Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Extraktion von semantischen Relationen aus natürlichsprachlichem Text mit Hilfe von maschinellem Lernen

Diplomarbeit

Leipzig, September 2002

vorgelegt von:

Christian Biemann
geb. am 3. März 1977
Studiengang Diplominformatik

-

Zusammenfassung

Inhalt der vorliegenden Arbeit ist die Entwicklung eines Lernverfahrens, das aus großen Textkorpora semantische Relationen automatisch extrahiert.
Den Kern des Verfahrens bildet die Iteration von Suchschritt und Verifikationsschritt, in denen in gesuchter Relation stehende Wörter gefunden und überprüft werden. Auf diese Weise ist es möglich, mit wenigen bekannten Wörtern eine große Anzahl in derselben Relation stehende Wörter zu gewinnen. So können mit wenig Aufwand große Listen von Wörtern erstellt werden, die in einem semantischen Zusammenhang stehen.

Nach der Skizzierung des Algorithmus werden theoretische Vorhersagen bezüglich der für das Verfahren geeigneten Relationen getroffen, sowie der Ablauf modelliert.
Einige mit einer Implementierung des Verfahrens erzielten Ergebnisse werden für verschiedene semantische Relationen vorgestellt, evaluiert und diskutiert, desweiteren werden Ausblicke und Verbesserungsmöglichkeiten angegeben.

Schließlich wird eine Anwendung des Verfahrens vorgestellt, die im Rahmen des Projekt Deutscher Wortschatz in Zeitungsartikeln Personennamen mit zugehörigen Berufsbezeichnungen markiert.

Inhalt

-

-

1 Einleitung

In den nunmehr über 60 Jahren, die seit der Erfindung der elektronischen Rechenmaschine vergangen sind, wurde die Welt durch die Möglichkeit, immer schneller und preiswerter immer größere Berechnungen durchzuführen, nachhaltig verändert. Schon früh kamen in der Science-Fiction-Literatur sprechende Computer vor – Automaten, die verstehen, was ein Mensch oder ein anderes künstliches Wesen ihnen mittels natürlicher Sprache mitteilt, und die ihrerseits darauf in einer Weise antworten, als hätten sie begriffen, worum es ging. Der Mythos des allwissenden und alles könnenden Computers beherrscht nach wie vor die Köpfe der Menschen, auch wenn die Vision eines elektronischen Gesprächspartners immer noch wie Zukunftsmusik klingt.

Weil Berechnungen in Schritten erfolgen, muß der menschliche Geist zunächst alle Schritte der Berechnung verstanden haben, bevor er sie in eine Maschine implementieren kann. Dabei hat sich die Implementierung von menschlicher Sprache, dem Charakteristikum des Menschen, in dem er sich von allen anderen Lebewesen auf dieser Erde unterscheidet, als besonders schwierig erwiesen. Vorliegende Arbeit soll auf dem Weg dorthin ihren bescheidenen Beitrag leisten.

Ziel der Arbeit war es, semantische Relationen zu erlernen, um diese in natürlichsprachlichen Texten erkennen zu können. Es wurde ein Verfahren entwickelt, das mit Hilfe einer großen Textsammlung Wörter lernt, die in einer bestimmten Relation zu anderen Wörtern stehen. Hierbei soll sich durch ein kleines vorausgesetztes Grundwissen hoher Datenertrag ernten lassen.

Die Arbeit ist folgendermaßen aufgebaut: Im ersten Kapitel wird eine Einordnung der Arbeit in die beteiligten wissenschaftlichen Gebiete vorgenommen. Es folgt eine Übersicht über die verwendete

-

Textsammlung, sowie über verschiedene semantische Relationen, und wie diese beschaffen sein müssen, damit sie für das Verfahren geeignet sind.

Im zweiten Kapitel wird der dem Verfahren zugrundeliegende Algorithmus erklärt. Nach einigen Vorüberlegungen zur theoretischen Beschaffenheit der gesuchten Relationen wird der Algorithmus im Detail dargelegt. Es folgen Betrachtungen zum erwarteten Verhalten, sowie zu Voraussetzungen, die für den Erfolg des Verfahrens notwendig sind.
Die durch eine Implementierung des Verfahrens erzielten Ergebnisse werden in Kapitel drei vorgestellt. Hauptgegenstand der Untersuchungen war die semantische Relation, die zwischen Vor- und Nachnamen besteht. Das Verhalten der Implementierung wurde aber auch an anderen Relationen untersucht. Das vierte Kapitel enthält Ausblicke zu weiteren Relationen, außerdem werden Verbesserungen und andere Anwendungsgebiete diskutiert. Eine Anwendung des Verfahrens in leicht modifizierter Form findet sich im fünften Kapitel. Ziel dieser Anwendung war es, Personennamen in Texten zu markieren und die Berufe der Personen der aktuellen Zeit zu finden. Im sechsten Kapitel finden sich abschließende Bemerkungen.

1.1 Computerlinguistik und Korpuslinguistik

Die Wissenschaft der *Computerlinguistik* ist zwischen der *Allgemeinen Sprachwissenschaft (Linguistik)* und der *Informatik* angesiedelt und befaßt sich mit Sprachanalyse und Sprachproduktion von natürlicher Sprache. Das Endziel der Computerlinguistik besteht seit ihrem Aufkommen in den 1950er Jahren, einen "sprechenden Computer" hervorzubringen, der menschliche Äußerungen verstehen und sinnvolle Antworten äußern kann. Dazu gibt es zwei Herangehensweisen, zum einen die Simulation des menschlichen Gehirns, zum anderen die Emulation von Sprache im Computer ohne Kenntnis der Funktionsweise des Gehirns.

-

Da die kognitiven Fähigkeiten des Menschen noch nicht in ausreichendem Maße bekannt sind, wird im wesentlichen der zweite Ansatz verfolgt. Die Aufgabenstellung jedoch ist nicht minder schwer, wenn man bedenkt, wie viele Faktoren beim Sprachverstehen und der Sprachproduktion zusammenkommen. Zum einen kann der Mensch in jeder Situation aus einer sehr großen Menge von Ausdrücken wählen, die sinnvoll geäußert werden können, zum anderen ist der Mensch in der Lage, aus den Äußerungen und Texten anderer linguistisches und semantisches Wissen zu extrahieren und mit Hilfe dieses Inputs nicht nur die Sprache an sich, sondern auch Wissen über die Welt zu lernen. Dies ist nur möglich, wenn gewisse Fähigkeiten zum Spracherwerb vorhanden sind – Chomskys *Universalgrammatik* (UG) (z.B. in [Malmkjaer 2002]), die angeborene Fähigkeit von Kindern, Sprache zu erwerben, sei als ein Beispiel genannt. Desweiteren ist ein gewisses Maß an nichtsprachlichem Weltwissen zum Verstehen von Äußerungen erforderlich.

Um ein System zu konstruieren, das Sprachverarbeitung auf allgemeinem Level realisiert, müssen wir verstehen, wie Sprache funktioniert, ferner muß das System eine Repräsentation der Welt beinhalten und es muß über eine Wissensbasis verfügen, auf der es auch Schlüsse ziehen kann.

Natürlich lassen sich diese Ziele nicht direkt erreichen und müssen in kleinere Teilprobleme untergliedert werden. Von der linguistischen Seite betrachtet unterscheidet man die Gebiete *Phonetik, Phonologie, Lexikon, Morphologie, Syntax, Semantik* und *Pragmatik*, mit der Hoffnung, zu einer linguistischen Theorie zu gelangen, die alle Sprachphänomene erklärt und so weit algorithmisierbar ist, daß eine Implementierung möglich ist.

Die *Automatische Sprachverarbeitung* hingegen befaßt sich mit Methoden, die Wissen aus Texten extrahieren, ohne die Texte im Gesamten zu verstehen, z.B. im *Information Retrieval* und im Bereich der *Dokumentenklassifikation*, von der Computerlinguistik werden Instrumente zur Validierung von linguistischen Theorien zur Verfügung gestellt.

-

Die *Korpuslinguistik* (von lat. corpus = Körper) als Teilbereich der Sprachwissenschaft hat ihre Ursprünge schon in den letzten Jahrzehnten des 19. Jahrhunderts, ihr Untersuchungsgegenstand stellen große Textsammlungen (Korpora) dar. Sie verwendet statistische Methoden über quantitativ-linguistische Analysen. Die Korpuslinguistik kann in die sogenannte *frühe Korpuslinguistik* (vor 1950) und die *heutige Korpuslinguistik* unterteilt werden. Bereits 1879 benutzte Kading ein Korpus von 11 Millionen deutschen Wörtern für statistische Untersuchungen der Buchstabenhäufigkeit und -abfolge. In der Folgezeit wurden meist Korpora von Kindesäußerungen benutzt, um den Spracherwerb empirisch zu untersuchen. In den fünfziger Jahren des 20. Jahrhunderts vollzog sich durch Noam Chomsky ein Richtungswechsel in der Linguistik, von der Empirie weg zum linguistischen Rationalismus. Dieser versucht, Theorien zu entwickeln, die den menschlichen Prozeß bei der Sprachverarbeitung modellieren. Die Arbeit im Bereich Korpuslinguistik geriet in den Hintergrund, da sich eine endliche, wenn auch große Sammlung von Sätzen nicht mit Chomskys Paradigma der endlichen Produktionsregeln, welche unendlich viele Sätze generieren können, vereinbaren ließ. Chomsky verließ sich auf die Intuition seines *idealen Sprechers* [Malmkjaer 2002] anstatt auf zwangsläufig unvollständige Textsammlungen. Ein weiterer Kritikpunkt an Korpora war deren Größe, die für manuelle Bearbeitung ungeeignet schien.

Erst mit dem *Brown Corpus of American English* [Francis 1964], das erste maschinenlesbare linguistische Korpus, reetablierte sich die Korpus-linguistik und stellte große Textsammlungen als Material für verschiedenste Forschungen zur Verfügung. Seit den achtziger Jahren des 20. Jahrhunderts, als Rechenkapazität und Speicherplatz in größerem Maße als je zuvor verfügbar wurden, stellt die Korpuslinguistik ein wichtiges Forschungsgebiet im Bereich der Computerlinguistik und für viele linguistischen Teilgebiete dar.

-

Ein Korpus im heutigen Sinne ist eine maschinenlesbare Textsammlung, die durch folgende Eigenschaften charakterisiert ist ([McEnery 1996], Kapitel 2):

- *Repräsentativität*, d.h. die Worthäufigkeiten im Korpus sollten mit den tatsächlichen übereinstimmen
- *endliche, feste Größe*
- *Maschinenlesbarkeit*
- *weite Verfügbarkeit*, d.h. das Korpus sollte auch anderen Forschungsgruppen zur Verfügung stehen

1.2 Projekt Deutscher Wortschatz

Seit 1994 wurde an der Universität Leipzig unter der Leitung von Uwe Quasthoff eine Infrastruktur erstellt, um elektronische Korpora zu analysieren und mit ihnen arbeiten zu können.

Das inzwischen angesammelte Datenmaterial macht das *Wortschatz*-Korpus zu einem der größten Korpora in Deutsch, Englisch und Italienisch (je über 9 Mio. Sätze), desweiteren existieren noch kleinere Korpora in Französisch, Niederländisch und Sorbisch, sowie Korpora eingeschränkter Sachgebiete.

Ziel bei der Erstellung war es, eine umfassende Datenbasis zu schaffen, um weitere Untersuchungen zu erleichtern. Die verarbeiteten Texte bestehen hauptsächlich aus einer Auswahl verschiedener Tageszeitungen, sowie Nachschlagewerken und von Verlagen für das Projekt freigegebenen Texten. Aus urheberrechtlichen Gründen dürfen die Texte nur satzweise gespeichert werden, weswegen der Satz die größte Einheit im Wortschatz darstellt.

Auf die Rohdaten gestützt existieren verschiedene Datenbanken, die zur Strukturierung und als Vorarbeit für linguistische Algorithmen erstellt wurden. Als Vertreter seien hier die Kollokationsdatenbank, in der

Wortpaare, die signifikant häufig miteinander in Sätzen oder nebeneinander auftreten, und die Datenbanktabelle *Saetze*, die für alle indizierten Wörter Beispielsätze liefert, genannt.
Im Gegensatz zum traditionellen Lexikon stellt die Wortschatz-Datenbank ein Vollformenlexikon dar, was sich aus der Konstruktion durch automatischen Aufbau ergibt. Hier gelten verschiedene Zeichenketten als verschiedene Wörter.
Zum Datenbestand des deutschen Hauptkorpus gehören über 6 Millionen Wortformen, über 36 Millionen Sätze, über 2,5 Millionen Grammatikangaben sowie Angaben zur Pragmatik, Morphologie, Sachgebieten und weiteren Worteigenschaften.
In einem Eintrag stehen die absolute Häufigkeit im Gesamtkorpus, Synonyme, Antonyme, Grammatikangaben, Sachgebiete, Beispielsätze und weiteres (siehe [Quasthoff 1998], [Quasthoff 2002a]).
Der Wortschatz stellt einen Korpus im linguistischen Sinne dar: Bei der Auswahl der Texte wurde auf Vielfalt von Quellen und Sachgebieten geachtet, das Hauptkorpus wächst nicht weiter, liegt in maschinenlesbarer Form vor und ist mit eingeschränkten Suchmöglichkeiten im Internet seit März 1998 frei verfügbar.

Im Rahmen dieser Arbeit wurde insbesondere die *Saetze*-Tabelle dazu benutzt, Belegstellen von Wörtern im Hauptkorpus aufzufinden.
Desweiteren wurde eine Anwendung erstellt, um aus dem *Wdtaktuell*-Korpus, der jeweils täglich frisch diverse Tageszeitungen enthält, Namen zu extrahieren.

1.3 Semantische Relationen

Semantik ist die Lehre der Bedeutung, und das Teilgebiet der Linguistik, welches am meisten mit Sprachphilosophie verwoben ist. Während die Philosophen sich mit der Natur der Bedeutung an sich auseinandersetzen,

ist die Aufgabe des Semantikers, herauszufinden, wie sich Bedeutung in der Sprache manifestiert. Dabei ist die Definition von Bedeutung alles andere als einfach, was sich allein schon an den verschiedenen Verwendungen des Stammes "bedeut-" festmachen läßt, wie folgende Sätze veranschaulichen:

Schwarze Wolken bedeuten Regen.

Englisch "meaning" bedeutet "Bedeutung".

Nach schweren Fehlern verlor sein Einfluß an Bedeutung.

Geld bedeutet mir nichts.

Im Mittelpunkt der Semantik steht das *Semantische Kompositionalitätsprinzip*: "Die Interpretation eines komplexen sprachlichen Ausdrucks entsteht aus der Interpretation der einfachen Ausdrücke, aus denen er zusammengesetzt ist, und der Art und Weise ihrer syntaktischen Kombination" ([Pause 2002], S. 4).

Die kleinsten bedeutungstragenden Einheiten in der Linguistik heißen Morpheme und stellen sinntragende Teile eines Wortes dar. Aus diesen sind die Wörter zusammengesetzt, die dann wiederum die Sätze bilden, aus denen ein Text aufgebaut wird.

Die Bedeutungsbeziehungen zwischen den bedeutungstragenden Einheiten heißen auch *Semantische Relationen.*

Für das automatische Verstehen der Bedeutung eines Textes ist es unerläßlich, sämtliche semantischen Relationen als solche zu erkennen. Wie viele semantische Relationen existieren, ist nicht bekannt, da die Granularität fast beliebig hoch angesetzt werden kann, um alle Facetten der Bedeutungskomposition zu erfassen. Eine technische Umsetzung muß daher immer einen Kompromiß zwischen ausreichender Repräsentation und Machbarkeit schließen.

-

1.3.1 Arten von semantischen Relationen

Die von I.A. Mel'čuk und A.K. Zholkovsky begründete *Meaning Text Theory* unterschiedet 61 Gruppen von lexikalischen Funktionen, die durch 13 Modifizierer sowie durch verfeinernde Indizes weiter spezialisiert werden können ([Steele 1990], Kap. 3). Diese beschreiben systematisch gewisse semantische Relationen zwischen *Lexemen*, also auf Wortebene.
Es wird zwischen *paradigmatischen* und *syntagmatischen Beziehungen* unterschieden. Paradigmatische Beziehungen haben mit mentalen Assoziationen zu tun und beziehen sich auf das Verhältnis der sprachlichen Elemente im gesamten Sprachsystem, während die syntagmatischen Beziehungen die Aneinanderreihung der sprachlichen Elemente zur linearen, beobachtbaren Sprache bestimmen ([Bartschat 1996]). Beispiele für paradigmatische Beziehungen sind *Synonymbeziehung, Antonymbeziehung, Teil-Von-Beziehung* und *Derivationsbeziehungen* zwischen Wortarten, als Vertreter von syntagmatischen Beziehungen gelten zum Beispiel *verbale Operatoren*, die angeben, welche Verben neben welchen Nomen gebraucht werden können.

