Institut für Maschinelle Sprachverarbeitung

Abteilung Theoretische Computerlinguistik

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 2529

Auflösung von geographischen Ambiguitäten im Deutschen

Reinhard Kuntz

Studiengang:	Informatik
Prüfer:	Prof. Hinrich Schütze
Betreuer:	Dipl. Inf. André Blessing
begonnen am:	16. Oktober 2006
beendet am:	17. April 2007
CR-Klassifikation:	I.2.6, I.2.7, H.3.1

Zusammenfassung

Im Rahmen dieser Arbeit wurden grundlegende Aspekte von geo / nicht-geo Ambiguitäten im Deutschen und deren Auflösung untersucht und diskutiert.

Da einfache Regeln zum Auflösen von Mehrdeutigkeiten nicht ausreichen, wurde der Ansatz verfolgt, mit Verfahren des maschinellen Lernens ein Modell zur Disambiguierung zu generieren. Auf der Basis von SVM wurde ein überwachtes Verfahren und ein Self-Training-Verfahren implementiert und analysiert. Außerdem wurde ein weiterer Self-Training-Ansatz, beruhend auf einem Mehrheitsverfahren, entwickelt, da das Ziel verfolgt wurde, einen Ansatz zu finden, der ohne größere handannotierte Lernmenge auskommt.

Aus diesem Grund wurde außerdem untersucht, auf welche Weise eine Lernmenge komplett maschinell generiert werden kann. Anhand dieser Untersuchungen wurde ein Bootstrap-Verfahren implementiert.

Aufbauend auf das überwachte Verfahren und das Bootstrap-Verfahren ist ein Disambiguierungs- und Visualisierungsprototyp erstellt worden, mit dem in der OpenJUMP-Plattform die disambiguierten Orte angezeigt werden können.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation . . . 1

1.2 Aufgabenstellung . . . 2

1.3 Überblick und Aufbau . . . 3

1.4 Danksagung . . . 4

2 Grundlagen **5**

2.1 Informationsextraktion . . . 5

2.2 Eigennamen und Eigennamenerkennung . . . 6

2.3 Verfahren der Eigennamenerkennung . . . 8

2.4 Maschinelles Lernen . . . 9

2.4.1 Überwachtes Lernen . . . 9

2.4.2 Halb-überwachtes Lernen . . . 10

3 Die Problematik **13**

3.1 Geographische Entitäten . . . 13

3.2 Geographische Ambiguitäten . . . 14

3.3 Text . . . 16

3.4 Gazetteer . . . 17

3.5 Visualisierung mit OpenJUMP . . . 18

3.6 Untersuchungen . . . 19

4 Verwandte Arbeiten **22**

4.1 Disambiguierung . . . 22

4.2 Geo-Coding . . . 23

4.3 Geographische Suchmaschinen . . . 24

4.4 Eigennamenerkennung . . . 25

4.5 Auswertung / Zusammenfassung . . . 27

5 Lösungsansatz **28**

5.1 Herangehensweise . . . 28

5.2 Repräsentation der Kandidaten . . . 31

5.3 Verfahrensansätze . . . 34

6 Überwachtes Verfahren **35**
6.1 SVM 35
6.2 SVMlight 37
6.3 Implementierung 38
6.4 Analyse 39

7 Self-Training-Verfahren (SVM) **45**
7.1 Algorithmus 45
7.2 Analyse 46

8 Self-Training-Verfahren (Mehrheitsverfahren) **51**
8.1 Chi-Square 51
8.2 Algorithmus 52
8.3 Analyse 54

9 Bootstrap-Verfahren **58**
9.1 Trainingsmenge maschinell erzeugen 58
9.2 Algorithmus 60
9.3 Analyse 62

10 Validierung der Lernverfahren **66**
10.1 Überwachtes und Self-Training-Verfahren (SVM) 66
10.2 Self-Training-Verfahren (Mehrheitsverfahren) 67
10.3 Bootstrap-Verfahren 70
10.4 Analyse der Klassifikationsbeispiele 71

11 Disambiguierungs- und Visualisierungs-Prototyp **76**
11.1 Implementierung 76
11.2 Beispiel 77

12 Zusammenfassung und Ausblick **79**
12.1 Zusammenfassung 79
12.2 Fazit 80
12.3 Ausblick 81

Kapitel 1

Einleitung

1.1 Motivation

Der Fortschritt der Technik im Bereich von mobilen Endgeräten hat bereits zu einem wachsenden Einsatz von mobilen Anwendungen geführt. Das Ziel des Sonderforschungsbereichs Nexus ist es, ein mobiles kontextbezogenes System zu entwickeln [RBB03]. Im Mittelpunkt dieses Systems steht ein Umgebungsmodell, welches den Kontext repräsentiert. Dieses Umgebungsmodell sollte die Umgebung möglichst zeitnah und vollständig abbilden. Dies wirft die Frage auf, wie das Kontextmodell aktualisiert werden kann, wenn sich die Umgebung verändert.

Eine Möglichkeit stellt explizites Modellieren und Ändern des Kontextmodelles durch einen Systemdesigner dar. Wünschenswert wäre jedoch ein automatisiertes Aktualisieren. Es gibt eine große Anzahl von Informationsquellen, bei denen das automatisierte Erstellen und Aktualisieren eines Modells sehr gut vorstellbar ist. Beispielsweise bei Sensoren und Datenbanken. Eine weitere Quelle ist die natürliche Sprache (in geschriebener oder gesprochener Form) [BKN+06].

Viele Eigenschaften und Veränderungen der Umgebung können mittels eines „niederen Kontextes“, wie beispielsweise Sensordaten, erkannt werden. Es wird aber häufig auch eine Selektion und Interpretation solcher Daten – ein „höherwertiger Kontext“ – benötigt. So kann ein GPS-Sensor den „niederen Kontext“, die GPS-Position liefern, nicht aber, ob es sich dabei um eine Fußgängerzone, eine Straße oder einen anderen bestimmten Ort handelt [BKN+06]. Wenn zwei Ortschaften zu einer zusammengefasst werden oder sich eine Landesgrenze verschiebt, kann dies ebenfalls nur mit Hilfe von Informationsquellen mit „höherwertigem Kontext“ erfasst werden. „Natürliche Sprache“ stellt die Hauptform der menschlichen Kommunikation dar. Sie enthält insbesondere „höherwertigen Kontext“, da der Inhalt vom Sprecher stets schon interpretiert ist. Außerdem steht „Natürliche Sprache“ in großer Menge zur Verfügung. Beispielsweise in Form von Internetseiten, Zeitungen, Newsfeeds und beim Rundfunk in gesprochener Form. Eine Zeitung meldet beispielsweise, dass eine Straße für einen bestimmten Zeitraum gesperrt wird, oder wegen einer Baustelle mit Behinderungen zu rechnen ist. Mit Sensordaten können hingegen lediglich die Auswirkungen – kein Autoverkehr bzw. stockender Verkehr – wahrgenommen werden. Nicht

jedoch der Grund, der in einem entsprechenden Zeitungsartikel gemeldet wird.

Wie zu Beginn schon angedeutet, ist die Aktualität und Vollständigkeit des Umgebungsmodells von zentraler Wichtigkeit. Mit Hilfe von Sensordaten kann ein Kontextmodell bezüglich „niederem Kontext“ aktuell gehalten werden. Was „höherwertigen Kontext“ betrifft, ist dies wie schon geschildert nicht möglich. Text (in Form von Webseiten, Zeitungen, Newfeeds, etc.) ist eine Informationsquelle, die Aktualität bezüglich „höherwertigem Kontext“ gewährleisten kann. Außerdem liefert Text Informationen, die beispielsweise Sensoren nicht bieten können, und stellt hierdurch eine Möglichkeit zur Vervollständigung dar.

Um Informationen aus Texten in das Umgebungsmodell aufnehmen zu können, müssen (geographische) Entitäten (siehe Abschnitt 3.1) in Texten mit (geographischen) Entitäten im Kontextmodell in Übereinstimmung gebracht werden. Hierfür müssen die Entitäten in der natürlichen Sprache bzw. den Texten zunächst richtig erkannt und verstanden werden.

