Text hinzufügen

Es wird aus dem folgenden HTML-Code:

```
<TR><TD><A href="/staff/Haerder/" class="subtopic">
Prof. Dr. Theo Härder</A> </TD></TR>
<TR><TD><A HREF="/staff/Ritter/" class="subtopic">Dr. Norbert
Ritter</A></TD></TR>
<TR><TD><A href="/staff/Flehmig/" class="subtopic">Marcus Fleh-
mig</A></TD></TR>
```

der folgende Text konstruiert:

```
Prof. Dr. Theo Härder. Dr. Norbert Ritter. Marcus Flehmig.
```

❑

Mit diesem Trick ist man sicher, dass wenn einmal ein Personenname detektiert wird, er sehr wahrscheinlich wirklich ein Name ist, d.h. man bekommt nie einen Personennamen, der aus 2 Teilen besteht, die eigentlich nicht zusammen gehören wie z.B. Ritter Marcus, im Fall, dass kein Punkt zwischen Ritter und Marcus (Beispiel 10) gibt.

Es ergibt sich aber das Problem dass, wenn ein Vorname und ein Nachname in einer Tabelle vorkommen, d.h. als ein Text im TD-TAG wie es das Beispiel 11 zeigt.

Beispiel 11 Vorname und Nachname treten in einer Tabelle auf

```
<TABLE border=0 width=100%>
<TR>
<TD>Prof. Dr.
<TD>Reinhard
<TD>Gotzhein
</TR>
<TR>
<TD>Dipl.-Inform.
<TD>Alexander
<TD>Geraldy
</TR>
</TABLE>
```

❑

Wenn man das Beispiel eingeht, wie es oben erklärt ist, so bekommt man den folgenden Text:

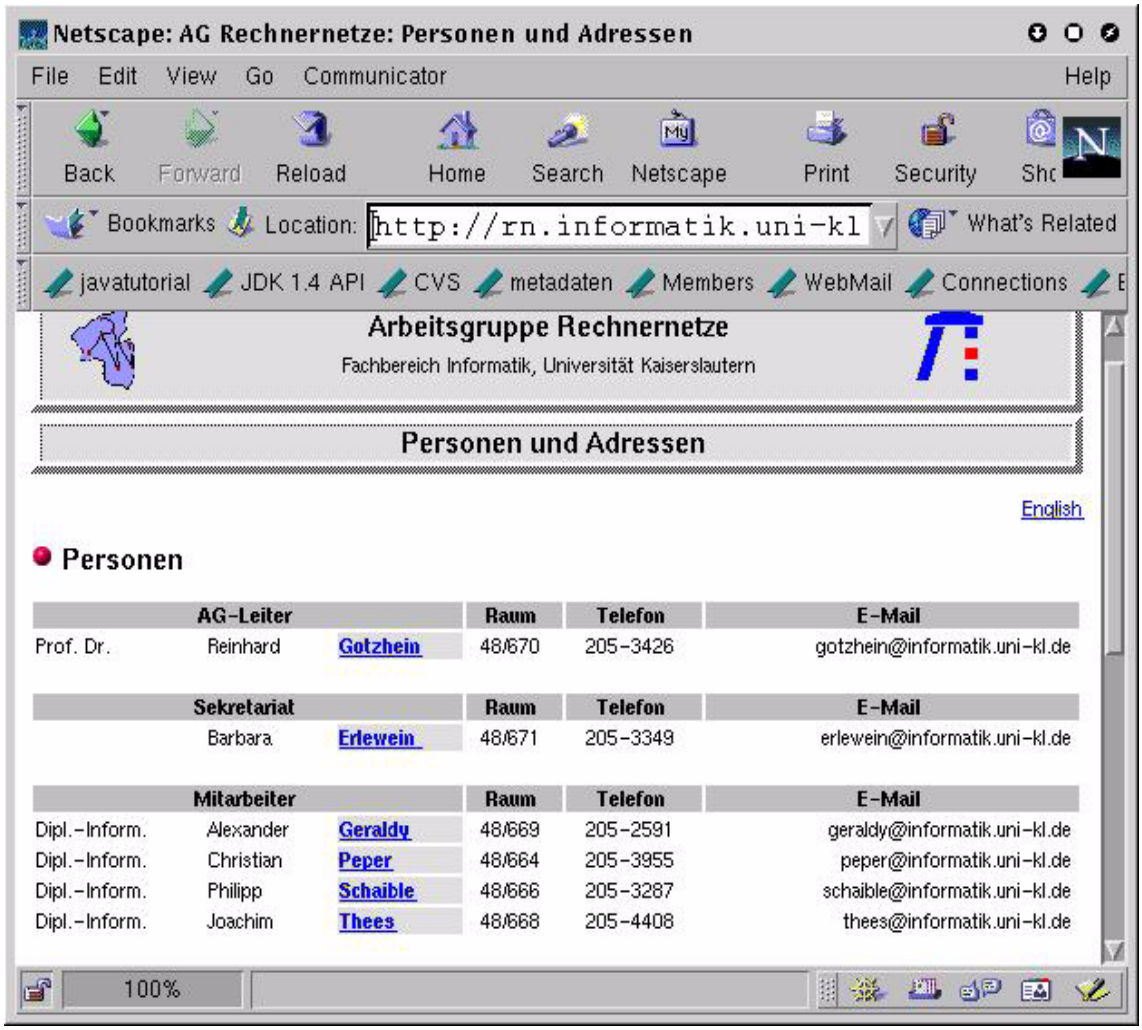
```
Prof. Dr.. Reinhard. Gotzhein. Dipl.-Inform.. Alexander. Geraldy.
```