Viele der von theoretischen Linguisten aufgestellten Relationen lassen sich nicht oder nur sehr schwer automatisch auffinden, auf der anderen Seite existieren für die automatisch auffindbaren Relationen teilweise keine Namen in der Linguistik. Die linguistische Klassifikation kann jedoch als Leitlinie und Maß dafür gelten, wie viel an dieser Stelle schon erreicht wurde beziehungsweise noch erreicht werden muß.
In dieser Arbeit werden lediglich semantische Relationen behandelt, die sich durch *Pattern Matching* auf der Wortfolge oder der Abfolge aufgrund flacher Eigenschaften zugewiesener Tags erkennen lassen.

-

1.3.2 Durch Pattern Matching auffindbare semantische Relationen

Ein *Tag* (engl.: Etikett) ist ein Kürzel, das einem Symbol (hier im Fall der natürlichen Sprache: einem Wort) aufgrund dessen Eigenschaften zugewiesen wird. Das *Tagging* erfolgt durch einen *Tagger*, der die zu taggende Symbolfolge (einen Text) einliest und den Symbolen (hier: Wörter, Satzzeichen, Numerale etc.) passende Tags aus dem *Tagset* (Gesamtheit aller Tags) zuweist.

Ein *Pattern* ist ein regulärer Ausdruck bestehend aus Symbolen oder Tags. Tritt in der Symbolfolge das Muster des Pattern auf, so spricht man von einem *Match*.

Abbildung 1.3.2-1 zeigt ein Beispiel für einen getaggten Text, in dem ein beispielhafter Pattern matcht.

Tagset:

GR: Großgeschriebenes Wort

KL: Kleingeschriebenes Wort

SZ: Satzzeichen

DET: Artikel

Pattern: DET KL KL GR

Text:	Auf	der	großen	grünen	Wiese	liegt	ein	großer	grauer	Berg	,	...
Tags:	GR	**DET**	**KL**	**KL**	**GR**	KL	**DET**	**KL**	**KL**	**GR**	SZ	
		| -	**Match**		-|		| -	**Match**		-|	

Abb. 1.3.2-1 Beispiel für Tags und Pattern Matching

Die Größe des Tagsets und damit die erreichte Granularität ist abhängig von der Problemstellung. Kleinere Tagsets bedeuten schnellere Verarbeitungsgeschwindigkeit und einfachere Pattern. Es ist jedoch auch

-

möglich, auf Standard-Tagsets wie das der *Penn Tree Bank* und existierende Tagger, wie z.B. den auf *Hidden Markov-Modellen* basierenden *TNT* von Thorsten Brants ([Brants 2000]) oder den regelbasierten *Part-of-Speech-Tagger* (POS-Tagger) von Eric Brill ([Brill, 1992]) zurückgreifen.
Die Pattern werden dazu verwendet, semantische Relationen im Text zu erkennen. Matcht ein Pattern, so liegt vermutlich die mit diesem Pattern assoziierte Relation vor.

Tabelle 1.3.2-2 zeigt einige Relationen, die durch Pattern auf Tags oder Wörtern erkennbar sind. Die Liste, insbesondere die aufgeführten Pattern, erhebt keinen Anspruch auf Vollständigkeit. Die benutzten Tags entsprechen dem in der Abbildung 1.3.2-1 verwendeten Tagset. Die fett markierten Patternteile entsprechen den in der Relation stehenden Wörtern.

Beschreibung der Relation	**Beispielpattern**	**Textbeispiel**	**Bemerkung**
Teil-von: X ist Teil von Y	DET **GR** "ist ein Teil von" DET **GR**	eine **Tür** ist ein Teil von einem **Haus**	nur selten stehen Teil-von-Beziehungen direkt im Text, für automatischen Ansatz siehe [Berland 99]
X ist Synonym von Y	**GR** ", auch" **GR** "genannt"	**Linguistik**, auch **Sprachwissenschaft** genannt	nur selten stehen Synonymbeziehungen direkt im Text
X ist Antonym von Y	**GR** "ist das Gegenteil von" **GR**	**Sommer** ist das Gegenteil von **Winter**	nur selten stehen Antonymbeziehungen direkt im Text
X ist Hyperonym zu Y	DET **GR** "ist ein" **GR**	eine **Katze** ist ein **Tier**	ebenso selten im Text
Vorname zu Nachname	KL **GR** NN	sagte **Klaus** Müller	siehe Kapitel 3.1
Nachname zu Vorname	KL VN **GR** KL	Klaus **Müller** sagte	siehe Kapitel 3.1
Verb zu Name	**KL** VN NN	**sagte** Klaus Müller	siehe Kapitel 3.2
Insel	"Insel" **GR**	die Insel **Rügen**	siehe Kapitel 3.2

Urheber	"das Werk" **GR** von **VN NN**	das Werk **Ulysses** von **James Joyce**	Studie im WWW-Kontext: siehe [Brin 1998]
Firmenübernahme	**GR** "übernimmt" **GR**	**AOL** übernimmt **Netscape**	Studie im WWW-Kontext: siehe [Duclaye 2002]
Adjektiv zu Substantiv	DET **KL GR**	die **grünen Ideen**	
Mörder zu Opfer	**GR** "tötete" **GR**	**Oswald** tötete **Kennedy**	siehe [Duclaye 2002]
...			

Tabelle 1.3.2-2: Semantische Relationen, die von Pattern erkannt werden können

Aus obenstehender Tabelle ist ersichtlich, daß die klassischen semantischen Relationen nur selten von Pattern erkannt werden können, weil sie von paradigmatischer Natur sind und somit die Elemente der Relation nicht zwangsläufig im Text zusammen auftreten. Desweiteren wird deutlich, daß dies für syntagmatische und für in der linguistischen Klassifikation nicht enthaltene Relationen möglich ist.

Pattern-Regeln werden in der Sprachverarbeitung zum automatischen Textverstehen eingeschränkter Bereiche benutzt, z.B. für das automatische *Ausfüllen von Templates* aus Stellenanzeigen ([Califf 1997]). Bei der Konstruktion der Pattern muß beachtet werden, daß stark generalisierende Pattern zu oft matchen und somit nur sehr unscharf die gewünschte Relation liefern, zu spezielle Pattern hingegen matchen zu selten und liefern nur unvollständige Daten.

Im Folgenden soll ein Verfahren vorgestellt werden, das mit Hilfe von unscharfen Pattern und einem Verifikationsschritt Elemente semantischer Relationen mit hoher Genauigkeit liefert.

-

2 Der Pendel-Algorithmus

Das nachfolgend beschriebene Verfahren stützt sich auf folgende Beobachtung: Vornamen und Nachnamen stehen in natürlichsprachlichem Text oftmals hintereinander. Es sollte daher möglich sein, von bekannten Vornamen ausgehend zugehörige Nachnamen zu finden und umgekehrt. Der naive Ansatz, einfach die im Text rechts von Vornamen stehenden Wörter als Nachnamen und die links von Vornamen stehenden Wörter als Vornamen zu bezeichnen, schlägt fehl, auch wenn man die Großschreibung dieser Wörter verlangt. Zu viele Nichtnamen passen in das Raster und nach einigen Iterationen dieses Lernschrittes ist die Genauigkeit auf ein Minimum gesunken. Es muß also ein Verifikationsschritt eingeführt werden, der die durch Nachbarschaftsregeln gefundenen Namenskandidaten überprüft und mit hoher Genauigkeit Namen von Nichtnamen unterscheidet: Belegstellen des Namenskandidaten werden daraufhin überprüft, ob der Vornamenskandidat (bzw. Nachnamenskandidat) genügend oft vor (bzw. hinter) einem schon bekannten Nachnamen (bzw. Vornamen) im Text auftritt. Ist dies der Fall, so wird der Name gelernt und kann zum Auffinden von weiteren Namen verwendet werden.

Im Rahmen vorliegender Arbeit wurde auf diese Beobachtung gestützt, der *Pendel-Algorithmus* entwickelt, eine Verfahrensweise, mit Hilfe derer bestimmte semantische Relationen, nicht nur Namen, im Korpus aufgefunden werden können.

Linguistisch gesehen paßt dies am ehesten zum Substitutionstest für Kategorien: Hier wird die Zugehörigkeit von Wörtern zu einer (syntaktischen) Klasse überprüft, indem das fragliche Wort für ein Wort der jeweiligen Klasse im Satz eingesetzt wird, der resultierende Satz wird auf grammatikalische Korrektheit überprüft. Ähnlich geschieht die

Klassifikation von Namen in obigem Beispiel, die Klassen sind jedoch semantischer Natur und die Korrektheitsprüfung wird durch Häufigkeit im Text approximiert.

2.1 Voraussetzungen an die Relation

Es bleibt zu klären, wie diese Relationen beschaffen sein müssen, damit sie geeignet für dieses Verfahren sind, bevor das Verfahren im Detail beschrieben wird.

In diesem Zusammenhang interessieren n-stellige, mindestens zweistellige Relationen $R \subseteq A_1 \times A_2 \times ... \times A_n$ mit $A_i \cap A_j = \varnothing$ für $i \neq j$.
Der Algorithmus soll folgendes leisten:

- Auffinden der Relation, gegeben einige Elemente aus $A_1,...,A_n$
- Auffinden möglichst vieler einzelner Elemente der Mengen $A_1,...,A_n$.
- Auffinden möglichst vieler Tupel $(a_1,...,a_n)$, die in der Relation R stehen.

Im natürlichsprachlichen Fall sind Elemente der Mengen A_i Wörter, Instanzen der Relation R sind Ausdrücke, die die Relation R realisieren.
Im Folgenden werden die Relationen sowie die Mengen A_i in Großbuchstaben mit selbsterklärenden Namen bezeichnet.

Definition: bipartiter Graph

Ein Graph (V,E), V Knotenmenge, E Kantenmenge heißt bipartit, wenn es Partitionen $V_1 \subseteq V$, $V_2 \subseteq V$, $V_1 \cap V_2 = \varnothing$, $V_1 \cup V_2 = V$ gibt, so daß für alle Elemente $e=(v_1,v_2) \in E$ gilt: $v_1 \in V_1$, $v_2 \in V_2$.

-

Definition: n-partiter Schichtgraph

Ein Graph G (V,E), V Knotenmenge, E Kantenmenge heißt n-partiter Schichtgraph, wenn es Partitionen $V_1, V_2, \ldots V_n$, $V_i \subseteq V$, $V_i \cap V_j = \varnothing$ für $i \neq j$, $\bigcup_i V_i = V$ gibt, so daß für alle Paare von Partitionen V_i und V_{i+1} (i=1,..,n-1) gilt: Die Einschränkung von G auf $V_i \cup V_{i+1}$ ist ein bipartiter Graph und für alle Paare von Partitionen V_i, V_j, $|i-j| > 1$ gilt: es existieren keine Kanten der Form (v_i, v_j) mit $v_i \in V_i$, $v_j \in V_j$.

Die Elemente der Mengen $A_1,..,A_n$ können als Knoten, das benachbarte Auftreten als Kanten eines Graphen aufgefaßt werden. Da die Mengen $A_1,...,A_n$ paarweise disjunkt sind, ist die Struktur des Graphen die eines n-partiten Schichtgraphen (im Fall der zweistelligen Relation: bipartit).

Instanzen der Relation bedeuten im Graphen Knoten eines Weges von A_1 nach A_n, die mit einer Kante verbunden sind.

Nur Relationen im oben ausgeführten Sinne sind geeignet für das Pendel-Verfahren, darüber hinaus muß der n-partite Schichtgraph, der die Relation beschreibt, eine gewisse Dichte vorweisen, dazu später mehr.

Drei Fälle sollen hier unterschieden werden: dichte n-partite Schichtgraphen, n-partite Schichtgraphen mit ungleichmächtigen Partitionen und dünne n-partite Schichtgraphen. Abbildung 2.2-3 veranschaulicht diese für den bipartiten Fall.

-

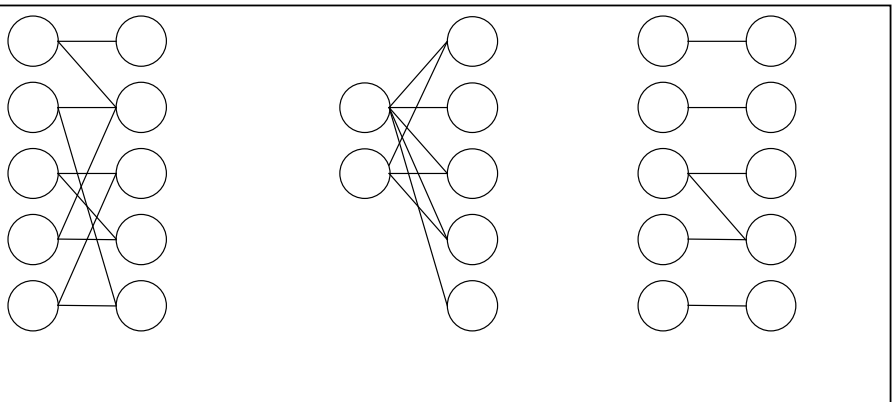


Abbildung 2.2-3: verschiedene Arten bipartiter Graphen

Ein natürlichsprachliches Beispiel für den dichten Fall ist die Relation PERSONENNAME⊆VORNAME×NACHNAME, da es viele Leute mit demselben Vornamen bzw. demselben Nachnamen gibt. Das Verfahren nutzt diese Eigenschaft aus und versucht anhand von bekannten Elementen, noch unbekannte, den bekannten benachbarte Elemente zu finden.

Die Relation INSEL⊆INSELBEZEICHER×INSELNAME ist ein Beispiel für den ungleichmächtigen Fall (Textbeispiel: Insel Rügen). Hier gibt es an der ersten Stelle der Relation wenig Inselbezeichner, wohingegen die Anzahl der Inseln an zweiter Position beträchtlich ist.

Als Beispiel für den dünnen Fall läßt sich die Relation GEFLÜGELTES-WORT⊆TEIL1×TEIL2 anführen, in der Ausdrücke wie "Blauer Engel", "elfter September" und "Schwarzer Freitag" stehen. Derartige Relationen sind dadurch charakterisiert, daß Elemente der Mengen WORT1 bzw. WORT2 nur sehr selten in verschiedenen Instanzen der Relation auftreten.

-

2.2 Algorithmus

In diesem Abschnitt wird zunächst der Algorithmus im Gesamten vorgestellt, darauf folgt eine Erklärung der Schritte.
Abbildung 2.2-1 erklärt den Algorithmus auf halbformale Weise.

```
Lade Beispielitems
Lade Regeln
Lade Grundwissen
StartItems newI:=Beispielitems
Wissen K:=Beispielitems+Grundwissen        //Grundwissen: Artikel etc.
do {
    Items I:=newI
    newI:=leer
    for all i ∈ I {
        text_i:=Hole Sätze aus Korpus, die i enthalten  // Suchschritt
        kandidaten:=Wende Regeln auf text_i an
        for all k ∈ kandidaten {                        // Verifikationsschritt
            kandText:= Hole Sätze aus Korpus, die k enthalten
            rating_k:= Wende Regeln auf kandText an und überprüfe, wie oft k wie in text_i klassifiziert wird
            wenn rating_k hoch genug, füge k zu K und zu newI hinzu
        } // for all k
    } // for all i
} while newI nicht leer.
```

Abbildung 2.2-1 Pendel-Algorithmus

Im Folgenden werden die Elemente $a_i \in A_i$, die aus Sicht des Algorithmus in der Relation $R=(A_1,...,A_n)$ stehen, mit *Items* bezeichnet.
Der Algorithmus startet mit einem gewissen Grundwissen, dazu gehören einige Elemente der Mengen A_i (*Beispielitems*) und Regeln, durch die aus der

-

Umgebung von bekannten Symbolen (Wörtern) neue Symbole gelernt werden können. Desweiteren gehört zum Grundwissen eine Liste von häufigen Wörtern, die zur Klassifikation beitragen können, wie zum Beispiel Pronomen, Artikel u.ä.

Die Liste der neu gefundenen Items wird auf die Liste der bekannten Beispiele gesetzt.

Nach der Initialisierung beginnt eine zweifach verschachtelte Programmschleife. In der äußeren Schleife wird jedes neue Item i behandelt. Für das Item i werden Belegstellen aus dem Korpus gesucht, auf welche die Regeln angewendet werden. Auf diese Weise entsteht eine Liste von möglichen neuen Items (*kandidaten*), mit Klassifizierungen auf Zugehörigkeit zu A_1, A_2,... oder A_n.

In der inneren Schleife werden die Kandidaten auf folgende Weise überprüft: Es werden Belegstellen für die Kandidaten gesucht, wenn der Kandidat k hier aufgrund derselben Regeln oft genug genauso klassifiziert wird wie zuvor, wird k zum Wissen und zu den neuen Items des nächsten Durchgangs der äußeren Schleife hinzugefügt.

Auf diese Weise liefern die jeweils in einem Schritt gelernten Items die Kandidaten für den nächsten Schritt. Der Algorithmus terminiert, wenn keine neuen Items mehr gefunden werden.

2.2.1 Grundwissen, Tagset und Regeln

Das Grundwissen teilt sich in einen problemunabhängigen und einen problemabhängigen Teil. Problemunabhängig sind häufige Füllwörter wie Artikel, Pronomen und Präpositionen. Je mehr verschiedene Arten dieser Wörter unterschieden werden sollen, desto mehr Tags sind nötig, was die Regeln verkomplizieren kann.