Im Internet sind zahlreiche geographische „Wörterbücher“, auch als Gazetteer bezeichnet, für verschiedene Kategorien von geographischen Entitäten zu finden. Diese stehen frei zur Verfügung und können somit durch einfaches Vergleichen als Basis zum Auffinden von geographischen Entitäten in Texten herangezogen werden. Außerdem sind beispielsweise für Navigationssysteme große sehr detaillierte Datenbasen erstellt worden, die ebenfalls wie die freien Gazetteer verwendet werden können. Ein entsprechender Datensatz der Firma Navteq liegt im GDF (Geographical Data Format) für Baden-Württemberg vor und kann somit ohne größeren Aufwand verwendet werden.

1.2 Aufgabenstellung

Wie in Abschnitt 1.1 schon deutlich wurde, ergibt sich das Thema bzw. die Aufgabenstellung dieser Diplomarbeit direkt aus der Problemstellung des Projekts E2 – „Semantische Verfahren zur Verwaltung von Umgebungsmodellen“ – im Sonderforschungsbereich Nexus. Das Ziel von Nexus ist, ein mobiles kontextbezogenes System zu entwickeln, in dessen Mittelpunkt ein Umgebungsmodell steht, das den Kontext repräsentiert. Von zentraler Wichtigkeit ist die Aktualität und Vollständigkeit des Umgebungsmodells. Text (in Form von Webseiten, Zeitungen, Newfeeds, etc.) ist eine Informationsquelle, die Aktualität und Vollständigkeit gewährleisten kann.

Hierbei stellt sich das Problem, dass die Abbildung von geographischen Entitäten (siehe Abschnitt 3.1) auf das Umgebungsmodell schwierig ist. Insbesondere gilt dies für geographische / nicht-geographische Mehrdeutigkeiten (geo / nicht-geo Ambiguitäten). Beispielsweise kann „Singen“ sowohl eine Stadt (der geographische Sinn) bezeichnen als auch eine musikalische Aktivität (der nicht-geographische Sinn) beschreiben.

Die Entwicklung und Evaluierung eines Algorithmus für die Auflösung von geo / nicht-geo Ambiguitäten im Deutschen soll Gegenstand dieser Diplomarbeit sein. Die korrekt identifizierten geographischen Lokationen sollen dann mit Hilfe der JUMP-Umgebung graphisch angezeigt werden. Grundlage der Erkennung geographischer Entitäten soll ein Ausschnitt der GDF-Datenbasis (von der Firma Navteq) sein, der eine vollständige Liste der geographischen Namen Baden-Württembergs enthält.

Die folgenden Teilaufgaben sind zu lösen:

- Identifizierung geographischer Randbedingungen für die Auflösung.
- Identifizierung von einfachen Wortmustern für die Auflösung der Ambiguitäten.
- Sonderbehandlung von Worten mit extrem seltener geographischer Zweitbedeutung (z.B. Achim – eine Stadt in Norddeutschland).
- Prototypische Implementierung einer Anwendung, die Webseiten verarbeitet und geo / nicht-geo Ambiguitäten auflöst.
- Einbindung von Jump in die Anwendung zwecks Visualisierung der gefundenen Lokationen.

1.3 Überblick und Aufbau

Zu Beginn dieser Arbeit wurden verschiedene Untersuchungen durchgeführt. Dabei stellte sich heraus, dass es zum Auflösen von geographischen / nicht-geographischen Mehrdeutigkeiten nicht ausreicht, einfache Regeln zu bilden und zu verwenden. Deshalb wurde im Rahmen dieser Arbeit der Ansatz verfolgt, mit Methoden des maschinellen Lernens Regeln bzw. ein Modell zur Disambiguierung zu generieren. Dabei wurde die Ähnlichkeit der Problematik bei der Eigennamenerkennung und der Problemstellung bei der Disambiguierung mitbetrachtet.

Diesen Aspekten entsprechend findet sich im nächsten Kapitel neben einigen Begriffserläuterungen ein Überblick über die Verfahren der Eigennamenerkennung. Außerdem werden verschiedene, für diese Arbeit relevante, grundlegende Methoden des maschinellen Lernens vorgestellt.

Im dritten Kapitel wird auf verschiedene Teilprobleme eingegangen, die sich aus der Aufgabenstellung ergeben. So wird diskutiert, was unter einer geographischen Entität zu verstehen ist. Darauf folgend wird aufgezeigt, welche Arten von Mehrdeutigkeiten es gibt. Dabei werden verschiedene Aspekte der Disambiguierungsthematik anhand einiger Beispiele illustriert. Einführend wird außerdem auf OpenJUMP eingegangen. Der in dieser Arbeit verwendete Korpus und die verwendeten Gazetteer werden vorgestellt. Für diese Arbeit grundlegende Untersuchungen schließen daraufhin das dritte Kapitel ab.

Verwandte Arbeiten werden im vierten Kapitel vorgestellt. Hierbei wird neben Arbeiten zur Disambiguierung insbesondere auf Publikationen aus den Bereichen Geo-Coding und geographische Suchmaschinen eingegangen. Aus dem Bereich der Eigennamenerkennung werden außerdem zwei Arbeiten vorgestellt. Das Kapitel „Verwandte Arbeiten“ wird abgerundet durch eine Zusammenfassung der für diese Arbeit relevanten Aspekte aus den „Verwandten Arbeiten“.

Der zu Beginn dieses Abschnitts bereits kurz umrissene grundlegende Lösungsansatz wird in Kapitel 5 detailliert vorgestellt und diskutiert. Dabei wird aufgezeigt, dass die Problematik der Disambiguierung auf die der Eigennamenerkennung bzw. der Textklassifikation zurückgeführt werden kann, weshalb Verfahren der Textklassifikation zum Einsatz

kommen können. Dementsprechend wird aufgezeigt, wie die einzelnen zu disambiguierenden Beispiele repräsentiert werden können.

Darauf folgend wird in Kapitel 6 ein überwachtes Verfahren zum Erlernen eines Modells zur Disambiguierung vorgestellt. In den Kapiteln 7 und 8 werden Verfahren beschrieben, bei denen Self-Training zum Einsatz kommt und in Kapitel 9 wird ein Bootstrap-Verfahren präsentiert. Dabei wird jeweils der Ansatz und die Implementierung erläutert. Außerdem werden die Ergebnisse experimenteller Untersuchungen zu den Verfahren dargestellt.

In Kapitel 10 findet sich eine Validierung der zuvor vorgestellten Verfahren und in Kapitel 11 wird die Implementierung des Disambiguierungs- und Visualisierungs-Prototypen erläutert. Dabei wird auf verschiedene Schnittstellen und die Einbindung in die OpenJUMP-Plattform eingegangen.

Abgerundet wird diese Arbeit mit einer Zusammenfassung und einer abschließenden Diskussion der Ergebnisse, welche in Kapitel 12 zu finden sind.

1.4 Danksagung

An dieser Stelle möchte ich mich bei meinem Betreuer Dipl. Inf. André Blessing bedanken, der mir mit kritischen Fragen, Anregungen, Hilfestellungen und durch Ermutigung zur Seite stand.

Vielen Dank an alle, die mich beim Erstellen dieser Diplomarbeit durch motivierenden Zuspruch unterstützt haben.

Kapitel 2

Grundlagen

Nachdem die Motivation zu dieser Arbeit und die genaue Aufgabenstellung im letzten Kapitel vorgestellt wurden, soll nun zunächst auf einige grundlegende Aspekte eingegangen werden.

In den folgenden Abschnitten werden verschiedene Begriffe wie Informationsextraktion, Eigennamen und Eigennamenerkennung erläutert. Darauf folgt ein Überblick über Verfahren der Eigennamenerkennung. Am Ende dieses Kapitels werden dann einige für diese Arbeit relevante Ansätze des maschinellen Lernens vorgestellt.

2.1 Informationsextraktion

Mit der stark wachsenden Anzahl von Texten, insbesondere im Internet, wird es immer schwieriger, das vorhandene Informationspotential auszunutzen. Deshalb entstand der Forschungsbereich der Informationsextraktion (IE - engl. Information Extraction) mit dem Ziel Systeme zu entwickeln, die gezielt relevante Informationen in großen Textmengen finden, extrahieren und kompakt repräsentieren können [Neu04] S. 502. Üblicherweise geschieht dies domänenspezifisch und ohne dass eine umfassende Analyse der Texte vorgenommen wird. Das heißt, dass versucht wird lediglich relevante Passagen der Texte zu analysieren bzw. zu „verstehen“, um die Verfahren effizient zu halten.