Die hinzugefügten Punkte machen es unmöglich, Prof. Dr. Reinhard Gotzhein als Personenname zu detektiern. Die Lösung dafür, ist, dass wenn ein Text in einem TD-TAG ist, dann wird kein Punkt hinzugefügt. Somit bekommt man aus dem obigen Beispiel den folgenden Text:

```
Prof. Dr. Reinhard Gotzhein. Dipl.-Inform. Alexander Geraldy
```

Und so ist es möglich Prof. Dr. Reinhard Gotzhein als Personenname zu finden und Namen in einem HTML-Dokument mit Tabellen zu lokalisieren, wie beim HTML-Dokument in Abbildung 14.

Abbildung 14 Ein Html-Dokument mit Tabellen



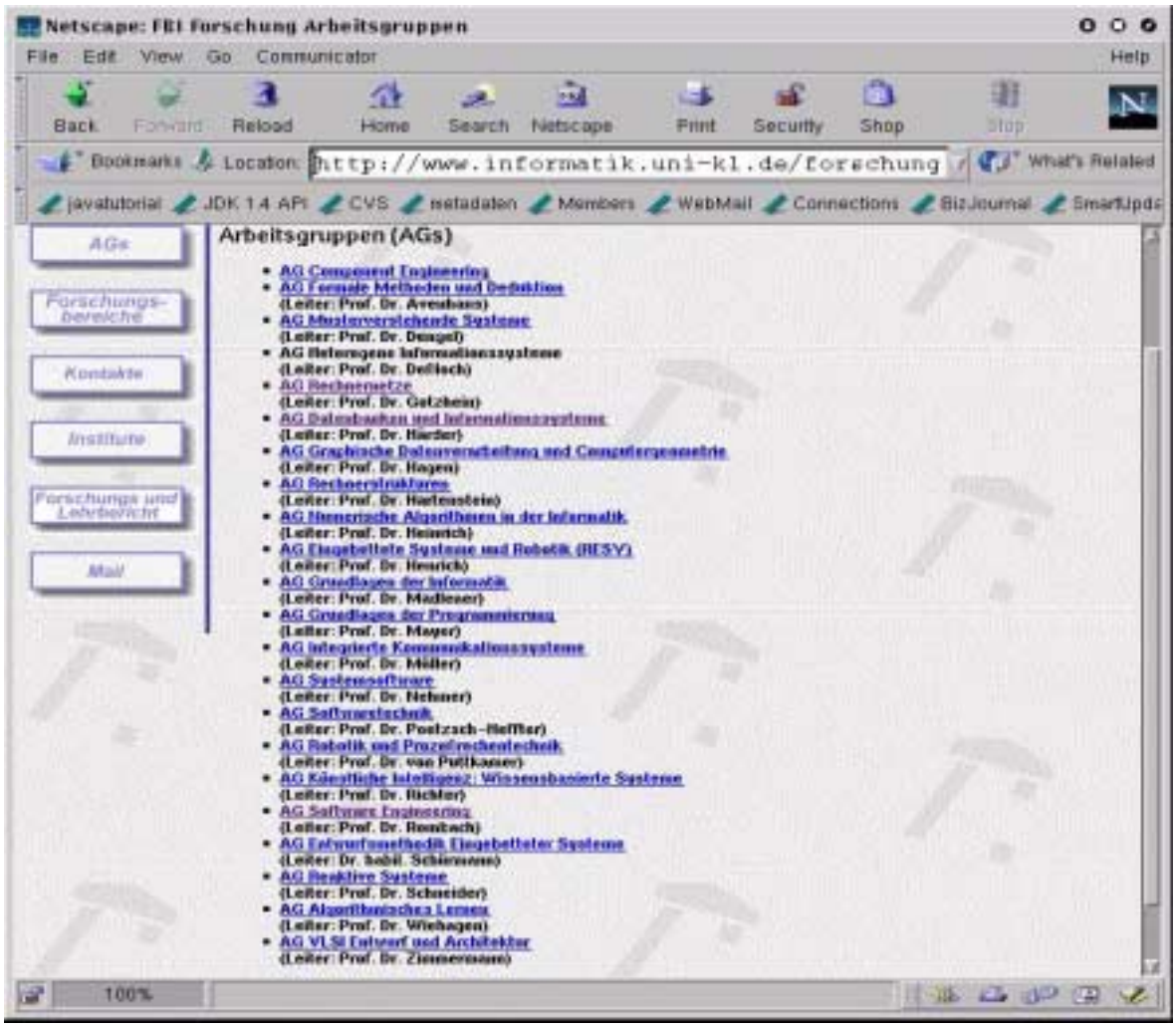
❑

Das waren die Maßnahmen, die berücksichtigt werden, damit man aus einem HTML-Dokument einen minimalen und gut strukturierten Text erhalten kann, der eine Eingabe für den ProperNameDetector sein soll. Wenn dieser Text einerseits minimal ist (nach der Nachlässigkeit der nicht relevanten Teile des Html-Dokuments) und andererseits gut strukturiert wird (nach dem Einfügen von Punkten an der richtigen Stellen), dann wird der Zeitaufwand geringer und die Systemleistung höher.

4.3.3 Beispiele

1. Beipiel aus einem Html-Dokument mit Honorar (Abbildung 15): In diesem Html-Dokument befinden sich Personennamen, die mit Honoraren wie Prof. Dr. eingeführt werden.