Im problemabhängigen Teil des Grundwissens befinden sich die Beispiele für die Mengen, über die die gesuchte Relation definiert ist. Für jede der Mengen A_i, sowie für die verschiedenen Klassen des problem-

-

unabhängigen Grundwissens wird ein Tag benötigt, um die Pattern definieren zu können.

Regeln bestehen aus einem Kopf, der das aus einer Tagfolge festgelegte Pattern enthält. Eine Position ist mit einem Stern markiert. Nach dem Pfeil (→) steht das neue Tag, das dem Wort, zu dem das mit dem Stern markierte Tag gehört, eventuell zuzuweisen ist.

Schreibweise: $Tag_1\ Tag_2\ \ldots\ Tag_j*\ \ldots\ Tag_n \rightarrow Tag_{neu}$
Die Tag_i sind nicht notwendigerweise verschieden, die Länge des Patterns ist nach oben nicht beschränkt.

Beispiele:
Tagset: A, B, C, D

Regel: A B* C → D
lies: Wenn im Text die Tagfolge A B C auftaucht, merke für das Wort mit Tag B die Klassifizierung zum Tag D vor.

Regel: A A* B B D → C
lies: Wenn im Text die Tagfolge A A B B D auftaucht, dann merke für das zweite Wort mit Tag A die Klassifizierung zum Tag C vor.

Abbildung 2.2.-2 Schreibweise für Regeln

Desweiteren werden Tags für flache Eigenschaften wie Großschreibung, Kleinschreibung, Numerale und Satzzeichentags vergeben.
Wörter können mehrere verschiedene Tags erhalten, beispielsweise wird ein großgeschriebenes Wort, das im Grundwissen enthalten ist, sowohl das Tag für Großschreibung als auch das Tag seiner bekannten Klasse zugewiesen bekommen.

-

Im praktischen Teil dieser Arbeit wurden Regeln nur aus Tags (und nicht aus Symbolen) aufgebaut. Um Symbole direkt zu simulieren, können für einzelne Wörter spezielle Tags definiert werden. Desweiteren wurden nur lineare Pattern, also Abfolgen aus Tags verwendet, eine Erweiterung auf reguläre Ausdrücke erscheint an dieser Stelle jedoch sinnvoll.
Die Pattern dienen dazu, Wörter, die einer bestimmten Bedingung genügen, und deren Umgebung mit dem Pattern matcht, auf Zugehörigkeit zu einer der vorher erwähnten Menge A_i, im Folgenden auc Klassen genannt, zu klassifizieren. Abbildung 2.2-2 zeigt einige Beispielregeln in Kurzschreibweise, und erklärt, wie diese zu lesen sind.
Das Auffinden geeigneter Regeln für eine bestimmte Relation kann entweder manuell oder mit einem automatischen Lernverfahren (siehe Kap. 2.4) geschehen.

2.2.2 Annotation mit bekannten Items und regulären Ausdrücken

Um die aus Tags bestehenden Pattern mit dem Text zu matchen, der Belegstellen der gesuchten Wörter enthält, muß der Text, welcher in Rohform aus der Wortschatz-Datenbank geliefert wird, zunächst durch einen Tagger annotiert werden. Dazu wird das vorhandene Grundwissen genutzt. Desweiteren können reguläre Ausdrücke zur Annotation erstellt werden, zum Beispiel `[A-Z|ÄÖÜ][a-z|äöüß]+` für großgeschriebene Wörter. Jedes Wort des vorliegenden Textes wird gegen die regulären Ausdrücke gematcht und es wird überprüft, ob das vorliegende Wort schon im Grundwissen enthalten ist. Trifft dies zu, dann wird dem Wort das Tag seiner Klasse zugewiesen. Am Ende des Annotationsvorganges liegen zwei Listen vor: Die Abfolge der Wörter und Sonderzeichen, wie sie im Text auftraten, sowie die Abfolge der zugehörigen Tags. Abbildung 2.2.2-1 zeigt eine Beispielannotierung eines kurzen Textstückes.

-

Reguläre Ausdrücke (REs):

Ausdruck	*Zielklasse*
`[A-Z|ÄÖÜ][a-z|äöüß|-]+`	GR
`[a-z|äöüß|-]+`	KL
`[0-9|,\.]+`	NUM
`[\\\.,;=:\"\'-?!/()]`	SZ

Grundwissen:

Wort	*Zielklasse*
der	DET
Peter	VN
Maier	NN

Text:	bestritt	Peter	Müller	gegenüber	dem	Vorsitzenden	Maier	,	der
Tags nach REs:	KL	GR	GR	KL	KL	GR	GR	SZ	KL
Tags:	KL	GR,VN	GR	KL	KL	GR	GR,NN	SZ	KL,DET

Abbildung 2.2.2-1: Beispielannotierung

Um das Matchen der Pattern effektiver zu machen, wurde eine vereinfachte Form des für kontextfreie Grammatiken entwickelten Algorithmus von J. Earley [Earley 1970] implementiert. Die Motivation ist folgende: Wird ein Pattern der Länge n mit jedem Teilstück einer Tagfolge verglichen, so wird jedes Tag im naiven Ansatz n mal gelesen. Es reicht aber aus, jedes Tag nur einmal zu verarbeiten. Jeder Pattern bekommt einen Punkt, der sich anfangs vor dem ersten Tag im Pattern befindet. Wird ein Tag gelesen, das dem Tag nach dem Punkt entspricht, bewegt sich der Punkt im Pattern einen Schritt weiter. Bei Bewegen des Punktes über das erste Tag wird zusätzlich noch eine Kopie des Patterns mit dem Punkt am Anfang in die Patternmenge eingefügt. Gelangt der Punkt in einem Pattern hinter das letzte Tag, so wurde der Pattern in der Tagfolge erkannt. Wird ein Tag gelesen, das nicht dem Tag hinter dem Punkt entspricht und befindet sich der Punkt nicht am Anfang, so wird das Pattern aus der Patternmenge gelöscht. Abbildung 2.2.2-2 verdeutlicht diese effiziente Art von Pattern Matching an einem Beispiel.

-

Tagset:	Pattern:	Markierungen:
A,B,C	A C C	Unterstrichen: Pattern matcht
	C A B	*Kursiv:* Pattern wird gestrichen
		Fett: Punkt wird bewegt

Symbolfolge: A A C A C C B

Symbol	(Anfang)	A	A	C	A	C	C	B
Pattern-menge	•**A C C** •C A B	*A •C C* •C A B •**A C C**	**A •C C** •**C A B** •A C C	*A C •C* **C •A B** •**A C C** •C A B	*C A •B* **A •C C** •**C A B** •A C C	•**C A B** **A C •C** *C •A B* •A C C	*C •A B* A C C• •C A B •A C C	•A C C •C A B

Abbildung 2.2.2-2: effizientes Pattern Matching

2.2.3 Suchschritt

Im Suchschritt wird anhand von bekannten Items versucht, neue Items aufzufinden. Hierbei wird wie folgt vorgegangen: Zu einem bekannten Item werden Beispielsätze aus der Datenbank gesucht. So wird sichergestellt, daß in den Sätzen jeweils mindestens ein bekanntes Item vorkommt. Die Annahme ist dabei, daß neben diesem Item andere, noch unbekannte Items stehen, die mit dem bekannten Item in der gesuchten Relation stehen.

Beobachtungen zufolge enthalten diese Beispielsätze meist sogar mehrere Instanzen der gesuchten Relation, dies ist natürlich problemabhängig.

Nach der Annotation der Beispielsätze werden die Regeln auf die Liste der Tags angewandt. Es werden Kandidaten für neue Items geliefert, zusammen mit den vermuteten Klassifizierungen, die im Verifikations-schritt überprüft werden. Dies erfolgt für alle im letzten Schritt neu gewonnenen Items.

-

2.2.4 Verifikationsschritt

Um zu ermöglichen, daß Regeln mit hoher Abdeckung aber geringer Schärfe verwendet werden können, ist ein Verifikationsschritt erforderlich. Zu jedem Kandidaten k werden Beispielsätze aus der Datenbank abgefragt, die wiederum nach oben beschriebener Methode annotiert werden. Zu beachten ist hierbei, daß dem jeweiligen k noch nicht das Tag seiner vermutete Klasse zugewiesen wird. Im Gegensatz zum Suchschritt interessieren an dieser Stelle nur die Klassifikationen von k. Überschreitet das Verhältnis der Klassifikationen von k, die der im Suchschritt gefundenen Klasse von k entsprechen, zur Gesamtanzahl der Vorkommen von k einen gewissen Schwellwert, so wird k mit seiner Klassifikation ins Wissen aufgenommen und gilt fortan als Item. Im nächsten Suchschritt wird dann k zum Finden von neuen Kandidaten benutzt.
Es hat sich als notwendig erwiesen, die Mindestanzahl positiver Belegstellen für k auf zwei festzusetzen.

2.2.5 Einordnung in bekannte Verfahren

Der Algorithmus implementiert *Expectation-Maximization* (siehe [Dempster 1977], [Collins 1999]) auf die Art, daß abwechselnd der Suchschritt und der Verifikationsschritt iteriert werden. Je mehr Items gefunden werden, desto mehr neue Items werden in der nächsten Iteration geliefert. Auf diese Weise werden beide Schritte effektiver, desto mehr Iterationen durchgeführt werden.
Das Verfahren läßt sich auch in die *Bootstrapping-Algorithmen* (vgl. z. B. [Riloff 1999]) einordnen, deren Eigenschaft es ist, aus wenig Anfangswissen unter Zuhilfenahme von neu erlerntem Wissen weiteres Wissen zu gewinnen.
Desweiteren bleibt zu erwähnen, daß hier eine Methode von unüberwachtem Lernen vorliegt: Der Algorithmus läuft ohne menschliche

-

Interaktion und die Bewertung der gelernten Items bleibt einzig ihm selbst überlassen.

Vergleichbare Ansätze lassen sich in [Riloff 1999], [Brin 1998], [Collins 1999] und [Duclaye 2002] finden. [Riloff 99] beschreibt einen Zwei-Level-Bootstrapping Mechanismus, der in alternierenden Schritten Items und Extraktionsregeln lernt und mit Hilfe einer Meta-Ebene nur die besten Items in die nächste Iteration übernimmt. Auf ähnliche Weise wird in [Brin 1998] die Relation URHEBER-VON$\subseteq$AUTOR$\times$TITEL aus einem Korpus von Webseiten extrahiert. Hier werden Items dazu benutzt, Extraktionsregeln zu finden, die wiederum neue Beispiele liefern. Collins und Singer benutzen in [Collins 1999] einerseits die Buchstabenfolge der Items, andererseits deren lokalen syntaktischen Kontext, um zwei Klassifikatoren zu lernen, die sich gegenseitig alternierend trainieren. In [Duclaye 2002] wird schließlich der Aufbau eines Frage-Antwort-Systems beschrieben, in dessen Rahmen Reformulierungen für Beispielinstanzen von semantischen Relationen im World Wide Web gesucht wurden.

2.2.6 Betrachtungen zur erforderlichen Korpusgröße

Um eine sinnvolle Anzahl von Daten mit dem Pendel-Algorithmus zu ermitteln, muß das verwendete Korpus eine gewisse Mindestgröße aufweisen. Da das Verfahren anhand von Belegstellen überprüft, ob ein Wort mit anderen Wörtern in Relation steht, muß gewährleistet sein, daß das betreffende Wort im Korpus häufig genug auftritt, um eine Aussage festlegen zu können.

Laut dem Zipfschen Gesetz (siehe [Zipf 1929]), welches im Zusammenhang mit Korpuslinguistik besagt, daß das Produkt der Wortfrequenz f und des Ranges r (der Position des Wortes in einer nach Frequenz absteigend sortierten Wortliste, wobei gleichfrequente Wörter denselben Rang haben) konstant ist, kommen die Hälfte aller im Korpus enthaltenen Wörter nur einmal, ein weiteres Viertel nur zweimal vor.

-

Für das Verfahren sinnvoll angesehen werden nur Wörter, die mindestens zwei passende Belegstellen haben, was bei einem kleinen Korpus nur selten gegeben ist. Experimente mit verschiedenen Korpora zeigten, daß der Algorithmus sinnvolle Ergebnisse ab ca. drei Million Sätze liefert.

2.2.7 Nachteile des Verfahrens

Das Verfahren ordnet Symbole (Wörter) gewissen Klassen zu, auf der Grundlage, wie oft das jeweilige Symbol in einem Kontext gesehen wurde, der für diese Klasse spricht. In natürlichsprachlichen Symbolfolgen werden folgende Arten von Wörtern nicht oder nur selten gelernt:

- Ambige Wörter, deren eine Lesart zur gesuchten Klasse gehört, deren andere Lesart jedoch häufiger im Korpus vertreten ist, z.B. der Vorname "*Roman*" oder der Nachname "*Marseille*"
- Seltene Wörter, die nicht genug geeignete Belegstellen aufweisen
- Cluster seltener Wörter, die nicht mit anderen Items auftreten und vom Pendelprozeß nicht besucht werden.

2.3 Theoretisches Verhalten

Die gewünschten Eigenschaften eines Verfahrens zum Auffinden semantischer Relationen in einem Korpus sind hoher *Recall*, also wenn möglich das Auffinden aller Elemente der beteiligten Mengen bei hoher *Precision*, es sollen nur die gewünschen Elemente und keine unerwünschten gefunden werden.

Definition: Precision

Der *Precision-Wert* P eines Verfahrens ist das Verhältnis der Anzahl relevanter gefundener Vorkommen, zur Gesamtzahl aller mit dem Verfahren gefundenen Vorkommen.

$$P = \frac{\text{Anzahl relevanter gefundener Stellen}}{\text{Anzahl gefundener Stellen insgesamt}}$$

-

Definition: Recall

Der *Recall-Wert* eines Verfahrens R ist das Verhältnis von der Anzahl tatsächlich gefundener relevanter Vorkommen zu der Gesamtzahl der relevanten Vorkommen im Suchraum.

$$R = \frac{\text{Anzahl relevanter gefundener Stellen}}{\text{Anzahl releventer Stellen im Suchraum}}$$

Definition: F-Wert

Der *F-Wert* F eines Verfahrens stellt ein Maß für das Zusammenspiel von Precision und Recall dar und berechnet sich als das harmonische Mittel von P und R:

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

Precision, Recall und F-Wert stellen gebräuchliche Meßlatten für automatische Informationsextraktionsverfahren dar.

Beide Basismaße (Precision und Recall) hängen von der Schärfe der Regeln sowie vom Akzeptanzschwellwert im Verifikationsschritt ab. Sind die Regeln zu spezialisiert oder der Schwellwert zu hoch, kann niedriger Recall mit hoher Precision erwartet werden. Bei niedrigem Schwellwert und unscharfen Regeln sollte sich hoher Recall, aber eine niedrige Precision einstellen. In der Praxis bedeutet das, einen Kompromiß einzugehen, um hohe Precision bei zufriedenstellendem Recall erreichen zu können und so den F-Wert zu optimieren.

2.3.1 Beschaffenheit von Ergebnismengen

Bei einem Bootstrapping-Verfahren können Programmabläufe durch die verschiedenen Arten ihrer Ergebnismenge unterteilt werden. Hier werden drei Arten unterschieden:

- *vorzeitige Konvergenz*: hohe Precision, niedriger Recall: Das Verfahren stoppt nach wenigen gefundenen Items, obwohl das Korpus viel mehr enthält.
- *Erfolgreicher Verlauf:* hohe Precision, hoher Recall: Die Menge der gefundenen Items stimmt weitgehend mit der Menge der im Korpus vorhandenen Items überein
- *Überlauf*: niedrige Precision, hoher Recall: Es werden zwar viele der gesuchten Items gefunden, jedoch auch zahlreiche falsche Items.

2.3.2 Beschränktes Wachstum bei erfolgreichem Verlauf

Im folgenden soll angenommen werden, daß das Verfahren wie oben beschrieben erfolgreich verläuft. In diesem Abschnitt wird untersucht, wie sich die Anzahl der zu erwartenden Items pro Iterationsschritt verhält.

Als Vereinfachung sei angenommen, daß jedes Item durch Suche und Verifikation genau N neue Items liefern kann. Pro Schritt sollte sich die Anzahl neu klassifizierter Items jeweils um N vervielfachen, was exponentielles Wachstum bedeutet. Nun ist aber die Gesamtanzahl der Items beschränkt und je höher die schon erreichte Abdeckung, desto mehr der N Items werden schon bekannt sein und sind somit nicht mehr neu klassifizierbar. Das mathematische Modell von beschränktem Wachstum exponentieller Natur ist das logistische Wachstum.

Definition: logistisches Wachstum

Die momentane Zuwachsrate ist proportional zu Bestand f(t) und Sättigungsmanko (S - f(t)). S ist der Sättigungswert.