Informationsextraktionssysteme lassen sich allgemein durch ihre Eingabe und Ausgabe charakterisieren [Neu04] S. 502f. Die Eingabe besteht dabei stets aus einer Menge von unstrukturierten Daten (Texten), aus denen Informationen extrahiert werden sollen. Hinzu kommt, dass die Eingabe sogenannte Templates enthält. Templates spezifizieren durch eine Menge von Merkmalen die Informationen, die extrahiert werden sollen. Beispielhaft ist ein solches stark vereinfachtes Template in Abbildung 2.1 links oben aufgeführt. Informationsextraktionssysteme generieren aus einer solchen Eingabe eine Menge als relevant identifizierter Textpassagen bzw. der daraus gewonnenen Werte für alle oder Teile der Merkmale der Templates aus der Eingabe. In Abbildung 2.1 rechts oben ist eine solche beispielhafte Ausgabe zum Beispiel Template (Abb 2.1 links oben) und dem Beispieltext (in Abbildung 2.1 unten) dargestellt.

Diese Definition von Informationsextraktion ist die ursprüngliche und engste, auch

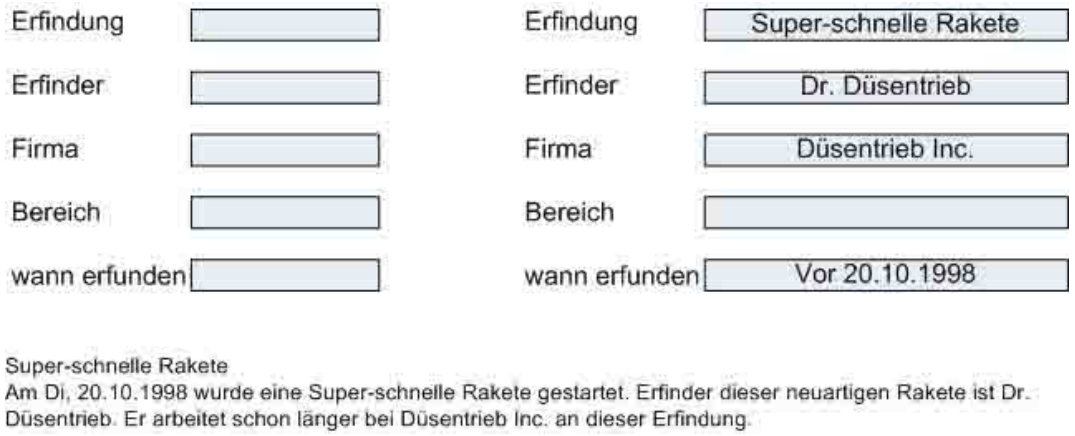

Abbildung 2.1: Beispiel Informationsextraktion

als „Domänenspezifische Informationsextraktion“ oder „Patternfilling“ bezeichnet (vergl. [LBC04] S. 7). Zum Teil wird Informationsextraktion jedoch allgemeiner als irgendeine Form der Informationsgewinnung verstanden. Dabei ist zu beachten, dass der Prozess der Informationsextraktion verschiedene Schritte bzw. Teilaufgaben beinhaltet, die oft ebenfalls als Informationsextraktion bezeichnet werden. Die Eigennamenerkennung (engl. Named Entity Recognition - NER), auch bekannt unter Entity Identification (EI - vergleiche [*W1]), ist eines von diesen Teilgebieten [Neu04] S. 506. So wird die Eigennamenerkennung auch als „Domänenunabhängige Informationsextraktion“ betrachtet [LBC04] S. 7.

Bei der Eigennamenerkennung geht es um das Aufspüren von spezifischen Ausdrücken (der Eigennamen) und dem Einteilen dieser in Kategorien wie beispielsweise Personennamen, Firmennamen, usw.. Auf Eigennamen und deren Erkennung wird in den Abschnitten 2.2 und 2.3 näher eingegangen.

2.2 Eigennamen und Eigennamenerkennung

Die Definition bzw. Abgrenzung, was alles unter einem Eigennamen (engl. Named Entity - NE) verstanden wird, ist in der Literatur fließend [Rot02] S. 5. Im Englischen setzt sich die Bezeichnung für Eigennamen aus Entität und „benannt“ bzw. „mit Namen“ zusammen. Diese zwei Aspekte sollen an dieser Stelle für ein besseres Verständnis des Begriffes Eigennamen betrachtet werden.

Als Entität bzw. Entity (synonym Informationsobjekt) wird in der Informatik ein eindeutig zu bestimmendes Objekt bezeichnet. Ein Objekt bzw. eine Entität kann dabei materiell oder immateriell sein. Jede Entität (das einzelne Objekt) wird einem Entitätstyp zugeordnet [*W2], wie dies in Abbildung 2.2 für die Entitätstypen Abteilung, Berg und Auto beispielhaft dargestellt ist. Das heißt, dass Entitäten somit konkrete und eindeutig identifizierbare Ausprägungen eines Entitätstyps darstellen. Einzelne Entitäten eines Entitätstyps können als Entitätsmenge zusammengefasst werden. Hierbei ist zu beachten, dass Entitäten als Elemente einer Entitätsmenge immer durch ihre Eigenschaften (Werte

von beschreibenden Attributen) unterschieden werden können [*W2]. Bei einer Menge von Entitäten des Typs Auto könnte ein identifizierendes Attribut beispielsweise die Fahrgestellnummer sein, wie in Abbildung 2.2 rechts dargestellt.

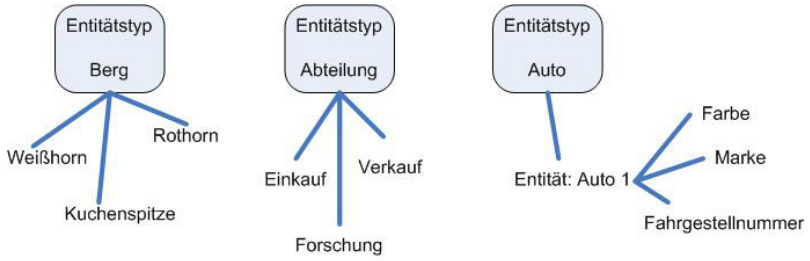

Abbildung 2.2: Entitäten und ihre Entitätstypen

Bei Eigennamen bzw. Named Entity kommt der Aspekt der Benennung hinzu. Um beim zwischenmenschlichen Kommunizieren einfacher auf einzelne Objekte Bezug nehmen zu können, ist es üblich, Informationsobjekten einen Namen zu geben. Hierbei wird versucht einen Namen zu verwenden, der „sprechend“ und eindeutig ist. Das heißt, Objekte, die mehrfach Teil oder Inhalt einer Kommunikation sind, erhalten einen referenzierenden und identifizierenden Namen. Besonders interessant sind hierbei Namen, die über längere Zeiträume Bestand haben, wie beispielsweise Personennamen, Namen von Bergen und Ähnliche. Es muss allerdings beachtet werden, dass Namen immer nur innerhalb eines bestimmten Kontext eindeutig sind.

Eigennamen bzw. Objekte mit Eigennamen können somit als materielle oder immaterielle Konstrukte betrachtet werden, die innerhalb eines Kontextes über eine relativ große Zeitspanne hinweg mit einem gleichbleibenden, eindeutigen Namen versehen sind.

In der menschlichen Kommunikation ist es allerdings schwer, Grenzen eines Kontextes klar zu definieren. Deshalb findet zu Beginn einer Unterhaltung oder eines Textes meist eine Art Kontextsynchronisation statt. Bei Eigennamen, bei denen der Sprecher oder Schreiber denkt, dass der Kommunikationspartner unter Umständen eine andere Entität mit dem Namen in Verbindung bringen könnte, werden zusätzliche identifizierende Informationen implizit oder auch explizit neben dem Eigennamen weitergegeben, um die „richtige“ Assoziation herbeizuführen. Geschieht das nicht, kommt es zu Missverständnissen durch Kontextübergriffe bzw. Namensraumvermischungen.

Der Begriff Eigennamenerkennung (engl. Named Entity Recognition - NER) ist etwas verwirrend, da der Aspekt der Erkennung bzw. Wiedererkennung (recognition) betont wird. Meist wird im Gegensatz hierzu jedoch unter Eigennamenerkennung (NER) sowohl das Aufspüren von Eigennamen als auch das Einteilen dieser in Kategorien verstanden. Um Missverständnissen vorzubeugen, werden im Folgenden drei Begriffe unterschieden. Unter Eigennamenentdeckung (engl. Named Entity Detection - NED vergl. [Lan06] S. 1) soll das Aufspüren von Eigennamen in Texten verstanden werden. Mit Eigennamenklassifikation (engl. Named Entity Classification - NEC) wird das Einteilen der aufgespürten Eigennamen in Kategorien bzw. Klassen bezeichnet. Die Eigennamenerkennung schließlich beinhaltet

die Eigennamenentdeckung und die Eigennamenklassifikation (vergleiche [Lan06] S. 1).