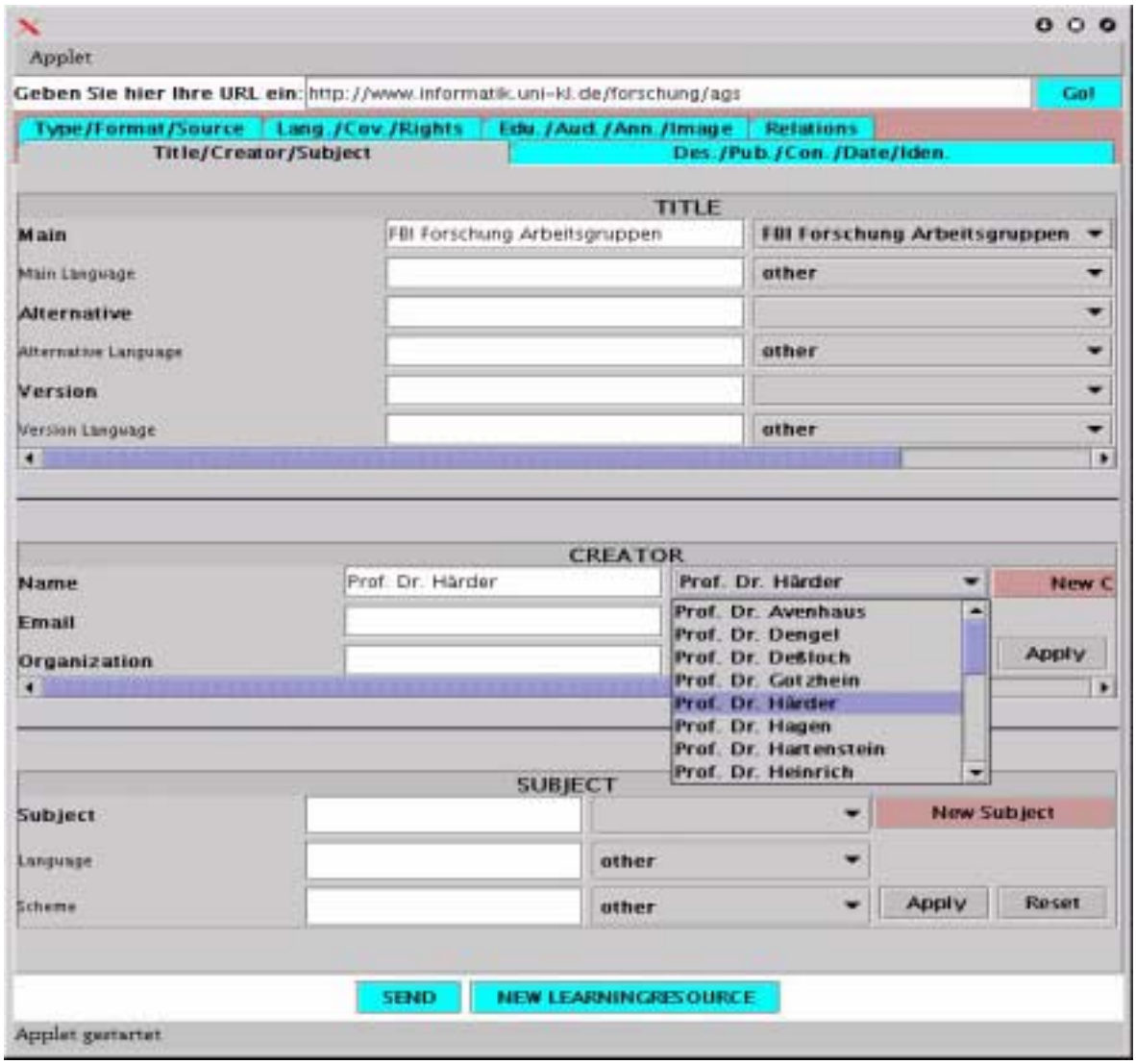
Abbildung 15 Html-Dokument mit Honorar



❑

Das Namenerkennungswerkzeug angewendet auf diesem Html-Dokument ergibt die Ausgabe in Abbildung 16. Für das Attribut CREATOR werden Vorschläge gegeben und der Benutzer der GUI-Fläche kann dann bei der Erschließung einen Eintrag aus der Liste wählen, der dieses Attribut im XML-Dokument darstellen wird.