Das *logistische Wachstum* wird durch folgende Differentialgleichung[1]. beschrieben:

$$f'(t) = k \cdot f(t) \cdot (S - f(t))$$
$$f(0) = a$$

(a Anfangswert, k konstant)

Abbildung 2.3.2-1 zeigt eine schematische Darstellung des logistischen Wachstums. Die monoton steigende Kurve beschreibt die Gesamtzahl von Items zur Zeit t, die andere Kurve die Anzahl der pro Zeiteinheit jeweils neu gefundenen Items. Es läßt sich zwischen der *Wachstumsphase*, in der die Anzahl der neu gefundenen Items ansteigt, und der *Sättigungsphase*, in der die Anzahl neu gefundener Items abnimmt, unterscheiden.

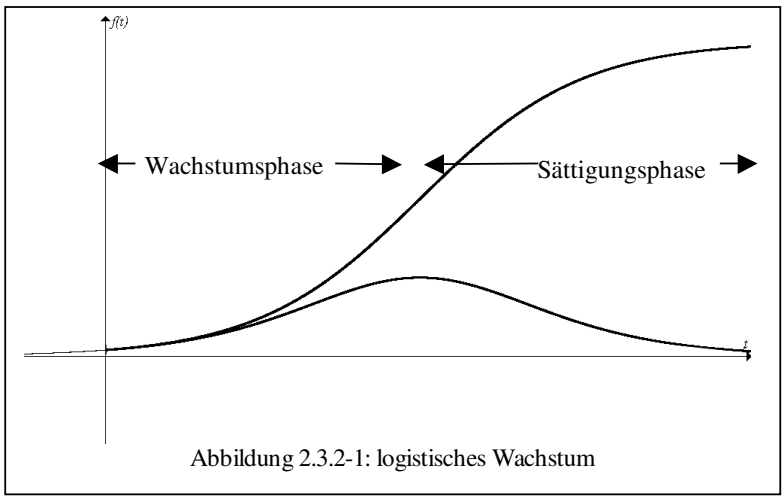


Abbildung 2.3.2-1: logistisches Wachstum

Das Wachstumsverhalten des Itembestandes beim Pendel-Algorithmus sollte in erster Näherung dem logistischen Wachstum folgen: a ist die Anzahl von Startitems, S die Gesamtanzahl auffindbarer Items im Korpus, f(t) der momentane Bestand an Items zum Iterationsschritt t und k ist ein

[1] Die explizite Lösung lautet: $f(t) = \frac{a \cdot S}{a + (S - a) \cdot e^{-S \cdot k \cdot t}}$

Maß für die durchschnittliche Anzahl von neuen Items, die ein bekanntes Item findet.

Für die erste Näherung wird nicht in Betracht gezogen, daß im Suchschritt zwar alle bekannten Items zur Annotation verwendet werden, jedoch nur für die im letzten Schritt gefundenen Items Beispielsätze gesucht werden. Den Messungen zufolge (siehe Kapitel 3) bildet jedoch die logistische Wachstumsfunktion die Realität in ausreichendem Maße ab.

2.3.3 Fehlerfortpflanzung

Bei einem unüberwachten Lernverfahren wie diesem kann es vorkommen, daß Wörter fälschlicherweise als der Relation zugehörig klassifiziert werden. Da diese in den nächsten Schritten als Grundwissen für weitere Klassifikationen verwendet werden, können sich Fehler fortpflanzen, indem falsche Items erneut falsche Items liefern.
Angenommen, ein Startitem ist falsch. Im nächsten Schritt produziert dieses Item unkorrekte Items, die in den Folgeschritten ihrerseits weitere falsche Items liefern. Die Fehlerrate jedoch bleibt in diesem Szenario zunächst konstant, weil in der Wachstumsphase die richtig klassifizierten Items jeweils viele neue richtige Items liefern, und steigt nur an, wenn falsche Items aufgrund richtiger Items gelernt werden. In der Sättigungsphase kann ein Anstieg der Fehlerrate erwartet werden, weil die Sättigung nur bei richtigen Items eintritt, während das Sättigungsmanko für falsche Items nach wie vor groß ist.

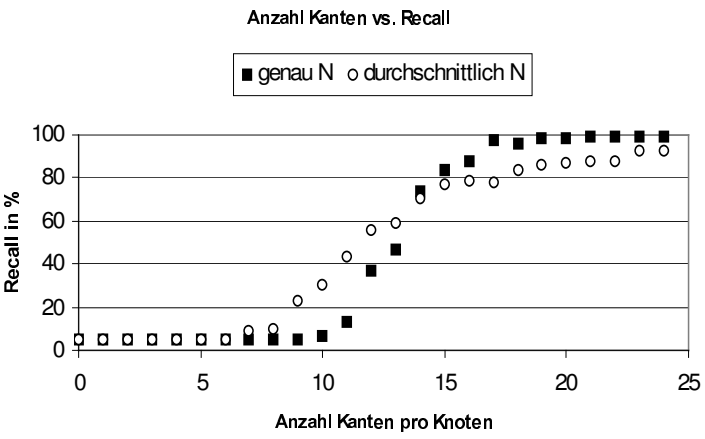
2.3.4 Anforderungen an die Symbolfolge

Es wurden Vorversuche durchgeführt, um zu untersuchen, welchen Anforderungen die Symbolfolge genügen muß, um mit dem beschriebenen Algorithmus durch Pattern auffindbare Relationen zu extrahieren.

-

Zunächst interessierte die Dichte des der Relation zugehörigen bipartiten Graphen, die diesem Zusammenhang die durchschnittliche Menge von Ausgangskanten eines jeden Knoten darstellt. Ein dünn besetzter Graph, in dem jeder Knoten nur mit ein oder zwei Kanten versehen ist, wird aufgrund des Verifikationsschrittes, der zur Bestätigung der Klassifizierung eines Knotens mehrere adjaszente, schon bekannte Knoten braucht, kaum vollständig abgesucht werden. Auf einem dichten Graph hingegen sollte das Verfahren hohen Recall erreichen, da hier ein noch unklassifizierter Knoten der einen Partition mit hoher Wahrscheinlichkeit mit einigen bekannten Knoten der anderen Partition zusammenhängt.

Bipartite Graphen mit gleich großen Partitionen zwischen je 50 und 750 Knoten wurden generiert, jeder Knoten aus der ersten Partition wurde mit N Knoten aus der anderen Partition verbunden, 5% der Knoten aus beiden Partitionen wurden als Grundwissen bereitgestellt, als Startitem



Abblildung 2.3.4-1: Recall vs. Anzahl von Kanten pro Knoten im bipartiten Graphen

wurde ein einziger Knoten verwendet. Der Suchschritt liefert in diesem Versuch alle mit dem bekannten Knoten verbundenen Knoten der anderen Partition, verifiziert wird ein Kandidat, wenn er mit mindestens zwei bekannten Knoten verbunden ist. Abbildung 2.3.4-1 zeigt den über alle Versuche gemittelten Recall für N von 1 bis 25, einmal für den Fall,

-

daß jeder Knoten aus der ersten Partition genau, einmal durchschnittlich N Kanten besitzt.

Für beide Fälle läßt sich ein Phasenübergang beobachten: für genau N Kanten findet dieser zwischen 11 und 16 Kanten statt. Der Übergang bei durchschnittlich N Kanten beginnt bei kleinerer Anzahl und erstreckt sich von 7 bis 16 Kanten. Aus diesem Vergleich wird deutlich, daß jeder Knoten durchschnittlich 17 Verbindungen in die andere Knotenpartition haben muß, damit nahezu alle Knoten gefunden werden. Desweiteren sind Knoten mit wenigen Kanten auffindbar, so lange es nur genug Knoten mit vielen Kanten im Graphen gibt, so daß eine Streuung über verschiedene Kantenanzahlen, wie sie auch in der natürlichen Sprache auftritt, für das Verfahren bei gleicher Gesamtzahl förderlich ist.

Um der Simulation von natürlicher Sprache näherzukommen, wurde in einem weiteren Vorversuch zum Suchen von Items im Text übergegangen. Die Texte bestanden aus jeweils 100 verschiedenen Wörtern aus den Mengen A, B und F, wobei die gesuchte Relation $R \subseteq A \times B$ war. Nach Generierung des Graphen von R wurden die Itempaare aus $A \times B$ zufällig aneinandergereiht, dazwischen befanden sich jeweils fünf Wörter aus F. Abhängig von einer Fehlerrate wurden falsche Itempaare aus $A \times F$ bzw. $F \times B$ eingestreut. Als Regeln standen `A X*` $\rightarrow$ `B` und `X* B` $\rightarrow$ `A` zur Verfügung, als Grundwissen wiederum je 5% der Mengen A und B, begonnen wurde mit einem Startitem.

Während beim ersten Vorversuch, in welchem die Precision per Definition 100% betrug, nur der Recall interessierte, stellt sich nun die Frage, ob das Einstreuen falscher Itempaare, das im Beispiel der Personennamen dem Vorkommen von einzelnen Namensteilen im Text entspricht, auf den Recall Auswirkungen hat und wie viele falsche Itempaare vom Verfahren ohne hohe Precision-Verluste verkraftet werden.

Abbildung 2.3.4-2 zeigt Precision und Recall für verschiedene Fehlerraten in Abhängigkeit von der durchschnittlichen Kantenanzahl pro Knoten im

-

Graphen. Eine Fehlerrate von 50% bedeutet, daß die Hälfte aller Stellen, auf die der Pattern einer Regel paßt, falsche Kandidaten im Suchschritt liefern. Die gepunktete Linie gibt die Baseline für die Precision an, bei der sämtliche vorkommende Wörter (insbesondere auch Wörter aus F) einer der Klassen A oder B zugeordnet werden.

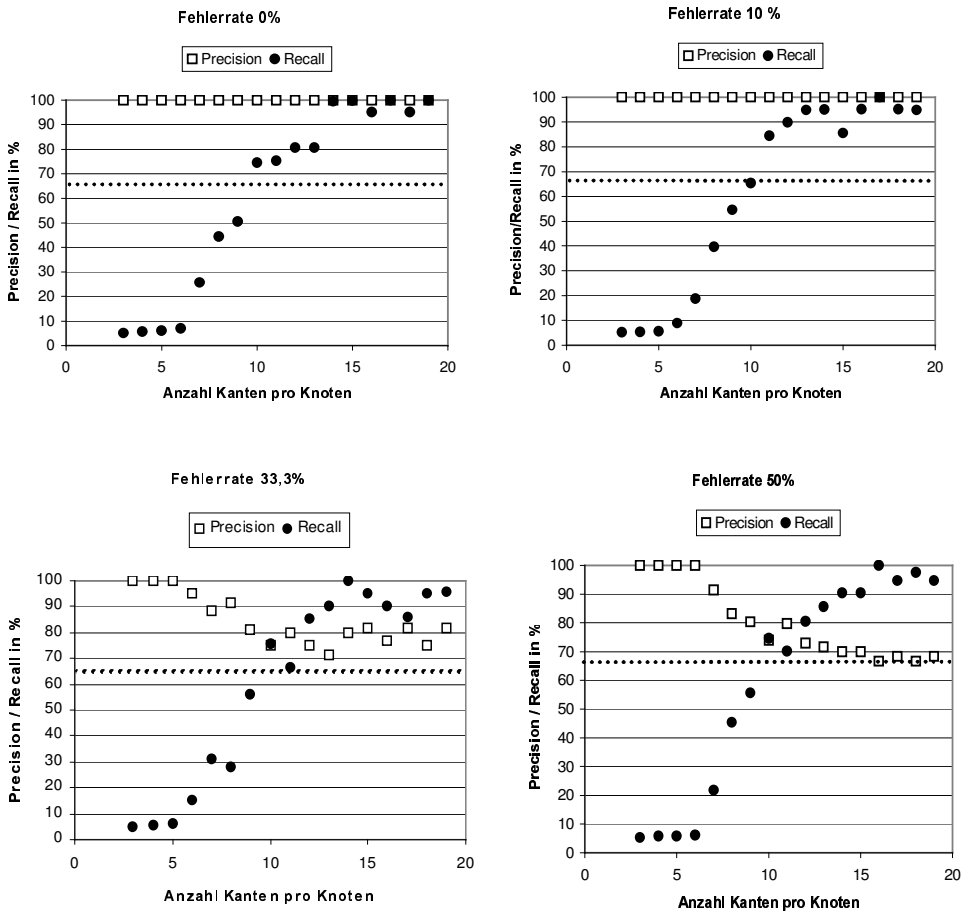


Abbildung 2.3.4-2: Precision und Recall in Abhängigkeit von der durchschnittlichen Anzahl Kanten pro Knoten. Meßpunkte sind jeweils Mittelwerte aus 20 Versuchen. Verifikation erfolgte bei ab 10% und mindestens zwei positiver Belegstellen

Während sich der Phasenübergang beim Recall genauso darstellt wie beim Versuch ohne Füllwörter, wird deutlich, daß schon ab Fehlerraten von 33% viele Füllwörter in die Klassen A und B eingeordnet werden und in diesem Aufbau Fehlerraten von 50% fast mit Sicherheit dafür sorgen, daß die Füllwörter nicht von den Zielitems unterschieden werden können. Bei

-

Fehlerraten unter 25% jedoch erweist sich das Verfahren als robust, wie Abbildung 2.3.4-3 zeigt.

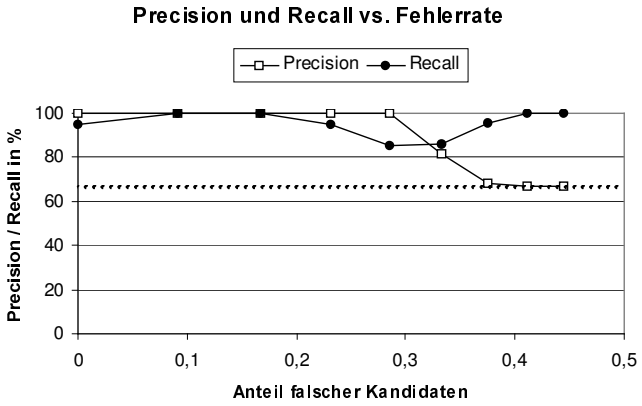


Abbildung 2.3.4-3: Precision und Recall bei fester Anzahl Kanten pro Knoten (Anzahl = 16) in Abhängigkeit von der Fehlerrate. Die gepunktete Linie gibt die Baseline für die Precision an.

Ein Rückschluß auf die Beschaffenheit von Relationen, die in natürlicher Sprache gesucht werden, ist nur begrenzt zulässig. Die kritische Fehlerrate ist abhängig von der Gesamtzahl und von der Anzahl der verschiedenen Füllwörter, die hier im Verhältnis zum natürlichsprachlichen Fall klein war. Mehr Füllwörter bedeuten eine kleinere Wahrscheinlichkeit für die Akzeptanz von falschen Items beim Verifikationsschritt.
Desweiteren verwendeten die Vorversuche keine Features, mit Hilfe derer viele Füllwörter von vornherein ausgeschlossen werden können. Die Fehlerrate gibt in diesem Fall die Rate von falschen Klassifizierungen ohne Verifikationsschritt an.

Mit dem Verfahren können Teilwörter von Elementen semantischer Relationen mit hohem Recall aus dem Korpus extrahiert werden, wenn die durchschnittliche Anzahl eines Teilwortes an einer Position in der Relation größer als 15 ist und der Anteil der Wörter, die durch die verwendeten

-

Regeln fälschlicherweise klassifiziert werden, nicht mehr als ein Viertel beträgt.

2.4 Regeln lernen

In der natürlichen Sprache wird eine semantische Relation häufig nicht nur durch einen einzigen Pattern erkannt, sondern manifestiert sich in verschiedenen Abfolgen von Wörtern und Wortarten, denen jeweils ihr eigenes Regelmuster zugrunde liegt. Zum Beispiel kann die Relation PERSONENNAME in folgenden Varianten aufteten:

- PERSONENNAME1$\subseteq$VORNAME$\times$NACHNAME
- PERSONENNAME2$\subseteq$VORNAME$\times$VORNAME$\times$NACHNAME
- PERSONENNAME3$\subseteq$VORNAME$\times$ADELSBEZ$\times$NACHNAME
- ...

Der Pendel-Algorithmus kann nur Instanzen der Unterrelation erkennen, wenn er über eine Regel mit dem entsprechendem Pattern verfügt. Da es einerseits mühsam ist, die verschiedenen Regeln pro Relation per Hand aufzustellen, andererseits nicht klar ist, inwieweit diese Regelmenge die Instanzen der Relation im Korpus abdecken, bietet sich ein automatisches Verfahren zum Regellernen an.

Im Gegensatz zum Regellernen in etwa der Induktiven Logischen Programmierung (vgl. [Dzeroski 2001]), wo Regeln mit 100% Precision induziert werden, werden hier unscharfe Pattern gesucht, für die eine Genauigkeit von etwa 70% ausreichend erscheint und die in ihrer Gesamtheit hohen Recall erreichen. Deswegen wurden an dieser Stelle keine bekannten Verfahren eingesetzt, sondern der triviale Ansatz verfolgt. Das heißt: Alle möglichen Regeln werden generiert und auf ihre Güte getestet.

Voraussetzung für das Regellernen ist ein genügend großer, wenn möglich vollständig annotierter Trainingstext, in dem viele Instanzen der gesuchten Relation enthalten sind.