In der Literatur zur Eigennamenerkennung werden unter Eigennamen in den meisten Fällen Entitäten der Entitätstypen Personen-, Orts-, Organisations- sowie Produktnamen verstanden. In manchen Fällen wird das Spektrum jedoch erheblich erweitert, indem Namen von Monaten, Wochentagen, Festtagen, Religionsnamen, ... ebenfalls als Eigennamen betrachtet werden [Rot02] S. 5.

Abhängig von der thematischen Domäne und der zugrunde liegenden Sprache werden zur Eigennamenerkennung unterschiedlich komplexe Verfahren benötigt und verwendet. Beispielsweise ist es im Englischen relativ einfach, Eigennamen zu erkennen, da diese praktisch die einzigen Wörter sind, welche mit einem Großbuchstaben beginnen, wogegen im Deutschen Eigennamen zwar ebenfalls groß geschrieben werden, allerdings auch alle anderen Substantive [Lan06] S. 1. Ein kurzer Überblick bezüglich der grundlegenden Verfahren zur Eigennamenerkennung findet sich im folgenden Abschnitt.

2.3 Verfahren der Eigennamenerkennung

Es lassen sich grundsätzlich drei verschiedene Lösungsansätze bei den Verfahren für die Eigennamenerkennung unterscheiden – der lexikonbasierte, der regelbasierte und der statistische Ansatz. Hinzu kommen sogenannte hybride Ansätze, die Aspekte aus den drei genannten Ansätzen vereinen [Lan06] S. 1. Oft werden zum Beispiel Eigennamenlisten (lexikonbasiert) zusammen mit Grammatiken (regelbasiert) verwendet, damit auch nichtlexikalische Ausdrücke behandelt werden können [Neu04] S. 506, was ohne Grammatiken nicht möglich wäre.

Die einfachste Methode zur Eigennamenerkennung ist der lexikonbasierte Ansatz. Dieses Verfahren verwendet ausschließlich ein Wörterbuch. Das Wörterbuch muss alle relevanten Wörter bzw. Phrasen explizit enthalten (vergl. [Lan06] S. 1). Hierbei ist zu beachten, dass auch alle Varianten im Lexikon abgespeichert werden müssen, damit diese erkannt werden können. Üblicherweise enthält das Lexikon bzw. Wörterbuch zusätzlich zu den Wörtern auch die Zuordnung dieser zu den zugehörigen Klassen. Durch schlichtes Vergleichen mit den Lexikonwörtern werden Wörter bzw. Wortsequenzen im Text als Eigennamen erkannt und klassifiziert. Der Vorteil eines solchen Verfahrens ist, dass es schnell und einfach ist. Nachteilig ist, dass das Lexikon meist per Hand erstellt werden muss. In manchen thematischen Gebieten stößt das Verfahren auf seine Grenzen durch die Größe des Lexikons. Bereits für Namen von Personen stellt die sehr große Anzahl ein Problem dar - beispielsweise gibt es über eine Million angelsächsische Nachnamen [Lan06] S. 3. Leicht ersichtlich ist, dass ein lexikonbasiertes Verfahren außerdem nicht in der Lage ist, Mehrdeutigkeiten (Ambiguitäten) aufzulösen. Aus diesen Gründen werden lexikonbasierte Systeme meistens nur als Grundlage bzw. in Kombination mit anderen Ansätzen verwendet [Lan06] S. 3.

Wie der Name schon zum Ausdruck bringt, arbeiten regelbasierte Systeme mit Regeln bzw. Regelmengen. Diese Regeln treffen Aussagen darüber, wann eine Sequenz als Eigennamen markiert werden soll und zu welcher Klasse ein Eigenname gehört [Lan06] S. 1. Für die Regelmenge werden oftmals verschiedene Wissensformen verwendet, z.B. Lexikonwis-

sen, syntaktisches Wissen, morphologisches Wissen und auch weiteres domänenspezifisches Wissen. Im Bereich der Eigennamenerkennung handelt es sich bei den Regeln oftmals um Produktionen einer Grammatik. Ein Parser versucht durch Anwendung der Regeln in den Sätzen und Teilsätzen des Textes Ableitungen für Eigennamen zu finden [Lan06] S. 1.

Das manuelle Erstellen von Regeln ist sehr zeitaufwendig und komplex. Es gibt allerdings Ansätze mittels maschinellen Lernens (siehe Abschnitt 2.4) Regelsysteme zu generieren. Ein Nachteil der regelbasierten Systeme ist, dass sich Regelmengen nur schwer auf eine andere Domäne übertragen lassen. Noch schwieriger wird ein Übernehmen einer Regelmenge für Texte in einer anderen Sprache, da eine Regelmenge oft viel spezifisches, heuristisches Wissen enthält.

Statistische Verfahren verwenden anstelle eines Regelsystems ein statistisches Modell. Hierzu wird mit Hilfe von meist überwachtem maschinellen Lernen (vergl. Abschnitt 2.4) ein Modell bzw. ein Klassifikator generiert. Dies geschieht anhand von (annotierten) Beispielen, die Schätzungen der zugrunde liegenden Verteilungen erlauben. Diese können für Voraussagen über unbekannte Beispiele herangezogen werden [Lan06] S. 12. Statistische Verfahren behandeln Wörter meist als atomare Einheiten. Zum Teil werden aber auch Ansätze verfolgt, welche auf Zeichenebene operieren, beispielsweise mit Hilfe von „Hidden Markov Models“ (HMM) [Lan06] S. 1.

Wie deutlich wurde, ist ein schwerwiegendes Problem aller Verfahren, dass ein erstelltes System nur sehr schwer auf eine andere Domäne oder Sprache übertragen werden kann. Daher geht im Bereich NER die Entwicklung dahin, dass nahezu ausschließlich maschinelle Lernverfahren eingesetzt werden [Neu04] S. 509. In den letzten zehn Jahre kamen sehr viele Ansätze des maschinellen Lernens im Bereich der Informationsextraktion zum Einsatz und wurden zum Standardvorgehen [LBC04] S. 1. Deshalb soll im nächsten Abschnitt auf einige Verfahren des maschinellen Lernens eingegangen werden.

2.4 Maschinelles Lernen

Maschinelle Lernverfahren lassen sich auf unterschiedliche Weise in Klassen aufteilen. Anhand der Art der Trainingsdaten bzw. ihrer Verwendung können drei Klassen gebildet werden. So unterscheidet man üblicherweise überwachtes Lernen (engl. Supervised Learning), halb-überwachtes Lernen (engl. Semi-Supervised Learning) und unüberwachtes Lernen (engl. Unsupervised Learning).

Bei der Eigennamenerkennung werden bisher meist überwachte Lernverfahren verwendet. Aber auch halb-überwachte Ansätze wurden bereits eingesetzt. Deshalb soll im Folgenden auf diese zwei Klassen näher eingegangen werden.

2.4.1 Überwachtes Lernen

Beim überwachten Lernen wird versucht anhand einer Menge von Beispielen direkt einen Klassifikator zu erlernen. Wie in Abbildung 2.3 links dargestellt, wird hierzu ein Klassifikationslernverfahren herangezogen, welches anhand der Trainingsmenge ein Modell berechnet. Dabei besteht die Trainingsmenge aus annotierten (engl. labeled) Beispielen. Das

heißt, bei jedem Beispiel ist vermerkt, zu welcher Klasse es zugeordnet werden muss. Diese Zuordnung der Trainingsbeispiele muss meist von Hand erstellt werden. Das ist in den meisten Fällen sehr aufwendig, da eine große Anzahl von Trainingsbeispielen benötigt wird, die zudem repräsentativ für das Anwendungsgebiet sein muss.

Das erlernte Modell kann dann auf zu klassifizierende Elemente angewendet werden, wie in Abbildung 2.3 gezeigt. Hierbei ist zu beachten, dass nicht unbedingt alle zu klassifizierenden Elemente auch klassifiziert werden können. Dieser Aspekt ist in Abbildung 2.3 rechts unten angedeutet.