Abbildung 16 Vorschläge für das Attribut CREATOR



❑

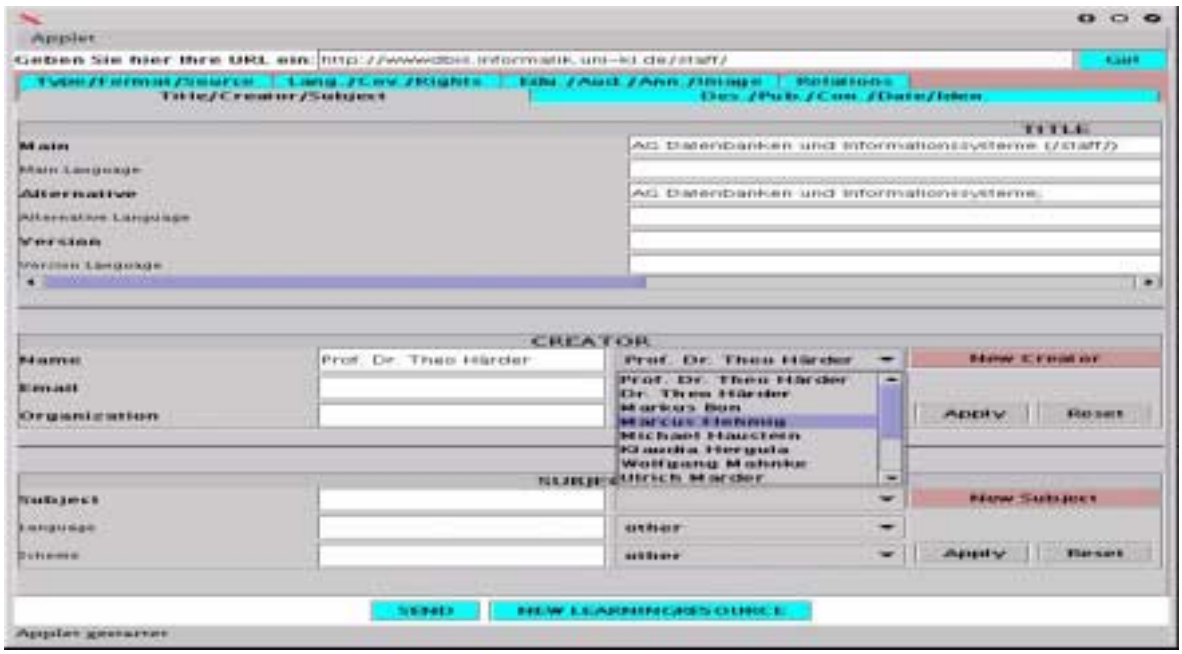
2. Beipiel aus einem HTML-Dokument mit Personennamen, die nicht mit Honoraren eingeführt werden (Abbildung 17). In diesem Fall werden mit Hilfe der Vor- und Nachnamenwörterbücher verschiedene Heuristiken angewendet.

Abbildung 17 Dokument mit Personennamen ohne Honorare

❑

Das Namenerkennungswerkzeug angewendet auf diesem HTML-Dokument ergibt die Ausgabe in Abbildung 18.

Abbildung 18 Vorschläge für das Attribut CREATOR



❑

Kapitel 5 Zusammenfassung, Bewertung und Ausblick

5.1 Zusammenfassung und Bewertung

Wir haben in dieser Arbeit das Namenerkennungswerkzeug vorgestellt, das im Rahmen des Projekts META-AKAD eingesetzt werden soll. Dieses Werkzeug folgt den Knowledge-Engineering-Approach und hat zum Ziel Personennamen sowohl aus einem englischen, als auch aus einem deutschen Text zu extrahieren. Dabei wurde zuerst nach einem kurzen Überblick auf das MetaAkad-Projekt die Problematik vorgestellt. Dann wurden die bekanntesten Namenerkennungssysteme und die dabei verwendeten Ansätze näher betrachtet. Danach wurden das Konzept des implementierten Namenerkennungswerkzeuges und dessen Komponenten beschrieben und schließlich wird der Einsatz des Werkzeuges im MetaAkad-Projekt erläutert.
Die Bewertung wird hier qualitativ und nicht quantitativ durchgeführt d.h. ohne “Recall” und “Precision“ Werte zu nennem. Das in dieser Arbeit vorgestellte Namenerkennungswerkzeug kann im Gegensatz zu anderen Werkzeugen, die im Bereich Information-Extraction entwickelt wurden, Personennamen aus Texten im Deutschen extrahieren. Mit anderen Werkzeugen stößt man dabei auf vielerlei Probleme, weil sie so entwickelt sind, dass sie mehr die Eigenschaften der englischen Sprache berücksichtigen. Da aber ProperNameDetector ein Knowledge-Engineering-Approach ist, der auf Wörterbüchern basiert, ist die Vollständigkeit solcher Hilfsmitteln von großer Bedeutung. Zudem scheitert ProperNameDetector bei Personennamen mit mehr als drei Einträgen, weil in dieser Arbeit angenommen wird, dass ein Personenname höchstens aus drei Wörtern besteht. Dies ist auch der Fall bei vielen nicht europäischen bzw. amerikanischen Namen, da sie nicht in der benutzten Wörterbüchern existieren und auch nicht von den Matching-Algorithmen abgedeckt werden.