2.4.1 Konstruktionsschritt

Pro verschiedener Klasse, die in der gesuchten Relation enthalten ist, wird ein Konstruktionsschritt durchgeführt. Im oben genannten Beispiel für die Relation PERSONENNAME wären dies drei Schritte, einmal für die Klasse VORNAME, einmal für ADELSBEZ (mit den Elementen "von", "van", "di" etc.) und einmal für NACHNAME.

Nacheinander werden alle Wörter der jeweiligen Klasse im Text gesucht und es werden alle möglichen Pattern eines gewissen Längenintervalls um das bekannte Vorkommen herum konstruiert. Abbildung 2.4.1-1 zeigt einen Beispieltext und alle Regeln der Längen zwei bis drei, die an dieser Stelle in die Menge der möglichen Regeln aufgenommen werden.

Text:	"John	Roth	und	**Frank**	Dunn	stehen	während ..."
Anno:	{GR}	{GR,NN}	{KL}	{GR,**VN**}	{GR,NN}	{KL}	{KL}

Konstruierte Regeln der Längen 2 und 3:

		KL	GR*			-> VN
			GR*	GR		-> VN
			GR*	NN		-> VN
	GR	KL	GR*			-> VN
	NN	KL	GR*			-> VN
		KL	GR*	GR		-> VN
		KL	GR*	NN		-> VN
			GR*	GR	KL	-> VN
			GR*	NN	KL	-> VN

Abb. 2.4.1-1 Konstruktion der Menge der möglichen Regeln

-

Zu beachten ist hierbei, daß die Zielklasse (im Beispiel: VN) nicht an der mit einem * versehenen Zielposition auftaucht.

Für den gesamten Text ergibt das eine große Menge an potentiellen Regeln, wovon die meisten jedoch ungeeignet für den Pendel-Algoithmus sind, da zu allgemein, z.B. die Regel `GR* GR` → `VN`, die praktisch alles erwischt und für Überlauf sorgen würde. Einschränkungen auf die Patternkonstruktion sind möglich, beispielsweise kann gefordert werden, daß im Pattern mindestens ein Tag der Mengen der gesuchten Relation enthalten ist.

2.4.2 Verifikationsschritt

Die so gefundenen Pattern werden im Folgenden anhand des Trainingstextes verifiziert. Die Regeln werden gegen den Text gematcht und es wird gezählt, wie oft jede Regel richtige und falsche Aussagen über ihr jeweiliges Zielwort machte. Liegt das Verhältnis dieser beiden Werte über einem gewissen Schwellenwert, so wird die Regel als geeignet bezeichnet und kann im anschließenden Pendelprozess verwendet werden.

3 Ergebnisse

In diesem Kapitel werden nach einem Überblick über Arbeiten zur Namensextraktion Experimente, die mit einer proprietären Implementierung des Pendel-Algorithmus durchgeführt wurden, im Einzelnen vorgestellt. Die meisten Versuche beziehen sich auf Personennamen.

3.1 Personennamen

3.1.1 Named Entity Extraction (NEE)

Named Entity Extraction (NEE) stellt eine der Hauptaufgaben der *Information Extraction* (IE) dar, ein Feld der Automatischen Sprachverarbeitung, das insbesondere durch die *Message Understanding Conferences* (MUCs, [Grishman 1996]) geprägt wurde. Die bisherige Forschung konzentrierte sich auf Erkennen und Tagging von Orts-Firmen- und Personennamen, sowie von numerischen Ausdrücken und Zeitangaben in englischsprachigem Text.

In den letzten Jahren wurden mehrere IE-Systeme entwickelt, genannt seien das *TextPro-System* ([Grishman 1995], [Appelt 1999]), das eine Weiterentwicklung des auf endlichen Übergangsautomaten beruhenden *FAUSTUS-Systems* ([Appelt 1993]) darstellt, sowie das *Alembic-System* ([Aberdeen 1995]) der New York University. Weitere Ansätze benutzen *Hidden Markov Models* (siehe z.B. [Rössler 2002]), oder *Maximum Enthropy Models*, wie MENE von Borthwick ([Borthwick 1999]). Eine relativ neue Verfahrensweise ist die Erkennung mit Hilfe von *Support Vector Machines* (SVM), eine effektive Methode ist in [Hirao 2002] beschrieben.

Sämtliche hier genannten Verfahrensweisen sind statistischer Natur.

Aus den in Frage kommenden Wörtern werden eine Vielzahl von Features extrahiert, die sowohl flache Eigenschaften wie Groß- und Kleinschreibung beinhalten, als auch POS-Tags, Buchstabenreihenfolge sowie Features, die sich aus Listen von bekannten Items ableiten lassen. Aufgrund dieser Features werden Tags zugewiesen. Zugrunde liegt jeweils ein vollständig getaggter Trainingskorpus, der die Datenbasis für die statistischen Klassifizierer bildet, die mit verfahrensspezifischen maschinellen Lernverfahren trainiert und anschließend auf den Zieltext angesetzt werden.

-

Die angesprochenen Verfahren benutzen folgende drei linguistischen Ressourcen als Grundwissen zum Erkennen von Namen [Poibeau 2001]:

- Listen von Markern, sogenannten *Trigger-Wörtern*, z.B. "Herr", oder "Minister"
- *Gazetteers*: Umfangreiche Listen von bekannten Namen
- generelle *Wörterbücher der Zielsprache*, um unbekannte Wörter zu identifizieren

Die bei MUC-7 teilnehmenden IE-Systeme erreichen beim Tagging der Klassen *Person, Ort, Organisation, Datum, Zeitangabe, Prozentangabe, Geldwert* und *Sonstig* für das Englische F-Werte von 0.82 bis 0.93 (siehe [Borthwick 99]), was menschlichen Durchschnittswerten von 0.96 sehr nahe kommt.

Generell ist anzumerken, daß das Erkennen von Namen im Deutschen schwieriger ist als in anderen Sprachen, da im Deutschen Substantive großgeschrieben werden und somit das verläßlichste Anzeichen für die Namenseigenschaft eines Wortes im Gegensatz zu anderen Sprachen wegfällt.

Der hier vorliegende Ansatz ist im Kern weniger ein System, das unstrukturierte Texte mit Tags versehen soll, sondern er dient zur Erstellung von Gazetteers sowie zur Wissensextraktion aus großen Korpora.

3.1.2 Vornamen und Nachnamen

In diesem Abschnitt soll ein Versuch mit dem Pendel-Programm (siehe Appendix B) beschrieben werden. Als Grundwissen wurden 155 Vornamen, 400 Nachnamen und 9 Titel bereitgestellt.
Gesucht wurde nach der schon erwähnten Relation PERSONENNAME.

-

Verwendet wurden zehn manuell aufgestellte Regeln, die in Tabelle 3.1.2-1 aufgelistet sind:

GR* NN → VN	Tagset:
SZ VN SZ GR* SZ → VN	SZ=Satzzeichen
TIT SZ GR* NN → VN	KL=Kleingeschrieben,
VN NN SZ GR* NN → VN	GR=Großgeschrieben
VN GR* SZ → NN	VN=Vorname
VN GR* KL → NN	NN=Nachname
TIT GR* KL → NN	TIT=Titel.
SZ NN SZ GR* SZ → NN	
TIT SZ GR* KL → NN	
NN SZ VN GR* SZ → NN	

Tabelle 3.1.2-1 manuell erstellte Regeln für Vor- und Nachnamen

Als Startitems wurden nur fünf der bekannten Vornamen verwendet, um mehrere Schritte zu erzwingen und das Verhalten des Algorithmus besser studieren zu können.

Im Suchschritt wurden jeweils maximal 255 Sätze aus dem deutschen Hauptkorpus entnommen, im Verifikationsschritt jeweils maximal 30 Sätze, ein Item wurde akzeptiert, wenn der Anteil bestätigender Vorkommen im Verifikationsschritt über 0.1 und die Anzahl dieser Vorkommen mindestens zwei betrug.

Der Algorithmus terminierte nach 18 Schritten und lieferte insgesamt 13866 Items, wovon 1553 als Vornamen und 12313 als Nachnamen klassifiziert waren. Die Precision für Nachnamen lag bei über 99%, die Precision für die Vornamen bei 80%. Grund für die niedrigere Precision bei den Vornamen war, daß viele Titel und Berufsbezeichnungen fälschlicherweise durch die Regeln als Vornamen klassifiziert wurden. Eingedenk der Tatsache, daß Vornamen normalerweise zwischen 4 und 10

-

Buchstaben lang sind, konnte die Precision für Vornamen nach einem entsprechenden Filterprozeß auf 88% erhöht werden.
Weiterhin fiel bei dem Versuch auf, daß die das TIT-Tag enthaltenden Regeln aufgrund der Seltenheit der verwendeten Titeln nur sehr selten zuschlugen.

Abbildung 3.1.2-2 zeigt die Anzahl der Items pro Schritt und die Anzahl der neuen Items pro Schritt. Das Verhalten entspricht der theoretischen Voraussage des logistischen Wachstums, die Wachstumsphase und die Sättigungsphase gehen im siebten Schritt ineinander über.

Versuche mit dem italienischen und dem englischen Korpus zeigten ein ähnliches Bild. Hier macht sich die Sprachunabhängigkeit des Verfahrens bemerkbar, die so weit geht, daß mit dem für das Deutsche erstellte Grundwissen und den deutschen Regeln vergleichbare Ergebnisse in anderen Sprachen erzielt wurden. Die Fehlerraten waren aufgrund der Kleinschreibung für Substantive sogar geringer (Precision im Englischen: 92,6% ohne Längenfilter für Vornamen, Nachnamen über 99%).

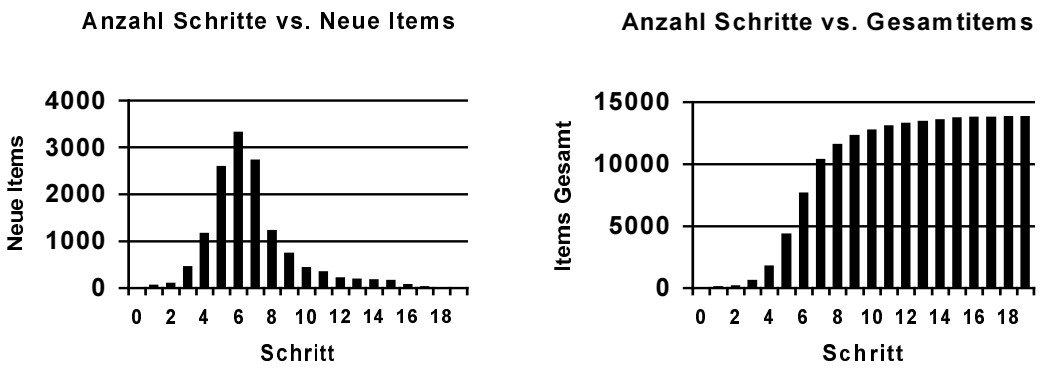


Abbildung 3.2.1-2: Items pro Schritt

Auffallend war weiterhin, daß der Algorithmus aufgrund des deutschen Grundwissens zunächst nur Items aus der Subgruppe der deutschen Namen innerhalb der englischen (bzw. italienischen) Texte fand, und erst nach und nach zu englischen (bzw. italienischen) Namen überging.

-

Der Französische Korpus erwies sich mit 0,86 Mio. Sätzen als zu klein, da die meisten Namen zu selten vorkamen und das Verfahren vorzeitig konvergierte.

3.1.3 Vornamen, Nachnamen und Titel

In den Folgeversuchen wurde die Stelligkeit der Personennamen-Relation auf drei erweitert. Gesucht wurden Elemente der Menge TIT (Titel und Berufsbezeichnungen), sowie weiterhin Vor- und Nachnamen.

Der Versuchsaufbau wurde leicht modifiziert: Als Grundwissen und gleichzeitig als Startitems wurden lediglich 19 häufige Vor- und Nachnamen benutzt*: Schmidt, Reuter, Wagner, Schäuble, Vogts, Hoffmann, Schulz, Möller, Maier, Beck, Michael, Thomas, Klaus, Wolfgang, Hans, Werner, Martin, Walter* und *Karl*. Die Akzeptanzschwelle im Verifikationsschritt wurde auf 0,09 herabgesetzt, es wurden zwölf reguläre Ausdrücke (wie z.B. *inister, *räsident) zum Taggen von Titeln verwendet und es wurde der Längenfilter für Vornamen implementiert. Alle anderen Parameter blieben unberührt und die vollständig erkannten Personennamen wurden in eine Datei geschrieben.

Die Regeln wurden im ersten Versuch manuell erstellt, im zweiten Versuch automatisch gelernt.

3.1.3.1 Versuch mit manuellen Pattern

Die verwendeten Regeln lassen sich aus Tabelle 3.1.3.1-1 ablesen:

GR* NN → VN	DET GR* VN NN → TIT
KOM VN KOM GR* → VN	GR* VN VN NN → TIT
TIT PU GR* NN → VN	DET GR* NN → TIT
VN NN KOM GR* NN → VN	GR* PU VN NN → TIT
GR* SZ → NN	GR* PU NN → TIT
TIT GR* KL → NN	NN KOM DET GR* → TIT
TIT PU GR* KL → NN	NN KOM GR* → TIT
NN KOM VN GR* SZ → NN	

Tabelle 3.1.3.1-1: manuell erstellte Regeln für Titel, Vor- und Nachnamen, zusätzliche Tags: KOM=Komma, PU=Punkt

Trotz eines deutlich kleineren Grundwissens setzte der Pendelprozeß ein und fand in den ersten sieben Schritten über 42000 Items, davon 15,0% Vornamen, 71,0% Nachnamen und 11,0% Titel. Der Versuch verlief erneut wie theoretisch vorhergesagt, ersichtlich in Abbildung 3.1.3.1-2.

Die Precision für Nachnamen und Titel erreichte wiederum über 99%, bei den Vornamen fiel die Precision während des Versuchsverlaufes leicht ab, was Abbildung 3.1.3.1-3 verdeutlicht: Die Precisionraten für Vornamen sind gegen die Anzahl von Items auftgetragen. Die Berechnung erfolgte jeweils für Itemintervalle von 1000.

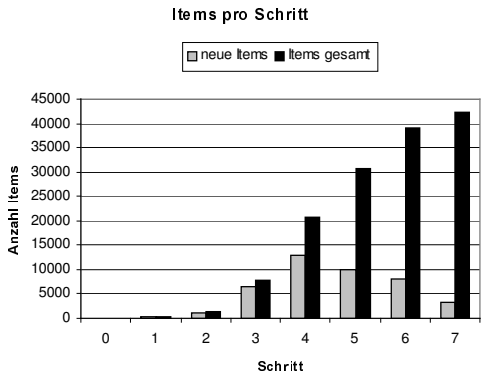


Abbildung 3.1.3.1-2 Items pro Schritt

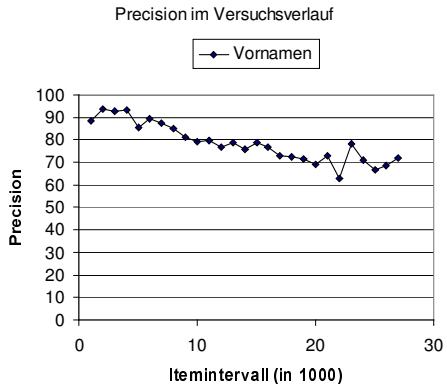


Abbildung 3.1.3.1-3: Precision der Vornamen im Versuchsverlauf

3.1.3.2 Versuch mit automatisch gelernten Regeln

Als Trainingstext für das Regellernen wurden 236 Sätze aus der Datenbank extrahiert, die jeweils mindestens einen Vor- und Nachnamen aus einer Menge von 50 häufigen Vornamen und 50 häufigen Nachnamen enthalten. Der Text wurde anschließend mit den Namen aus dem im Kapitel 3.1.2 beschriebenen Versuch annotiert und Regeln mit Patternlängen zwischen zwei und vier wurden gemäß Kapitel 2.4 inferiert,

-

wobei ein Pattern als geeignet angesehen wurde, wenn das Verhältnis von korrekten Matches zu der Gesamtzahl seiner Matches über 0,7 betrug.
Die Gesamtzahl der Regeln betrug 177, wovon 106 auf Vornamen, 67 auf Nachnamen und 4 auf Titel entfielen, die geringe Anzahl der Regeln für Titel erklärt sich aus dem seltenen Vorkommen von Titeln im Trainingstext.
Die im Vergleich zum manuellen Vorgehen hohe Anzahl von induzierten Regeln zeigt die Notwendigkeit eines automatischen Lernverfahrens an dieser Stelle auf: Viele der automatisch inferierten Regeln waren überraschend, wie z.B. `VN KL GR* NN` → `VN` für ein Textfragment wie "Thomas und Heinrich Mann". An diesem Pattern wird deutlich, daß für häufige Wörter wie Konjunktionen ein eigenes Tag günstig wäre, um das allgemeine Tag "`KL`" mit diesem zu ersetzen, um präzisere Regeln zu erhalten.
Für eine komplette Liste der automatisch inferierten Pattern, siehe Appendix A.
Im Unterschied zu dem Versuch mit manuell aufgestellten Regeln werden mehr neue Items pro behandeltem Item gefunden, im ersten Schritt 890, im zweiten Schritt 13241 Items, was für höheren Recall spricht. Der Versuch wurde im dritten Schritt abgebrochen. Abbildung 3.1.3.2-1 zeigt die Fehlerraten in Intervallen von je 1000 Items, die nacheinander gelernt

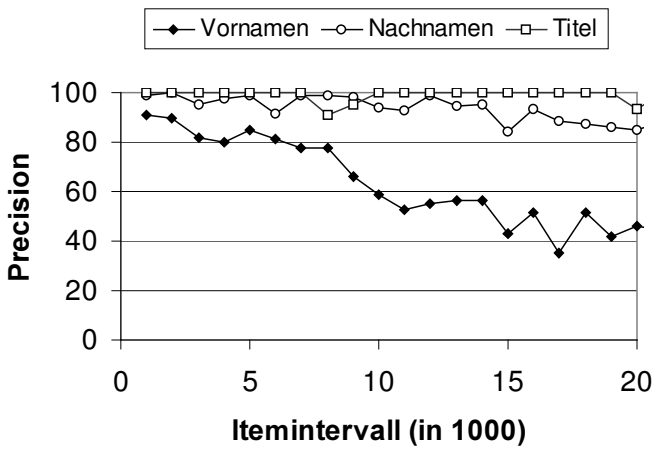


Abbildung 3.1.3.2-1: Precision im Versuchsverlauf für die Zielklassen

-

wurden. Trotz der hohen Fehlerraten für Vornamen erzielte der Versuch eine Gesamtprecision von 85,8%.