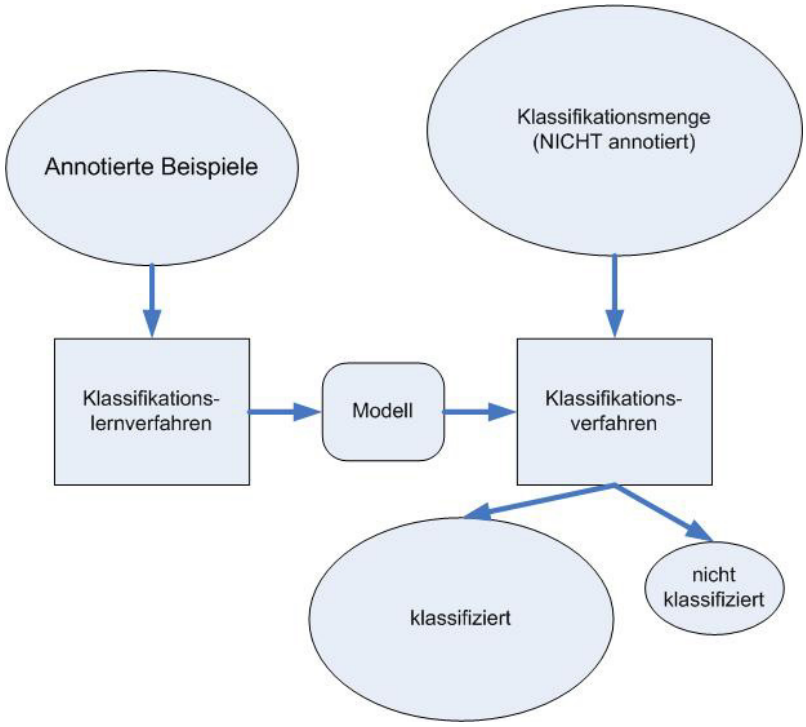

Abbildung 2.3: Überwachtes Lernen

In der Klasse der überwachten Lernverfahren gibt es ein großes Spektrum unterschiedlicher Ansätze bzw. Implementierungen. Einige davon sind das so genannte „Hidden Markov Model“, „Maximal Entropy“ und „Support Vector Maschine“ (vergl. [LBC04] S. 2). Eine der erfolgreichsten und populärsten Methoden des überwachten Lernens bzw. Klassifizierens ist SVM (siehe Abschnitt 6.1). Wie in [LBC04] geschildert wird, entwickelte sich SVM durch seine guten Ergebnisse bei vielen Klassifikationsaufgaben unter anderem auch bei der Eigennamenerkennung zu einem Standardverfahren.

2.4.2 Halb-überwachtes Lernen

Überwachtes Lernen bzw. Klassifizieren betrachtet, wie schon beschrieben, ausschließlich annotierte Daten. Dabei ergab sich das Problem, dass annotierte Daten oftmals schwer zu erstellen sind, insbesondere im Vergleich zu nicht annotierten Daten. Mit dem halb-überwachten Lernen wird genau dieser Aspekt adressiert, indem eine große Menge nicht

annotierter Daten und eine kleine Menge annotierter Daten zum Lernen bzw. Trainieren eines Klassifikators verwendet werden [Zhu06] S. 4. Deshalb wird das halb-überwachte Lernen oftmals auch als „learning from labeled and unlabeled data“ oder als „learning from partially labeled / classified data“, manchmal auch als „bootstrapped supervised learning“ bezeichnet [Zhu06] S. 6.

In der Klasse der halb-überwachten Lernverfahren gibt es ein großes Spektrum unterschiedlicher Ansätze [Zhu06] S.3. Einige davon sind „EM with generative mixture models“, „Self-Training“, „Co-Training“, „Transductive Support Vector Machines“ (TSVM) und „graph-based methods“ [Zhu06] S. 5.

Allgemein stellt sich die Frage, welches Verfahren für welche Aufgabe am besten geeignet ist. Leider lässt sich das nicht allgemein beantworten. Bei halb-überwachten Lernverfahren werden immer starke Annahmen bezüglich des Modells getroffen. Idealerweise sollte also ein Verfahren gewählt werden, dessen Annahmen zum Problemraum passen. In der Praxis kann dies sehr schwierig sein [Zhu06] S. 5.

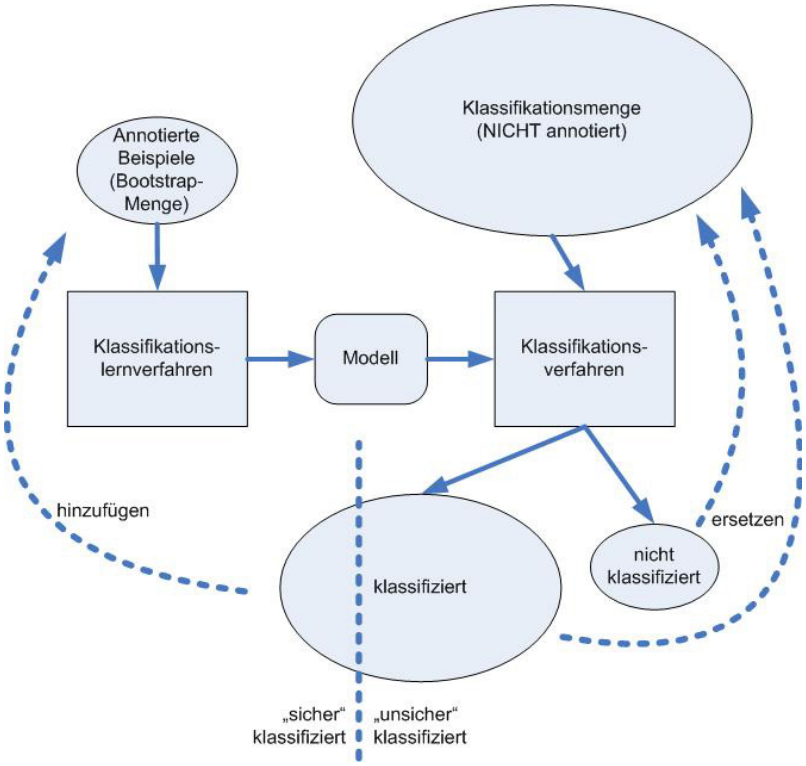

Abbildung 2.4: Technik des Self-Trainings

Das Self-Training ist im Bereich der halb-überwachten Lernverfahren eine häufig angewendete Technik. Deshalb soll im Folgenden ein wenig genauer auf das Self-Training eingegangen werden. Wie in Abbildung 2.4 dargestellt, erlernt man beim Self-Training zunächst einen Klassifikator anhand der annotierten Lernmenge. Also genauso, wie beim überwachten Lernen, nur mit weniger annotierten Daten. Der erlernte Klassifikator bzw. das erlernte Modell wird dann herangezogen, um die nicht annotierte Trainingsmenge zu

klassifizieren. Bei den auf diese Weise klassifizierten Daten gibt es nun welche, die relativ sicher einer Klasse zugeordnet werden können, wie in Abbildung 2.4 unten angedeutet. Üblicherweise werden diese mit ihrer Klassenzuordnung zur annotierten Lernmenge hinzugenommen (vergleiche Abbildung 2.4 links). Wie Abbildung 2.4 rechts zeigt, besteht die nicht annotierte Lernmenge somit nur noch aus den unsicher klassifizierten und den nicht klassifizierten Beispielen. Mit der neu gebildeten annotierten Lernmenge wird nun erneut ein Modell berechnet und das gesamte Vorgehen wiederholt. Dabei muss beachtet werden, dass das Verfahren seine eigenen Einschätzungen heranzieht, um sich selbst zu lehren bzw. zu belehren. Deshalb wird das Self-Training auch „Self-Teaching“ oder „Bootstrapping“ genannt (vergleiche [Zhu06] S. 10f).

Self-Training wurde bereits für verschiedenste Bereiche der maschinellen Sprachverarbeitung herangezogen (vergl. [Zhu06] S. 10f). Unter anderem um zu entscheiden, ob das Wort „plant“ in einem gegebenen Kontext einen lebenden Organismus oder einen Betrieb referenziert. Ebenso ist Self-Training zum Parsen, zur maschinellen Übersetzung und zum Erkennen von Objekten auf Bildern verwendet worden (vergl. [Zhu06] S. 10f).