5.2 Ausblick

Im Projekt MetaAkad wird ein semi-automatisches Erschließungswerkzeug erstellt, um bei der Erschließung sinnvolle Vorschläge für die Metadaten, die in einem semi-strukturierten Dokument gesucht werden, machen zu können.

Das Namenerkennungswerkzeug kann dabei helfen, konkrete Vorschläge für Creator-Namen zu geben. Man kann aber die Funktionalität dieses Werkzeuges erweitern, um auch Vorschläge für andere Attribute zu erhalten, die nicht nur Namen enthalten. Die Frage aber ist, ob der Einsatz unter der Beobachtung, dass der Zeitaufwand bei der Analyse eines Dokuments nicht unerheblich ist, sinnvoll ist. Das Namenerkennungswerkzeug könnte auch das automatische Sammeln unterstützen. Insbesondere deshalb, da dort das Laufzeitverhalten nicht so kritisch einzuschätzen ist wie im Bereich der Endbenutzerschnittstelle. Dies wird eine Evaluation durch die Benutzer zeigen.

Kapitel 6 Literaturverzeichnis

[STE2001] Ilona Steiner
Warum "Named Entities" für die Chunk-Analyse wichtig sind
Giessen, 2001

[AI1999] Douglas E. Appelt, David J. Israel
Introduction to Information Extraction Technology
SRI International
Stockholm, Sweden, 1999

[MUC 7] Seventh Message Understanding System Evaluation and Message Understanding Conference
Sponsored by:
The Human Language Systems Tipster Text Programm of theDefense Advanced Research Projects Agency, Information Technology Office (DARPA/ITO)
Washington, D.C. area, April 1998

[MUC 6] The sixth in a series of Message Understanding Conferences
November 1995

[BOR1999] Andrew Borthwick
A Maximum Entropy Approach to Named Entity Recognition
Computer Science Department, New York University
September 1999

[MAG1994] David M. Magerman
Natural Language Parsing as Statistical Pattern Recognition
PhD Thesis, Standford University
1994

[BSW1997] Daniel M. Bikel, Richard Schwartz, Ralph M. Weischedel
An Algorithm that Learns What's in a Name
BBN Systems&Technologies
Cambridge, 1997

[BRM1998] William J. Black, Fabio Rinaldi and David Mowatt
FACILE: Description of the NE System used for MUC 7
Department of Language Engineering
Manchester, United Kingdom 1998

[KH1998] George R. Krupka and Kevin Hausman
IsoQuest: Description of the NetOwl(tm) Extractor System as used in MUC 7
1998

[BSAG1998] A. Borthwick, J.Sterling, E.Agichtein, R. Grishman
NYU: Description of the MENE Named Entity System as used in MUC 7
New York University
1998

[FASTUS] Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel,
Megumi Kameyama, Mark Stickel and Mabry Tyson
FASTUS: Extracting Information from Natural Language Texts
SRI International, California

[RW1997] Yael Ravin and Nina Wacholder
Extracting Names from Natural-Language Text
IBM Research, 1997

[PEN] Breck Baldwin, Jeff Reynar, Mike Collins, Jason Eisner, Adwait Ratnaparkhi, Joseph Rosenzweig and Anoop Sarkar Srinivas
Description of the University of Pennsylvania System used for MUC-6
University of Pennsylvania