Zunächst fällt die Precision für Vornamen ab, anschließend aufgrund der Fehlerfortpflanzung die Precision für Nachnamen, während sie für die wenigen extrahierten Titel konstant hoch blieb. Die Verteilung der Gesamtitems auf die Klassen war: VN: 24,6%, NN: 74,0%, TIT: 1,4%.

3.1.3.3 Diskussion der typischen Fehler

Beim genaueren Untersuchen von Fehleritems der Vornamen-Klasse konnten vier Fehlerklassen unterschieden werden:

1. Titel/ Berufsbezeichnungen werden fälschlicherweise Vornamen, zum Beispiel bei "Bundeskanzler Kohl"
2. Wörter, die vor Nachnamen stehen, wie *Ära, Soko*, ... werden Vornamen, zum Beispiel bei "Ära Stresemann"
3. Firmenbezeichnungen werden falsch eingeordnet, zum Beispiel bei "Autohaus Müller"
4. Wörter, die u.a. Nachname sind, liefern falsche Vornamen, zum Beispiel bei "Sternbild Löwe", "Blauen Engels" und bei Songtiteln

Fehleritems der Klassen 1, 2 und teilweise der Klasse 3 liefern im nächsten Suchschritt keineswegs weitere Fehleritems, sondern völlig korrekte Nachnamen. So lieferte z.B. der falsche Vorname "Ära" unter anderem die Politiker *Hinrichs, Strauß, Bangemann, Albrecht, Gorbatschow* und *Jelzin*. In diesen Fällen erfolgt nicht nur eine automatische Fehlerkorrektur, die falschen Items unterstützen sogar noch den Pendelprozeß. Zur Fehlerfortpflanzung kommt es lediglich bei Firmenbezeichnern der Klasse 3 und bei Fehlern der Klasse 4.

Der Hauptanteil falscher Vornamen beim Versuch mit inferierten Regeln (siehe Kapitel 3.1.3.2) bestand aus großgeschriebenen englischen Wörtern jeder Art, wie z.B. *Let, Change, Rule, For, Secret*. Verantwortlich dafür ist

beispielsweise eine gewisse "Bibliothekarin Mary Love", deren Nachnamen in Songtiteln wie "Let Love Rule" vorkommt. Derartige Titel werden im Text meist in Anführungszeichen aufgeführt, so daß sie mit Hilfe eines Tagsets, das dieses Feature abbildet, ausgeschlossen werden können. Ein flexiblerer Ansatz wäre das Regellernen während des Pendelprozesses, siehe Kapitel 4.3.1.

Bevor im fünften Kapitel eine Anwendung zur Namensextraktion aus kurzen Texten vorgestellt wird, soll im Rest dieses Kapitels beleuchtet werden, wie sich das Verfahren bei anderen semantischen Relationen verhält.

3.2 Weitere Relationen

Im Gegensatz zu den Personennamen-Versuchen, wo die Parameter des Algorithmus genau abgestimmt waren und der zusätzliche Verarbeitungsschritt des Längenfilters implementiert wurde, um möglichst gute Ergebnisse zu erzielen, wurden in den folgenden Experimenten weder reguläre Ausdrücke zum Taggen von Komposita verwendet noch andere Anpassungen an das Problem getätigt. Die vorgestellten Relationen sollen exemplarisch die Arbeitsweise des Algorithmus in der Praxis dokumentieren. Anhand von auftretenden Fehlern werden Schwächen diskutiert.

3.2.1 Inseln

Um eine Relation, deren charakteristischer Graph ungleichmächtige Partitionen besitzt, zu untersuchen, wurden Versuche für die Relation INSEL⊆INSELBEZEICHNER×INSELNAME unternommen, bei welcher wenigen Inselbezeichnern viele Inselnamen gegenüberstehen.
Es wurden vier Regeln verwendet:

IBEZ GR* KL → INAM	IBEZ GR* SZ → INAM
GR* INAM KL → IBEZ	GR* INAM SZ → IBEZ

Tabelle 3.2.1-1: Regeln für INSEL, Tags: INAM: Inselname, IBEZ: Inselbezeichner

Das einzige Grundwissensitem war "Insel = IBEZ".

Es wurden zwei Insel-Versuche mit verschiedenen Parametereinstellungen durchgeführt. Es ist anzumerken, daß insbesondere die Einstellung der Akzeptanzschwelle und der Anzahl der Mindestbelegstellen für jede Relation individuell gewählt werden muß.

Die Akzeptanzschwelle beim ersten Insel-Versuch war zu hoch (0.15), weswegen der Versuch vorzeitig konvergierte. Abbildung 3.2.1-2 zeigt die gefundenen Items (schwarz), sowie einige zurückgewiesene Inselbezeichner (grau). Die Kanten zwischen den Wörtern verdeutlichen die Belegstellen.

In der Abbildung wird deutlich, daß die meisten Inselnamen direkt vom Startitem "Insel" geliefert wurden. Die durch den Bootstrapping-Prozeß gefundenen Kanten sind fett markiert, wie z.B. der Weg *Reichenbach-Bodenseeinsel-Mainau-Blumeninsel.*

In einem weiteren Versuch mit niedrigerer Akzeptanzschwelle (0.075) wurden 55 Inselbezeichner (davon korrekt: 45) und 176 Inselnamen (davon korrekt: 154) gefunden. Der Hauptverursacher für die Fehler bei den Inselnamen war das durch die Japanische Insel und Provinz *Okinawa* gelernte Item "Präfektur = IBEZ", und begünstigte die Klassifikation von 20 Japanischen Präfekturen (*Aichi, Iwate, Ywata, Aomori, Fukui, Fukuoka, Gifu, Haibei, Hyogo, Ibaraki, Ishikawa, Kagawa, Kanagawa, Kumamoto, Kyusu, Shizuoka , Shimane, Tochigi, Yamaguchi, Yamanashi*) und dem französischen Department *Gironde* als Inselnamen.

-

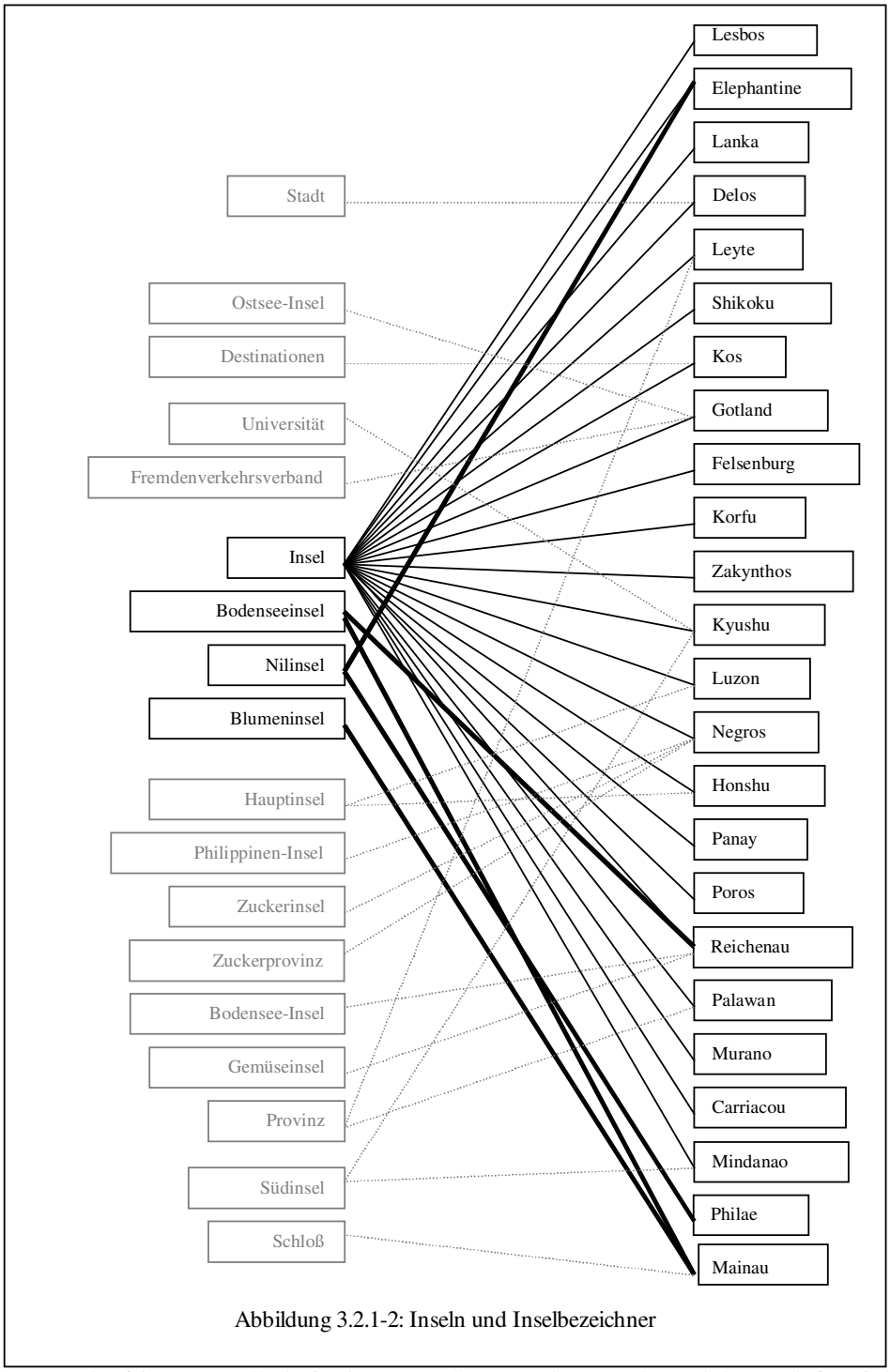


Abbildung 3.2.1-2: Inseln und Inselbezeichner

Der dürftige Recall bei den Insel-Versuchen rührt daher, daß viele Inselnamen, vor allem die der bekannteren Inseln, meist ohne Inselbezeichner im Text vorkommen. So wurden z.B. die Inseln *Kreta* und *Irland* zwar in die Kandidatenmenge aufgenommen, in den Beispielsätzen zur Verifizierung befanden sich aber zu wenig bzw. keine Inselbezeichner.

-

3.2.2 Städte und Ortsadjektive

Anhand der dreistelligen Relation ORT⊆LOC×HH×STA, wobei LOC Geographieadjektive, HH Städtebezeichner und STA Städtenamen beinhaltet, z.B. manifestiert in "südfranzösische Hafenstadt Marseille", soll veranschaulicht werden, wie sehr unterschiedliche Regeln die Ergebnismenge des Pendel-Algorithmus beeinflussen.

Im ersten Ort-Versuch wurden Regeln mit Pattern der Länge drei verwendet, im zweiten Ort-Versuch Pattern der Länge zwei. Im dritten Versuch wurde nur nach den Klassen HH und STA ohne Berücksichtigung der Kontexte gesucht, im vierten Versuch wurden die Kontexte mit einbezogen (siehe Tabelle 3.2.2-1).

erster Versuch: **Patternlänge 3**	**zweiter Versuch:** **Patternlänge 2**	**dritter Versuch:** **nur HH und STA**	**vierter Versuch:** **HH und STA mit Kontext**
LOC GR* STA → HH	KL* HH → LOC	GR* STA → HH	NGR GR* STA NGR → HH
KL* HH STA → LOC	LOC GR* → HH	HH GR* → STA	NGR HH GR* NGR → STA
LOC HH GR* → STA	GR* STA → HH		
	HH GR* → STA		

Tabelle 3.2.2-1: Regelmengen für die verschiedenen Ort-Versuche,
NGR = "nicht großgeschrieben"

Als Startitems dienten jeweils die vier Städtebezeichner *Hafenstadt, Hauptstadt, Stadt* und *Kleinstadt*, desweiteren wurden 50 Städtenamen als Grundwissen verwendet.

Während der erste Versuch nach Auffinden von insgesamt 371 Items mit hoher Precision vorzeitig konvergierte, fiel die Precision im zweiten Versuch schnell unter 5%, der Versuch divergierte und wurde abgebrochen.

Die Regeln im zweiten Versuch sind zu generell, so daß sich der Algorithmus frei über alle passenden Textstellen bewegte und nicht mehr von der Extraktion einer semantischen Relation gesprochen werden kann.

Der dritte Versuch brachte 1886 Items, davon 615 Städtebezeichner (Precision: 59,6%) und 1271 Städtenamen. Der Hauptanteil von

Fehleritems für die Klasse HH bestand aus Teilen von Zweiwort-Städtenamen, denen die Regeln nicht Rechnung tragen. Deswegen wurde ein vierter Versuch durchgeführt, bei dem die Kontexte derartig berücksichtigt werden, daß Zweiwort-Städtenamen mit Städtebezeichner von der Zielmenge ausgeschlossen wurden. In diesem Versuch wurden 1409 Items klassifiziert, davon 402 Städtebezeichner mit 76,3% Precision. Die Fehlerraten für die Klasse STA wurden nicht berechnet.

3.2.3 Rechte und linke Nachbarn von Namen

Wie schon in Kapitel 3.1.1 beschrieben, benutzen viele NEE-Verfahren Wörter aus dem Kontext. Um Wörter zu finden, die bevorzugt entweder vor oder nach Personennamen auftreten, wurde ein Personenname-Versuch (siehe Kapitel 3.1.2) durchgeführt, bei dem mit Hilfe von Zusatzregeln alle kleingeschriebenen Wörter, die links (Klasse PRAE) bzw. rechts (Klasse: SUFF) eines erkannten Personennamens auftraten, gefunden und verifiziert wurden.
Die Elemente der Klassen PRAE und SUFF unterstützten die Klassifikation von Namensitems in diesem Versuch nicht.

Es ergab sich folgende Aufteilung in Wortklassen:

Wortklasse	**Klasse PRAE (insg. 337)**	**Klasse SUFF (insg. 150)**
Adjektive/ Adverbien	78,6%	20,7%
davon ortsangebende	16,9%	0%
Verben	16,6%	74,6%
davon des Äußerns	75,0%	57,1%
kleingeschriebene Namen	3,3%	2,7%
Sonstige	1,5%	2,0%
Summe	100%	100%

Tabelle 3.2.3-1 Aufteilung von Namensnachbarn in Wortklassen

-

Auf diese Weise können Gazetteers von Wörtern gelernt werden, welche die Klassifikation der eigentlich gesuchten Wörter (hier: Namenselemente) unterstützen.
Desweiteren wird deutlich, daß anhand von Personennamen noch mehr semantische Relationen (wie z.B. eine Relation, die Äußerungsverben als Klasse beinhaltet) extrahiert werden können.

4 Ausblicke und Verbesserungen

In diesem Kapitel soll auf weitere Anwendungsgebiete und Verbesserungen des Pendel-Algorithmus eingegangen werden. Zunächst werden einige weitere Relationen theoretisch besprochen, anschließend wird eine Anwendung des Verfahrens zur Unterstützung von automatischer Übersetzung diskutiert, letztendlich folgen einige Möglichkeiten, das Verfahren im Allgemeinen zu verbessern.

4.1 Weitere Relationen

Wie im Kapitel 3.2 deutlich wird, eignet sich das Pendel-Verfahren für eine Vielzahl von semantischen Relationen, wobei manche Relationen mehr, manche weniger geeignet sind. Im folgenden werden einige Relationen kurz besprochen, für die der Pendel-Algorithmus geeignet sein sollte.

Für die Relation ÜBERNAHME$\subseteq$FIRMA1$\times$KAUFVERB$\times$FIRMA2, z.B. "AOL übernimmt Netscape" kann wegen des Verifikationsschrittes eine höhere Precision als in [Duclaye 2002] erwartet werden, wo die Ergebnismengen nach einigen Schritten nur noch wenig mit der gesuchten Relation in Zusammenhang stehen.