Kapitel 3

Die Problematik

Im letzten Kapitel wurden ein Überblick über die Verfahren der Eigennamenerkennung gegeben und verschiedene Ansätze des maschinellen Lernens vorgestellt. Nach diesem grundlegenden Teil soll nun in diesem Kapitel auf verschiedene Aspekte bzw. Teilprobleme eingegangen werden, die sich bei der in Abschnitt 1.2 geschilderten Aufgabenstellung ergeben.

Zunächst wird der Begriff „geographische Entität“ erläutert. Im Anschluss daran wird diskutiert, was unter geographischen Mehrdeutigkeiten, insbesondere geographisch / nicht-geographischen Mehrdeutigkeiten zu verstehen ist. Daraufhin werden das Korpus und die Gazetteer vorgestellt, die im Rahmen dieser Arbeit verwendet werden. Nach einer einführenden Vorstellung von OpenJUMP wird das Kapitel durch für diese Arbeit grundlegende Untersuchungen abgeschlossen.

3.1 Geographische Entitäten

In dieser Arbeit soll es um das Auflösen von geo / nicht-geo Ambiguitäten gehen. An dieser Stelle wird deshalb zunächst darauf eingegangen, was unter geo (geographisch) zu verstehen ist. Darauf folgend werden verschiedene Aspekte von geographischen Ambiguitäten diskutiert und einige Definitionen bzw. Abgrenzungen derer in dieser Arbeit gemacht.

Unter Geographie wird üblicherweise

> die Wissenschaft von der Erdoberfläche in ihrer räumlichen Differenzierung, ihrer physischen Beschaffenheit sowie als Raum und Ort des menschlichen Lebens und Handelns

verstanden [Blo02] S. 15. Diese Definition von Geographie beinhaltet ein sehr großes Spektrum mit großem Interpretationsspielraum. Ein solches Verständnis von Geographie macht die Definition und Abgrenzung von geographischen Entitäten (vergl. Abschnitt 2.2) von anderen Entitäten sehr schwierig oder gar unmöglich. Dies ist wohl auch der Grund, warum sich recht unterschiedliche Definitionen von geographischen Entitäten finden lassen, insbesondere auch im Bereich der Eigennamenerkennung. In diesem Bereich wird zudem meist durch Aufzählen von Entitätstypen (vergl. Abschnitt 2.2) oder anderen Kategorisierungen definiert, was eine geographische Entität ist. In [Rot02] S. 6 beispielsweise werden geographische Entitäten durch die Entitätstypen Stadt, Hafen, Flughafen, Insel, Region, Provinz,

Land, Kontinent und Gewässer beschrieben. In [Mih03] wird eine Beschreibung von geographischen Entitäten durch die Kategorien „Natürliche geographische Stätten“ (Berge, Meere, Seen, ...), „Bauwerke“ (Häuser, Türme, Denkmäler, Brücken, Straßen, ...), „Politische Gebietsaufteilungen“ (Gemeinden, Kreise, Länder, Staaten, ...) und „Administrative Gebietsaufteilungen“ (Postleitzahlengebiete, Telefonvorwahlengebiete, ...) vorgenommen.

Eine genaue, allgemeingültige Definition von „geographische Entität“ wird aus diesem Grund im Rahmen dieser Arbeit nicht durchgeführt. Anstelle dessen sollte eine pragmatische Eingrenzug des Begriffes „geographische Entität“ im Rahmen dieser Arbeit genügen.

Unter geographischen Entitäten sollen Verwaltungseinheiten, die in den verwendeten Gazetteer (siehe Abschnitt 3.4) vorhanden sind, verstanden werden. Dies sind Ortschaften, Städte, Gemeinden, Landkreise und Bundesländer in Deutschland sowie Stadtteile in Baden-Württemberg. Dabei ist zu beachten, dass diese Abgrenzung naiv zu betrachten ist. Das heißt, eine Entität wie beispielsweise „Stiftskirche Stuttgart“ soll genauso wie „Stuttgart“ gewertet werden – als geographische Entität – auch wenn „Stiftskirche Stuttgart“ eigentlich ein Gebäude ist. Ebenso sollen „Berlin beschloss die Gesetzesänderung“ oder „Der VFB Stuttgart gewann das Spiel“ als die geographischen Entitäten „Berlin“ und „Stuttgart“ betrachtet werden. Also immer, wenn eine direkte geographische Verbindung herstellbar ist, wie bei der Regierung, die in Berlin ihren Sitz hat.

Im Verlauf dieses Kapitels werden unter anderem verschiedene Aspekte vorgestellt und diskutiert, die zu dieser pragmatischen „Definition“ von geographischen Entitäten geführt haben.

3.2 Geographische Ambiguitäten

Beim Bestimmen von geographischen Entitäten ergeben sich verschiedene Schwierigkeiten insbesondere durch Mehrdeutigkeiten. Bei geographischen Entitäten können zwei Arten von Mehrdeutigkeiten auftreten – geographisch / geographische Mehrdeutigkeiten und geographisch / nicht-geographische Mehrdeutigkeiten (siehe [AHSS04] S. 1).

Allgemein spricht man von geographisch / geographischen Mehrdeutigkeiten, wenn verschiedene geographische Entitäten denselben Namen tragen. Beispielsweise London in England und London in Ontario [AHSS04] S. 1. Eine geographisch / nicht-geographische Mehrdeutigkeit liegt vor, wenn eine (oder mehrere) geographische Entität(en) denselben Namen tragen wie anderweitige Entitäten oder mit einem, eine andere Bedeutung tragenden Wort in derselben oder auch einer anderen Sprache übereinstimmen. Zum Beispiel der Personenname „Tamm“, das Bindewort “Weil„ bei Begründungen, die Farbe „Rot“, oder die Mehrzahl von Hof oder Haus („Höfen“, „Häusern“), welche alle gleichzeitig Namen von Ortschaften in Baden-Württemberg darstellen.

Einige weitere Beispiele von Ortschaften aus Baden-Württemberg, deren Namen geographisch / nicht-geographisch mehrdeutig sind, findet sich in Tabelle 3.1.

Wie in Abschnitt 1.2 schon erläutert, sollen im Rahmen dieser Arbeit lediglich geographisch / nicht-geographische Ambiguitäten aufgelöst werden. Die im Folgenden aufgeführten Beispiele mit sowohl geographisch /geographischen als auch geographisch / nicht-

Ahorn	Kuchen	Wald
Berg	Lauf	Wangen
Buchen	Leimen	Weingarten
Forst	Mengen	Winden
Keltern	Moos	Zimmern
Korb	Süßen	

Tabelle 3.1: Beispiele mehrdeutiger Ortschaftsnamen bei Ortschaften aus Baden-Württemberg.

geographischen Mehrdeutigkeiten sollen trotzdem kurz vorgestellt werden, da sie zeigen, wie komplex die Thematik der Disambiguierung ist. Auch wird deutlich, dass das Auflösen von geo / nicht-geo Ambiguitäten relativ eng mit dem Auflösen von geo / geo Ambiguitäten verbunden sein kann.

1. Alter See : See in Potsdam - Straße in Ludwigsburg
2. Freiberg : Stadtteil des Stadtbezirks Stuttgart-Mühlhausen - Schloss bei Gleisdorf in der Steiermark - Berg im Süden von Kärnten
3. Altach : Gemeinde in Vorarlberg - Straße in Ludwigsburg
4. Schwäbisch Hall : Bausparversicherung - Stadt
5. Wüstenrot : Versicherungsunternehmen - Stadt

An den aufgeführten Beispielen wird deutlich, dass nicht nur geo / geo Ambiguitäten vorkommen, die aus derselben Klasse (Stadt - Stadt) kommen, sondern ebenso aus unterschiedlichen Klassen stammen. In 1 z.B. Gewässer - Straße. In 2 sogar aus den drei Klassen Stadtteil - Bauwerk - Berg. Bei den Beispielen 4 und 5 wird deutlich, dass sich im Deutschen eine Problematik daraus ergibt, dass Unternehmen oder andere Entitäten nach Orten benannt werden. An dieser Stelle wird ein Zuordnen zu geographischen oder nicht geographischen Entitäten sehr schwierig.