Desweiteren sollte sich das Verfahren mit leichter Adaption für die Annotation von Adjektiven mit funktionaler Information eignen, wie in

[Bohnet 2002] beschrieben. Hier wurden Adjektiven fünf verschiedenen Klassen zugewiesen, durch welche die Reihenfolge mehrerer auf ein Nomen bezogener Adjektive festgelegt ist. Beispielsweise ist die Abfolge der Adjektive in "einige ehemalige Erstligisten verlassen freudig das Feld" korrekt, hingegen ist "ehemalige einige ..." falsch, somit muß die Klasse von "einige" niedriger sein als die von "ehemalige".
Notwendig hierzu wären anstatt auf eine Zielklasse bezogene Tags die Zuweisung einer Menge von möglichen Tags, die durch weitere Klassifikationen eingeschränkt wird.

Um nicht für jede Relation ein passendes Tagset erfinden zu müssen, scheint die Verwendung eines Standard-Tagsets mit dazugehörigem Parser (z.B. TNT, [Brants 2000]) sinnvoll. Im Zuge dessen wäre es erstrebsam, negative Tags (z.B. NICHTNAME) oder Alternativen von Tags in den Regeln zuzulassen, um die Anzahl der notwendigen Regeln bei gleicher Ausdrucksstärke gering zu halten.

4.2 Erstellung von Wörterbüchern für idiomatische Wendungen

Ein Schwachpunkt der automatischen Übersetzung von Texten in andere Sprachen sind Wendungen, die von einem guten Übersetzer nicht wörtlich, sondern idiomatisch übersetzt werden. Rein wörterbuchbasierte automatische Übersetzungen ähneln zu oft Passagen aus "English for Runaways" (Englisch für Fortgeschrittene, [Sellheim 2002]), wo beispielsweise die Übersetzung des Ausdrucks "Mein lieber Herr Gesangsverein!" mit "My dear Mr. singing association" angegeben würde. Deswegen benutzen heutige Übersetzungssysteme (z.B. Pangloss, [Brown 1996]) sogenannte *Translation Memories*, eine Datenbank mit Beispiel-übersetzungen, in der dem vorliegenden Satz ähnliche Sätze gesucht werden. Die Übersetzung wird unter Anpassung der Unterschiede aus der

-

Datenbank übernommen. Außerdem werden dort linguistische Kollokationen beachtet, weil man zum Beispiel nicht "starker Tee" mit "powerful tea" übersetzen darf.

Eine Anpassung des Pendel-Verfahrens zur Auffindung von Übersetzungen idiomatischer Wendungen könnte folgendermaßen ablaufen:

Gegeben einen bilingualen Korpus, in dem Sätze der ersten Sprache zusammen mit den Übersetzungen in der zweiten Sprache gespeichert sind, und ein Wörterbuch der beiden Sprachen, werden die in den Sätzen enthaltenen Wörter zunächst auf Grundformen reduziert.

Anschließend werden aus beiden Sätzen die Grundformen gestrichen, für die das Wörterbuch die entsprechenden Übersetzungen liefert. Dies wird im Suchschritt für einige Sätze durchgeführt, bis ein Satzpaar gefunden wird, in dem jeweils ein kleines Satzfragment in beiden Sätzen übrigbleibt. Dies stellt eine vermutete Wendung und deren Übersetzung dar. Im Verifikationsschritt wird nach Satzpaaren gesucht, die nach Grundformenreduktion und Streichung der bekannten Übersetzungen ebenfalls diese Wendung enthalten. Finden sich genügend Satzpaare dieser Art, wird die Wendung mit ihrer Übersetzung in das Wörterbuch aufgenommen.

Auf diese Weise lassen sich nicht nur Wendungen finden, sondern auch unvollständige Wörterbücher erweitern.

4.3 Verbesserung im Verfahren

Im verbleibenden Teil dieses Kapitels sollen generelle Verbesserungen des in dieser Arbeit beschriebenen Pendel-Verfahrens beschrieben werden.

4.3.1 Regellernen während des Pendel-Prozesses

In der vorliegenden Implementierung geschieht das Regellernen vor dem eigentlichen Pendelprozeß und benötigt einen Trainingstext, für den ein ungleich größerer Aufwand getrieben werden muß als zum Erstellen von

-

manuellen Regeln und den Startitems-Listen. Wie die Versuche in Kapitel 3 gezeigt haben, sind insbesondere die Regeln für die Konvergenz des Verfahrens verantwortlich.

In einer zukünftigen Version sollen aus diesem Grund Regeln auch

```
Lade Beispielitems
Lade Startregeln
Lade Grundwissen
StartItems newI:=Beispielitems
NeueRegeln newR:=leer
Wissen K:=Beispielitems+Grundwissen  //Grundwissen: Artikel etc.
verifizierte Regeln R:=Startregeln
noch nicht verifizierte Regeln: youngR:=leer
do {
    Items I:=newI
    newI:=leer
    for all i ∈ I  {
        text_i:=Hole Sätze aus Korpus, die i enthalten
        kandidaten:=Wende R auf text_i an          // Suchschritt Items
        for all k ∈ kandidaten {                   // Verifikationsschritt Items
            kandText:= Hole Sätze aus Korpus, die k enthalten
            rating_k:= Wende R auf kandText an und über-
                prüfe, wie oft k wie in text_i klassifiziert wird
            wenn rating_k hoch genug, füge k zu K und zu newI hinzu
        } // for all k
        Konstruiere um alle Vorkommen von i in text_i, die durch R nicht
            erfaßt wurden, Regeln und
            füge sie zu newR hinzu.  //Suchschritt Regeln
        for all r ∈ youngR {          // Verifikationsschritt Regeln
          Matche r gegen text_i, erhöhe rating von r, falls r bekannte Items liefert
        } // for all r
    Füge newR zu youngR hinzu
    newR:=leer
    } // for all i
    Füge für gut befundene Regeln aus youngR in R ein, entferne sie aus youngR
} while newI nicht leer.
```

Abbildung 4.3.1-1: Pendel-Algorithmus mit Regellernen während des Prozesses

während des Pendel-Prozesses gelernt werden. So wird ein Bootstrapping-Verfahren mit zwei Ebenen implementiert: Ausgehend von einigen Startitems und einfachen, manuell erstellten Regeln werden mit Hilfe der Regeln neue Items gesucht und mit Hilfe der Items neue Regeln konstruiert. Die neu konstruierten Regeln befinden sich zunächst in einer Testphase und werden erst zur Klassifikation von Items verwendet, wenn sie genügend häufig mit ausreichender Genauigkeit Items richtig klassifiziert haben. So würde sich der Regelfindungsprozeß wie in den Ort-Versuchen (Kapitel 3.2.2) erübrigen. Abbildung 4.3.1-1 zeigt eine Beschreibung des erweiterten Algorithmus.

Zu klären bleibt die Frage, ob lediglich positive Belegstellen von Regeln ausreichen (siehe Verifikationsschritt Regeln), oder ob das Regelrating durch das Liefern falscher (nicht unbekannter) Items, also Items, die bekannt sind, aber nicht der Klasse angehören, die die Regel liefert, vermindert werden muß. Davon hängt die Gütefunktion ab, aufgrund welcher Regeln von der Menge der noch zu verifizierenden Regeln in die der Regeln wandern, die zum Klassifizieren von Items verwendet wird.

In [Yangarber 2002] wird zum Zwecke des Erkennens negativer Belegstellen eine weitere Itemklasse für falsche Items eingeführt und ebenso wie die Zielklassen durch Bootstrapping erweitert. Außerdem erhalten hier auch die Items ein Rating.

4.3.2 Supervisor-Schritt zum Ablehnen von Items

Wie im zweiten Insel-Versuch (siehe Kapitel 3.2.1), so werden desöfteren durch wenige falsche Items zahlreiche neue falsche Items gefunden, wodurch die Fehlerrate steigt und die Ergebnisse in Rohform für mögliche Anwendungen unbrauchbar macht. Abhilfe könnte an dieser Stelle eine Abkehr vom rein unüberwachten Lernparadigma schaffen, und zwar folgendermaßen: Für jedes Item wird gespeichert, aufgrund welcher anderer Items es im Verifikationsschritt akzeptiert wurde. Wird nun zu

einem beliebigen Zeitpunkt (also während oder nach dem Programmablauf) ein Item durch die menschliche Überwachungsinstanz als falsch eingeordnet, werden die Akzeptanzraten der mit Hilfe dieses Items klassifizierten anderen Items neu berechnet. Fällt hierbei ein anderes Item unter die Akzeptanzschwelle, so wird es ebenfalls als falsch markiert, dieser Prozeß wird iteriert. Auf diese Weise kann die Qualität der aus dem Pendel-Verfahren gewonnenen Daten schnell verbessert werden.

5 Anwendung: Extraktion von Namen aus kurzen Texten

In diesem Kapitel soll eine Anwendung beschrieben werden, die aus der Wortschatz-Datenbank *wdtaktuell* (Artikel aus jeweils heutigen Zeitungen, ca. 20'000 Sätze) mit Hilfe des Hauptkorpus Personennamen extrahiert.

Der in *wdtaktuell* enthaltene Text ist zu kurz, um ihn als Korpus für den Pendelprozeß zu verwenden, da es nur jeweils wenige Belegstellen für die meisten Wörter gibt. Er ist jedoch kurz genug, um ihn vollständig verarbeiten zu können.

Die Verarbeitung erfolgt satzweise und in zwei Stufen. In der ersten Stufe wird der Satz annotiert und es werden Regeln zum Finden von Items angewendet (siehe Kapitel 3.1). Alle noch unbekannten Itemkandidaten werden in einem Verifikationsschritt an Belegstellen in *wdtaktuell* und im Hauptkorpus überprüft, hierbei werden eventuell neue Items gelernt. Durch die Größe des Hauptkorpus wird ein sinnvoller Verifikationsschritt möglich, durch Verwenden von *wdtaktuell* können Kandidaten überprüft werden, die im Hauptkorpus nicht enthalten sind.

In einem zweiten Schritt wird der Satz nochmals annotiert, wobei neu gelernte Items im Unterschied zur vorherigen Annotierung ihr entsprechendes Tag bekommen. Auf die Tagfolge wird ein zweiter Regelsatz angewendet, welcher nur die kompletten Personennamen mit

-

Berufsbezeichnung erkennt, z.B. Pattern wie `VN VN NN, TIT VN VN NN, TIT TIT NN`, während die Pattern der Regeln zur Itemsfindung meist nur Teile von Personnenamen mit Kontexten beinhalten.

Die extrahierten Personennamen werden in eine Datenbank gespeichert, und zwar mit und ohne Berufsbezeichung, um sie später im Text wiedererkennen zu können.

Das Tagset wurde um zwei Tags erweitert: ein Tag für Adelsbezeichner innerhalb von Namen wie *von, van, Freiherr, bin, ben, el*, zum anderen ein Tag für *Rejection Items*. Die Klasse der Rejection Items enthält häufig mißklassifizierte Wörter, z.B. *Ära, Gebrüder, Firma* usw (siehe Fehlerklasse 2 in Kapitel 3.1.3.3), die nicht als Namensteile erwünscht sind.

```
Lade regexps                          // Reg. Ausdrücke zum Annotieren
Lade ItemRegeln                       // Regeln zur Itemfindung
Lade PersonenRegeln                   // Pattern zu Erkennen von Personen
Lade Grundwissen                      // Artikel und viele Namen  und Titel

Wissen K:= Grundwissen
for all Sätze s aus wdtaktuell  {
    Annotiere s mit Hilfe von K und regexps
    kandidaten:=Wende ItemRegeln auf s an      // Suchschritt
    for all k ∈ kandidaten {                   // Verifikationsschritt
        kandText:= Hole Sätze aus
            Hauptkorpus und aus Wdtaktuell, die k enthalten
                rating_k:= Wende ItemRegeln auf kandText an und über-
             prüfe, wie oft k wie in s klassifiziert wird
        wenn rating_k hoch genug, füge k zu K hinzu
    } // for all k
    Annotiere s mit Hilfe von neuem K und regexps
    personen:= Wende PersonenRegeln auf s an
    Speichere personen in Datenbank
} // for all s
```

Abbildung 5-1: Algorithmus zum Extrahieren von Namen aus kurzen Texten

-

Um großen Recall zu erreichen, wurde ein großes Grundwissen (2008 Vornamen, 1417 Titel, 13637 Nachnamen, ferner Artikel und 31 Rejection Items) bereitgestellt.
Abbildung 5-1 zeigt den Algorithmus.

Innerhalb von 89 Tagen wurden auf diese Weise 56311 Personennamen von 35909 verschiedenen Personen gefunden (in 20402 Fällen gab es eine Berufsbezeichnung), was ca. 400 Peronennamen pro Tag entspricht. Die Namensteile im Einzelnen hatten eine Precision von 81,7% (Vornamen), 96,5% (Titeln) und 99,3 % (Nachnamen), was eine Gesamtprecision[2] von 96,7% ergibt.
Stichproben ergaben eine Precision von 97,5% für die Personennamen im Ganzen.
In einer Stichprobe von 1000 Sätzen wurde der Recall für Personennamen inklusive Berufsbezeichnung bei 63,6%, für Personennamen ohne Berufsbezeichnung bei 78,2% ermittelt, was F-Werte von 0,77 bzw. 0,868 ergibt.

Abbildung 5-2 zeigt einige Sätze aus den Texten vom 6. Juni 2002. Aufgrund des Grundwissens erkannte Namen sind fett markiert, neu gelernte Namensteile zusätzlich kursiv, nichtklassifizierte Namen sind unterstrichen, die Tags werden in eckigen Klammern nach dem Wort angegeben.

---

[2] Die Auszählung erfolgte per Hand, bei Nachnamen ist das Bewerten schwierig, da fast alle Wörter als Nachname möglich sind. Als falsch gewertet wurden daher nur die offensichtlichen Nichtnachnamen. Insgesamt wurden 1333 Vornamen, 8061 Nachnamen und 5701 Titel gefunden.

-

Diese Beweislast sollte umgedreht werden, sagte **Maren[VN] Geisler[NN]**, Bankenexpertin beim BdV. Die mangelnde Sicherheit beim Online-Kauf ist ein großes Problem mit steigenden Zahlen. [...]
Diese Nachlässigkeit ist für Betrüger die Eintrittskarte, sagte seinerzeit ***Experian-Chef[TIT]*** **Peter[VN]** ***Brooker[NN]***. [...]
Post-Chef **Klaus[VN]** Zumwinkel muss daher heute auf der Hauptversammlung mit Kritik rechnen. „Der Aktie fehlen die positiven Nachrichten", sagt **Sven[VN] Madsen[NN]** von der BHF-Bank, der das Papier mit „marketperform" bewertet. [...]
Konzeptkünstlerin **Maria[VN] Eichhorn[NN]**, die weder Interviews gibt noch ihre Vernissagen besucht, hat für die d11 eigens eine Aktiengesellschaft gegründet.

Abbildung 5-2: Ausschnitt aus der Tageszeitung vom 6. Juli 2002.
Sätze ohne Namen wurden weggelassen.

6 Zusammenfassung

Das in dieser Arbeit dargelegte Pendel-Verfahren verdeutlicht, wie Algorithmen mit marginaler Stützung auf linguistische Theorien Daten für die Sprachverarbeitung liefern können. Gefragt wurde hier nicht nach dem theoretischen Unterbau, sondern nach einer einfachen Möglichkeit, durch Ausnutzung eines großen Korpus große Datenmengen hoher Qualität anzusammeln.

Die durchgeführten Experimente verdeutlichen Stärken und Schwächen des Verfahrens. Als großer Vorteil ist die geringe Menge an Startinformation anzusehen, die für den Pendel-Prozeß nötig ist. Ein Schwachpunkt bildet sicherlich der große Annotierungsaufwand pro gelerntem Item, was sich bei den Versuchen in langen Laufzeiten bemerkbar machte, sowie die notwendige Korpusgröße.

Die Ausblicke im vierten Kapitel zeigen Weiterentwicklungen, die den Rahmen dieser Arbeit gesprengt hätten, aber noch mehr Anwendungsgebiete des Verfahrens ermöglichen, sowie die Menge des bereitgestellten Grundwissens vermindern helfen.

Durch die guten Ergebnisse bei der Namensextraktion wird deutlich, daß nichtstatistische Verfahren in der Lage sind, in der IE gute Dienste zu leisten, da diese auch seltene Wörter klassifizieren können, bei denen statistische Verfahren aufgrund weniger Belegstellen versagen.

-

Quellenverzeichnis

[Aberdeen 1995] Aberdeen, J., Burger, J. Day, D., Hirschman, L., Robinson, P. und Vilain, M. (1995): MITRE: Description of the Alembic System as Used for MUC-6, Proceedings of the Sixth Message Understanding Conference, Morgan Kaufmann Publishers, San Francisco

[Appelt 1993] Appelt, D., Hobbs, J., Bear, J., Israel, D., Kameyana, M. und Tyson, M. (1993): FAUSTUS: a finite state Processor for Information Extraction from Real-World Texts. In Proceedings of the International Joint Conference on Artificvial Intelligence (IJCAI'93)

[Appelt 1999] Appelt, D. und Martin, D. (1999): Named Entity Recognition in Speech: Approach and results using the TextPro-System, HUB, Proceesdings of the DARPA Broadcast News Workshop, Hernon, Virginia

[Bartschat 1996] Bartschat, B. (1996): Methoden der Sprachwissenschaft, Erich Schmidt Verlag, Berlin

[Bohnet 2002] Bohnet, B., Klatt, S., Wanner, L. (2002): A Bootstrapping Aproach to Automatic Annotation of Functional Information of Adjectives with an Application to German, Proceedings of the LREC-3 Workshop: Linguistic Knowledge Aquisition and Representation - Bootstrapping Annotated Language Data, Las Palmas, Spanien

[Borthwick 1999] Borthwick, A. (1999): A Maximum Enthropy Approach to Named Entity Recognition, PhD thesis, New York

[Brants 2000] Brants, T. (2002): TNT-A Statistical Part-of-Speech-Tagger, Universität des Saarlandes, Saarbrücken

[Brill 1992] Brill, E. (1992): A Simple Rule-Based Part Of Speech Tagger, in Proceedings of the Third Conference on Applied Computational Linguistics (ACL), Trento, Italien

-

[Brin 1998] Brin, S. (1998): Extracting Patterns and Relations from the World Wide Web, in Proceedings of the WebDB Workshop (EDBT-98), Valencia, Spanien

[Brown 1996] Brown, R.D. (1996): Example-Based Machine Translation in the Pangloss System, Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-96)

[Califf 1997] Califf, M. E., Mooney, R. J. (1997): Relational Learning of Pattern-Match Rules for Information Extraction, Proceedings of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing

[Church 1988] Church, K. W. (1988): A stochastic parts program and noun phrase parser for unrestricted text, Proceedings of the ANLP 2

[Collins 1999] Collins, M., Singer, Y. (1999): Unsupervised Models For Named Entity Classification. Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora

[Dempster 1977] Dempster, A., Laird, N., Rubin, D. (1977): Maximum likelihood from incomplete data via the EM algorithm. Journal of Royal Statistical Society 39

[Duclaye 2002] Duclaye, F., Yvon, F. und Collin, O. (2002): Using the Web as a Linguistic Resource for Learning Reformulations Automatically, Proceedings of the Third Conference on Language Resources and Evaluation (LREC), Las Palmas, Spanien

[Earley 1970] Earley, J. (1970): An efficient context-free parsing algorithm. Communications of the ACM, 13:94--102

[Francis 1964] Francis, W.N. and Kučera, H. (1964): A Manual of Information to Accompany a Standard Sample of Present-Day Edited American English, for Use with Digital Computers, Providence, RI: Linguistics Department, Brown University,

-

[Grishman 1995] Grishman, R. (1995): Where's the Syntax? The NYU MUC-6 System, Proceedings of the Sixth Message Understanding Conference, Morgan Kaufmann Publishers, San Francisco

[Grishman 1996] Grishman, R. und Sundheim, B. (1996): Message Understanding Conference-6: A Brief History. In Proceedings of the 16th International Conference on Computational Linguistics, Copenhagen, Denmark.

[Hirao 2002] Isozaki, H, Kazawa, H. (2002): Efficient Support Vector Classifiers for Named Entity Recognition, Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002), Morgan Kaufmann Publishers, San Francisco

[Malmkjaer 2002] Malmkjaer, K (Hrsg.) (2002): The Linguistics Enzyclopedia, Second Edition, Routledge, London

[Manning 1999] Manning, C. D., Schütze, H. (1999): Foundations of Statistical Natural Language Processing, Massachusetts Institute of Technology

[McEnery 1996] McEnery, T., Wilson, A. (1996): Corpus Linguistics, Edinburgh University Press

[Mel'čuk 1976] Mel'čuk, I.A. (1976): Towards a linguistic 'Meaning-Text Model', in Das Wort, Fink Verlag, München

[Mikheev 1999] Mikheev, A, Moens, M. and Grover, C. (1999): Named entity Recognition without Gazetteers, Proceedings of EACL'99

[Pause 2002] Pause, P.E. (2002): Einführung in die Semantik, Skript zur Vorlesung, Universität Konstanz

[Poibeau 2001] Poibeau, T., Kosseim, L. (2001): Proper Name Extraction from Non-Journalistic Texts, Proceedings of the Eleventh Meeting of Computational Linguistics in the Netherlands (CLIN)

[Quasthoff 1998] Quasthoff, U. (1998): Projekt Der Deutsche Wortschatz, in Heyer, G. / Wolff, C. (Hrsg.), Linguistik und neue Medien, Deutscher Universitätsverlag, Wiesbaden

-

[Quasthoff 2002a] Quasthoff, U., Wolff, C. (2002): Text-based Knowledge Aquisition for Onthology Engeneering, Submitted for Konvens 2002, Saarbrücken.

[Quasthoff 2002b] Quasthoff, U., Biemann, C., Wolff, C. (2002): Named entity learning and verification: EM in large corpora, Proceedings of the Sixth Conference on Natural Language Learning (CoNNL-2002), Morgan Kaufmann Publishers, San Francisco

[Riloff 1999] Riloff, E., Jones, R. (1999): Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. Proceedings of the sixteenth National Conference on Artificial Intelligence (AAAI-99)

[Rössler 2002] Rössler, M. (2002): Using Markov Models for Named entity recognition in German Newspapers, Proceedings of the ESSLLI'02 Workshop on Machine Learning Approaches in Computational Linguistics, Trento, Italien

[Schütze 1994] Schütze, H. , Singer, Y. (1994): Part-of-Speech tagging using a variable memory Markov Model, Proceedings of ACL-32

[Sellheim 2002] Sellheim, K.-H. (2002): English for Runaways. This is the perfect homework-trainer, you must not in the evening school, R.G. Fischer-Verlag

[Steele 1990] Steele, J. (Hrsg.) (1990): Meaning Text Theory, University of Ottawa Press, Canada

[Yangarber 2002] Yangarber, R., Lin, W., Grishman, R. (2002): Unsupervized Learning of Generalized Names, Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002), Taipei, Taiwan

[Zipf 1929] Zipf, G. K. (1929): Relative Frequency as a Determinant of Phonetic Change, Wiederabdruck in den Harvard Studies in Classical Philology, Volume XL

-

Appendix A: Inferierte Regeln

In folgender Tabelle sind die im Versuch im Kapitel 3.1.3.2 verwendeten Regeln aufgeführt, sortiert nach Zielklasse und Genauigkeit (acc) auf dem Trainingstext. Aus der Spalte r/ f ist zu entnehmen, wie oft die Regel im Trainingstext richtigerweise und falscherweise anwendbar war.

Verwendete Tags:

SZ	Satzzeichen	GR	groß am Wortanfang
PU	Punkt	KL	klein am Wortanfang
KOM	Komma	MIX	groß im Wort
NUM	Zahl	DET	Artikel
VN	Vorname	NN	Nachname
TIT	Berufsbezeichnung		

Regel	**r/f**	**acc**
KOM VN GR* KL → NN	19/ 0	1,00
VN VN GR* → NN	7/ 0	1,00
KOM VN GR* PU → NN	6/ 0	1,00
VN VN GR* SZ → NN	6/ 0	1,00
NN SZ GR* VN → NN	5/ 0	1,00
VN VN GR* KOM → NN	5/ 0	1,00
TIT GR GR* PU → NN	4/ 0	1,00
TIT VN GR* PU → NN	4/ 0	1,00
DET VN GR* SZ → NN	3/ 0	1,00
KL VN VN GR* → NN	3/ 0	1,00
DET VN GR* → NN	3/ 0	1,00
SZ VN GR* KL → NN	29/ 1	0,97
VN GR* SZ DET → NN	27/ 2	0,93
VN GR* PU GR → NN	23/ 2	0,92
GR VN GR* PU → NN	10/ 1	0,91
VN GR* KOM DET → NN	18/ 2	0,90
VN GR* PU → NN	25/ 3	0,89
PU VN GR* KL → NN	7/ 1	0,88
VN GR GR* SZ → NN	7/ 1	0,88
KOM VN GR* → NN	74/ 12	0,86
VN GR* KL GR → NN	59/ 10	0,86
VN GR* KL KL → NN	54/ 10	0,84
KL VN GR* KL → NN	51/ 10	0,84
KL TIT VN GR* → NN	15/ 3	0,83
KOM KL VN GR* → NN	15/ 3	0,83
VN GR GR* KOM → NN	5/ 1	0,83
VN KL VN GR* → NN	5/ 1	0,83
KOM VN GR* SZ → NN	54/ 12	0,82
VN GR* KL → NN	126/ 29	0,81
SZ VN GR* → NN	98/ 25	0,80
KL TIT GR GR* → NN	16/ 4	0,80
PU GR VN GR* → NN	8/ 2	0,80
KL VN GR* DET → NN	4/ 1	0,80
KOM VN GR* KOM → NN	38/ 10	0,79
TIT GR GR* SZ → NN	11/ 3	0,79

-

TIT VN GR* SZ → NN	11/ 3	0,79
SZ KL VN GR* → NN	18/ 5	0,78
NN KL VN GR* → NN	30/ 9	0,77
VN MIX PU GR* → NN	10/ 3	0,77
VN MIX SZ GR* → NN	10/ 3	0,77
PU VN GR* → NN	10/ 3	0,77
TIT VN GR* → NN	22/ 7	0,76
VN GR* SZ GR → NN	96/ 32	0,75
SZ VN GR* SZ → NN	66/ 22	0,75
SZ GR VN GR* → NN	12/ 4	0,75
KL VN GR* PU → NN	6/ 2	0,75
MIX SZ GR* SZ → NN	6/ 2	0,75
SZ KOM VN GR* → NN	6/ 2	0,75
SZ SZ VN GR* → NN	6/ 2	0,75
MIX PU GR* KOM → NN	3/ 1	0,75
TIT PU VN GR* → NN	3/ 1	0,75
TIT SZ VN GR* → NN	3/ 1	0,75
VN GR* MIX → NN	3/ 1	0,75
KL VN GR* → NN	100/ 35	0,74
VN GR* KL DET → NN	14/ 5	0,74
GR VN GR* KL → NN	40/ 15	0,73
KL VN GR* KOM → NN	35/ 13	0,73
NN KOM VN GR* → NN	24/ 9	0,73
TIT GR GR* → NN	24/ 9	0,73
VN GR* → NN	283/ 111	0,72
KL VN GR* SZ → NN	48/ 19	0,72
KOM GR GR* KOM → NN	38/ 15	0,72
NN SZ VN GR* → NN	31/ 12	0,72
DET GR VN GR* → NN	15/ 6	0,71
VN GR* DET GR → NN	5/ 2	0,71
SZ TIT GR GR* → NN	5/ 2	0,71
SZ TIT VN GR* → NN	5/ 2	0,71
KOM GR* VN GR → TIT	5/ 0	1,00
KOM GR* GR NN → TIT	3/ 0	1,00
KOM GR* DET MIX → TIT	3/ 1	0,75
SZ GR* DET MI → TIT	3/ 1	0,75
VN GR KOM GR* → VN	86/ 0	1,00
NN SZ GR* NN → VN	31/ 0	1,00
GR* NN SZ DET → VN	27/ 0	1,00
NN KOM GR* NN → VN	24/ 0	1,00
KOM GR* NN KL → VN	19/ 0	1,00
GR* NN KOM DET → VN	18/ 0	1,00
GR GR* NN KOM → VN	17/ 0	1,00
TIT GR* GR SZ → VN	14/ 0	1,00
TIT GR* NN SZ → VN	11/ 0	1,00
TIT GR* GR KOM → VN	9/ 0	1,00
VN GR* NN → VN	8/ 0	1,00
VN GR* NN SZ → VN	7/ 0	1,00
SZ TIT GR* GR → VN	7/ 0	1,00
KL GR* NN PU → VN	6/ 0	1,00
KOM GR* NN PU → VN	6/ 0	1,00
TIT GR* NN KOM → VN	6/ 0	1,00
GR* NN DET GR → VN	5/ 0	1,00
VN GR* NN KOM → VN	5/ 0	1,00
KOM TIT GR* GR → VN	5/ 0	1,00
SZ TIT GR* NN → VN	5/ 0	1,00
VN KL GR* NN → VN	5/ 0	1,00
GR* NN DET → VN	5/ 0	1,00
GR* NN SZ NUM → VN	4/ 0	1,00
TIT GR* GR PU → VN	4/ 0	1,00
TIT GR* NN PU → VN	4/ 0	1,00
NN GR SZ GR* → VN	4/ 0	1,00
KL VN GR* NN → VN	3/ 0	1,00
PU GR* NN KOM → VN	3/ 0	1,00
PU GR* NN SZ → VN	3/ 0	1,00
KL VN GR* NN → VN	3/ 0	1,00
KOM GR GR* NN → VN	3/ 0	1,00
KOM TIT GR* NN → VN	3/ 0	1,00
TIT PU GR* NN → VN	3/ 0	1,00
TIT SZ GR* NN → VN	3/ 0	1,00

DET KL TIT GR* → VN	3/ 0	1,00
KOM TIT PU GR* → VN	3/ 0	1,00
GR* NN MIX → VN	3/ 0	1,00
GR GR* NN SZ → VN	31/ 1	0,97
KL GR GR* NN → VN	48/ 2	0,96
GR* NN SZ KL → VN	43/ 2	0,96
GR* NN PU → VN	25/ 1	0,96
KOM GR* NN → VN	74/ 4	0,95
GR* NN KOM KL → VN	37/ 2	0,95
NN KL GR* NN → VN	30/ 2	0,94
DET GR GR* NN → VN	15/ 1	0,94
KL TIT GR* NN → VN	15/ 1	0,94
SZ GR* NN → VN	98/ 7	0,93
GR* NN KOM → VN	97/ 7	0,93
KOM GR* NN SZ → VN	54/ 4	0,93
GR* NN KL GR → VN	59/ 5	0,92
GR* NN KOM GR → VN	56/ 5	0,92
TIT GR* NN → VN	22/ 2	0,92
GR* NN SZ → VN	149/ 15	0,91
KOM GR* GR KOM → VN	48/ 5	0,91
SZ GR* NN KL → VN	29/ 3	0,91
GR GR* NN PU → VN	10/ 1	0,91
GR* NN SZ GR → VN	95/ 10	0,90
GR GR* NN → VN	73/ 8	0,90
KL TIT GR* GR → VN	18/ 2	0,90
TIT GR* GR → VN	29/ 4	0,88
GR* NN KL DET → VN	14/ 2	0,88
VN GR* NN → VN	7/ 1	0,88
GR* NN → VN	282/ 41	0,87
GR GR* NN KL → VN	40/ 6	0,87
GR* NN KL → VN	126/ 21	0,86
SZ KL GR* NN → VN	18/ 3	0,86
SZ GR GR* NN → VN	12/ 2	0,86
VN GR* NN SZ → VN	6/ 1	0,86
GR KOM GR* GR → VN	76/ 13	0,85
KL GR* NN KOM → VN	34/ 7	0,83
KOM KL GR* NN → VN	15/ 3	0,83
VN GR* GR KOM → VN	5/ 1	0,83
KOM TIT GR* → VN	5/ 1	0,83
KL GR* NN SZ → VN	47/ 10	0,82
TIT GR* NN KL → VN	9/ 2	0,82
GR* NN KL KL → VN	54/ 13	0,81
TIT GR* GR KL → VN	13/ 3	0,81
KL GR* NN → VN	99/ 25	0,80
NN KL GR* GR → VN	39/ 10	0,80
PU GR* GR NN → VN	8/ 2	0,80
GR* NN KL PU → VN	4/ 1	0,80
DET TIT GR* GR → VN	4/ 1	0,80
TIT PU GR* GR → VN	4/ 1	0,80
TIT SZ GR* GR → VN	4/ 1	0,80
NN KOM GR* GR → VN	33/ 9	0,79
GR* NN KL SZ → VN	11/ 3	0,79
SZ TIT GR* → VN	7/ 2	0,78
PU GR* NN → VN	10/ 3	0,77
VN GR* GR SZ → VN	6/ 2	0,75
SZ KOM GR* NN → VN	6/ 2	0,75
SZ SZ GR* NN → VN	6/ 2	0,75
VN KL GR* GR → VN	6/ 2	0,75
GR* VN SZ GR → VN	3/ 1	0,75
DET GR* NN SZ → VN	3/ 1	0,75
GR KL GR* MIX → VN	3/ 1	0,75
KL NN KOM GR* → VN	3/ 1	0,75
KOM KL TIT GR* → VN	3/ 1	0,75
KOM TIT SZ GR* → VN	3/ 1	0,75
GR* VN KOM → VN	3/ 1	0,75
GR GR GR* NN → VN	11/ 4	0,73
KL KL TIT GR* → VN	8/ 3	0,73
KL TIT GR* → VN	21/ 8	0,72
TIT GR* → VN	32/ 13	0,71
PU GR* NN KL → VN	7/ 3	0,70

Erklärung

Ich versichere, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, den 9. September 2002 Unterschrift