Bei „Ludwigsburg“, „Stuttgart“, „Ulm“ oder Ähnlichem ist leicht ersichtlich, dass es sich um eine geographische Entität (Stadt) handelt. Schon schwieriger wird es mit „Schleyerhalle Stuttgart“, „Stiftskirche Stuttgart“ oder „VFB Stuttgart“. Bei allgemeiner bzw. intuitiver Betrachtung sollten „Schleyerhalle Stuttgart“ ebenso wie „Stiftskirche Stuttgart“ sicher als geographische Entität (Gebäude) erkannt werden. Möchte man jedoch lediglich „reine“ Nennungen von Ortschaften oder Ortschaftsteilen als geographische Entität betrachten, dürfen diese Beispiele nicht als geographisch angesehen werden. Hieraus würde sich also die Problematik ergeben, das verschiedene intuitiv als geographisch zu betrachtende Klassenzuordnungen aufgelöst werden müssen. Ein weiteres Beispiel hierzu ist „Freiberg“ (2), bei dem bestimmt werden muss, ob es sich um das Schloss, den Stadtteil oder den Berg handelt. Je nach Zielsetzung sollten jedoch Beispiele wie „Stiftskirche Stuttgart“ auch als Ort, besser gesagt als „Stuttgart“ erkannt werden. Bei „VFB Stuttgart“ kann allerdings

sowohl der Sitz des Fussballvereins (Ort bzw. Stuttgart) als auch die Mannschaft des Vereines gemeint sein („der VFB Stuttgart gewann das Spiel"). Im zweiten Fall wäre es besser, „VFB Stuttgart" nicht als geographische Entität zu klassifizieren, ähnlich wie beim „Schloss Freiberg". Ebenso ist es mit den bereits angesprochenen Beispielen 4 und 5.

Wie sichtbar wurde, ist das Auflösen von geographischen Mehrdeutigkeiten eine sehr komplexe und vielschichtige Problematik. Um die Komplexität in einem überschaubaren Rahmen zu halten, wurde deshalb entschieden, die anhand des Beispieles „VFB Stuttgart" erläuterte Problematik nicht zu behandeln. Deshalb sollen beispielsweise „VFB Stuttgart", „Kreis Esslingen", „Stadt Stuttgart", „Richtung Weil der Stadt", „Wüstenrot" und „Stiftskirche Stuttgart" als geographische Entität betrachtet werden, sofern sie einen direkten geographischen Bezug zu dem im Namen enthaltenen Ort haben (vergl. Abschnitt 3.1).

3.3 Text

Wie in Abschnitt 1.2 ersichtlich soll das Auflösen von Mehrdeutigkeiten in Texten geschehen, die aus dem WWW, aus Zeitungsartikeln, Newstickern oder Ähnlichem stammen. Daher war die Wahl des Stuttgarter Zeitung Korpus (im Weiteren nur noch STZ genannt) naheliegend, da dieses für diese Art von Texten repräsentativ ist und innerhalb des IMS zur Verfügung steht.

Das STZ beinhaltet die Jahrgänge 1992 und 1993 der Stuttgarter Zeitung. Zeitungsartikel sind zum Teil in verschiedenen Varianten enthalten. Insgesamt umfasst das STZ 36 Millionen Wörter.

Da OpenJUMP (vergl. Abschnitt 3.5), welches zur Visualisierung herangezogen werden soll (siehe Abschnitt 1.2), in Java implementiert ist, macht es Sinn, weitere Implementierungen ebenfalls in Java vorzunehmen. Um einfach und schnell unter Java mit dem STZ arbeiten zu können, wurden 15.976 Artikel des STZ aus der IMS Corpus Workbench (CWB) [*CW] mit dem CWB-Tool cwb-decode (cwb-decode -n -H STZ -P word -S article -S s > STZ.txt) in eine Textdatei extrahiert (näheres siehe [Eve02] S. 5). Diese enthält Zeilenweise die Korpus-Positionsnummer und ein Wort. Artikelgrenzen und Satzgrenzen sind in den entsprechenden Zeilen ebenfalls enthalten. Mit Hilfe eines eigens erstellten Java-Tools wurde diese Datei zerlegt, so dass jeder Artikel in einer gesonderten Datei mit dem Dateinamen

```
<Korpus Position Artikelanfang>-<Korpus Position Artikel Ende>.txt
```

vorliegt. Dabei wurden die Zeilenumbrüche, Positionsnummern und die Artikelgrenzen aus dem Text entfernt und die Satzgrenzen zu alleinstehenden Token umgewandelt.

Angemerkt muss an dieser Stelle werden, dass in der Aufgabenstellung gefordert wird, dass die Disambiguierung auf Internetseiten angewendet werden kann. Das STZ, auf dem im Rahmen dieser Arbeit alle Untersuchungen und Implementierungen durchgeführt werden, liegt in tokenisierter Form vor. Es sollte jedoch möglich sein Internetseiten (durch Parsen und Tokenisieren) vorzuverarbeiten, um die im Rahmen dieser Arbeit erstellten Verfahren auf sie anwenden zu können. Die Aspekte des Parsens und Tokenisierens werden im Rahmen dieser Arbeit nicht weiter betrachtet.

3.4 Gazetteer

In den Abschnitt 1.1 und 1.2 wurde bereits erläutert, dass Gazetteer für verschiedene geographische Bereiche zur Verfügung stehen und zu einem ersten Identifizieren von (geographischen) Entitäten durch einfaches Vergleichen der Wortsequenzen verwendet werden sollen. Eine Datenbasis der Firma Navteq mit Informationen zu Baden-Württemberg liegt vor. Wie in Abschnitt 3.1 diskutiert umfasst Geographie bzw. ein intuitives Verständnis von geographischen Entitäten ein sehr großes Spektrum. Da die Betrachtung des gesamten Spektrums den Rahmen dieser Arbeit sicher gesprengt hätte, wurde zunächst entschieden, lediglich Ortschaften und Ortschaftsteile als geographische Entitäten zu betrachten. Außerdem sollten nur Ortschaften innerhalb von Baden-Würtemberg untersucht und disambiguiert werden, da die zur Darstellung mit OpenJUMP benötigten Daten lediglich für Baden-Würtemberg vorliegen. Daher wurden aus der Datenbasis von der Firma Navteq Ortschaften und Ortschaftsteile extrahiert. Dies sind insgesamt 5836 verschiedene Namen.

Allerdings ergab sich, wie in Abschnitt 3.6 erläutert, dass lediglich 28% der zu disambiguierenden Kandidaten nicht-geographisch sind. Nach Entfernen der kleingeschriebenen sogar nur noch 18%. Deshalb wurde entschieden, zusätzlich einen frei verfügbaren Gazetteer zu verwenden.

Die Wahl fiel auf den weltweiten Gazetteer [*WG]. Er steht zum Herunterladen als Datei zur Verfügung. Diese Gazetteer-Datei enthält eine Liste aller Orte, Verwaltungseinheiten und Ballungsräume. Zu diesen sind außerdem die aktuellen Bevölkerungszahlen, ihre deutschen Namen (falls diese vom internationalen Namen abweichen) und den übergeordneten Ländern enthalten. Die Datei liegt unter [*WG] im UTF-8-Format vor, wurde für die Verwendung innerhalb dieser Arbeit jedoch ins ISO-Format konvertiert. Die Gazetteer-Datei ist zeilenweise und Tab-separiert aufgebaut, wobei die in Abbildung 3.1 aufgeführten Spalten enthalten sind. Dabei ist zu beachten, dass bei nicht vorhandenen Informationen oder bei nicht passenden Daten (wie z.B. die Geokoordinaten für Verwaltungsgebiete) die entsprechenden Felder leer sind [*WG].

Der Gazetteer von [*WG] enthält insgesamt 171.021 Einträge. 13.089 Einträge für Deutschland und 1.111 Einträge für Baden-Württemberg. Die Einträge für Deutschland beziehungsweise für Baden-Württemberg wurden extrahiert und separat gespeichert.

Eine weitere Möglichkeit wäre die Verwendung von [*AG]. Dieser ist sehr umfangreich (ca 6 Mio Einträge), enthält jedoch lediglich Namen ohne weitere zusätzliche Informationen wie beispielsweise Bevölkerung oder Längen- / Breitengrad, die eventuell zum Visualisieren hilfreich wären.

Dem Verwenden der beiden Gazetteer entsprechend, wurde die Eingrenzung von geographischen Entitäten von Orten und Ortschaftsteilen auf Verwaltungseinheiten erweitert. Unter geographischen Entitäten sind somit Verwaltungseinheiten, die in den Gazetteer vorhanden sind, zu verstehen – wie bereits in Abschnitt 3.1 beschrieben. Verwaltungseinheiten sind Ortschaften, Städte, Gemeinden, Landkreise und Bundesländer in Deutschland sowie Stadtteile in Baden-Württemberg (vergleiche auch Abschnitt 3.6).

- Eindeutige Identifikationsnummer des geographischen Objektes
- Name des geographischen Objektes (wenn möglich deutscher Name)
- alternative Namen
- Originalnamen in griechisch, kyrillisch oder arabisch
- Art des geographischen Objektes (z.B. Land, Ort etc.)
- aktuelle Bevölkerung
- Breitengrad (bei Orten)
- Längengrad (bei Orten)
- übergeordnetes Land
- übergeordnete Verwaltungseinheit erster Ordnung
- übergeordnete Verwaltungseinheit zweiter Ordnung
- übergeordnete Verwaltungseinheit dritter Ordnung

Abbildung 3.1: Gazetteer Datei Spalten (vergl. [*WG])

3.5 Visualisierung mit OpenJUMP

Im Rahmen dieser Arbeit soll, wie in Abschnitt 1.2 bereits erläutert, eine prototypische Implementierung zur Visualisierung der richtig erkannten geographischen Entitäten erstellt werden. Deshalb soll an dieser Stelle die Softwareplattform JUMP bzw. OpenJUMP kurz vorgestellt werden. Auf die prototypische Implementierung selbst wird in Kapitel 11 näher eingegangen.

Im Jahr 2002 sollte die Firma Vivid Solutions Inc. eine Software zum Zusammenführen unterschiedlicher digitaler Karten von Straßen und Flüssen erstellen. Bei der Entwicklung achtete das Entwicklerteam auf Flexibilität, so dass eine Software entstand, die nicht nur für Straßen und Flüsse, sondern für jegliche geographische Daten (Stromnetze, Straßenkarten, Satellitenbilder, ...) verwendet werden kann. Das Programm wurde JUMP (Java Unified Mapping Plattform) genannt und entwickelte sich zu einem populären, freien geographischen Informationssystem (engl. Geographic Information System - GIS) [*W3].

Nach der initialen Entwicklung wurde die Weiterentwicklung von JUMP durch Vivid Solutions eingestellt. Es entstanden jedoch viele kleine Projekte, in denen JUMP von Anwendern weiterentwickelt und angepasst wurde. Bei den Entwicklern und Anwendern entstand bald der Wunsch nach einer einheitlichen JUMP-Plattform. Es formierte sich das JPP Development Committee, welches die Reorganisation und Zusammenführung der verschiedenen Projekte zu einer einheitlichen Plattform übernahm. Dabei wurde ein Name basierend auf JUMP, „OpenJUMP“, für die open source GIS-Plattform gewählt.

OpenJUMP kann unter [*OJ] heruntergeladen werden. Es ist kostenlos und quelloffen (GPL-Lizenz). Implementiert ist OpenJUMP wie JUMP in Java. Daher wurde entschieden, im Rahmen dieser Arbeit alle Implementierungen in Java durchzuführen, um auf diese Weise das Einbinden in OpenJUMP zu erleichtern. OpenJUMP verfügt über ein Plugin-Konzept. Hierdurch ist ein Einbinden neuer Funktionen relativ leicht möglich. Weitere Informationen zum Plugin-Konzept von OpenJUMP finden sich unter [*PL].

3.6 Untersuchungen

Wie in Anschnitt 3.1 bereits diskutiert soll unter einer geographischen Entität eine Verwaltungseinheit, die in den verwendeten Gazetteer (siehe Abschnitt 3.4) vorhanden ist, verstanden werden. Verwaltungseinheiten beinhalten dabei Ortschaften, Städte, Gemeinden, Landkreise und Bundesländer in Deutschland sowie Stadtteile in Baden-Württemberg. Hierbei ist wie diskutiert zu beachten, dass diese Abgrenzung naiv zu betrachten ist.

Auf Basis dieser Abgrenzung wurden in 450 zufällig ausgewählten Artikeln des STZ mit Hilfe der Gazetteer case-insensitiv alle eventuell geographischen Entitäten herausgesucht und von Hand annotiert. Wie bereits in Abschnitt 3.4 geschildert und in Tabelle 3.2 ersichtlich, sind relativ wenig mehrdeutige Vorkommen zu verzeichnen. Deshalb wurde entschieden, den extrahierten Teil der Datenbasis von der Firma Navteq (im Weiteren mit Nav abgekürzt) gemeinsam mit den Daten für Deutschland aus dem in 3.4 vorgestellten Gazetteer [*WG] (mit Deu im Folgenden bezeichnet) zu verwenden.

Bei der Verwendung beider Gazetteer (Nav & Deu) und einer case-insensitiven Suche ergibt sich ein Verhältnis von 60% geographischen und 40% nicht-geographischen Vorkommen (vergleiche Tab. 3.2), wobei die Hälfte der nicht-geographischen Vorkommen kleingeschrieben sind. Bei case-sensitiver Suche erhält man somit ein Verhältnis von 75% geographischen und 25% nicht-geographischen Vorkommen. Würde man also zur Disambiguierung nur den Gazetteer einsetzen, indem man jeden Kandidaten als geographisch betrachtet, würde man 60% bei case-insensitiver Betrachtung und 75% bei case-sensitiver Betrachtung korrekt bestimmen.

Das heißt, ein case-insensitives Verfahren muss besser als 60%, ein case-sensitives Verfahren besser als 75% sein, damit eine Verbesserung vorliegt. (Baseline case-insensitiv: 60% Acc bzw. 60% F_1; Baseline case-sensitiv 75% Acc bzw. 75% F_1)

Bei der Betrachtung der Untersuchungsergebnisse in Tabelle 3.2 fällt auf, dass fast alle innerhalb eines Artikels mehrfach vorkommenden Kandidaten auch der gleichen Klasse angehören. Das heißt, eine Heuristik, die davon ausgeht, dass eine Bedeutung pro Artikel vorliegt, wäre gut. Das Verhältnis bei mehrfach vorkommenden Kandidaten ist 70% geo und 30% nicht-geo (vergl. Tab. 3.2). Hierbei sind jedoch lediglich in 46% der Artikel überhaupt mehrfach vorkommende enthalten.

In Tabelle 3.3 sind die Untersuchungsergebnisse bezogen auf alle extrahierten Artikel des STZ aufgeführt. Wie zu sehen ist muss beim Betrachten der Untersuchungsergebnisse zu den annotierten Artikeln beachtet werden, dass sich einige Abweichungen zwischen der zufällig gebildeten annotierten Menge und der gesamten Artikelmenge ergaben. Bei den

	Nav	BaWü	Nav & BaWü	Deu	Nav & Deu
Artikel gesamt	450	450	450	450	450
Art. mit Kandidat	306	252	312	438	443
Art. mit mehrfach vorkommenden Kand.	125	111	129	198	208
— die gleich annotiert sind	125	111	129	198	208
Kandidaten gesamt	876	690	936	1613	1785
Kand. die mehrfach in Art. vorkommen	412	355	429	693	742
— und gleich annotiert sind	412	355	429	691	740
— — und geo sind	337	332	354	503	521
— — und nicht-geo sind	75	23	74	188	219
durchschnittlich Kand. pro Art.	1,95	1,53	2,08	3,58	3,97
— in Art. mit mind. einem Kand.	2,86	2,74	3,00	3,68	4,03
geo von Kandidaten	633	613	689	1013	1068
geo von Kandidaten (in %)	72,26	88,84	73,61	62,80	59,83
nicht-geo von Kandidaten	243	77	247	600	717
nicht-geo von Kandidaten (in %)	27,74	11,16	26,39	37,20	40,17
kleingeschriebene nicht-geo Kand.	87	51	91	334	360
nicht-geo ohne kleingeschriebene	156	26	156	226	357
nicht-geo ohne kleingeschr. (in %)	17,81	3,77	16,67	14,01	20,00

Tabelle 3.2: Grundlegende Untersuchungsergebnisse zu den 450 annotierten Artikeln

	Nav	BaWü	Nav & BaWü	Deu	Nav & Deu
Artikel gesamt	15976	15976	15976	15976	15976
Art. mit Kandidat	11562	9155	11785	14719	14963
Art. mit mehrfach vorkommenden Kand.	5066	4160	5303	8540	8915
Kandidaten gesamt	40994	28067	44019	87932	98995
Kand. die mehrfach in Art. vorkommen	19648	15213	21033	39828	43656
durchschnittlich Kand. pro Art.	2,57	1,76	2,76	5,50	6,20
— in Art. mit mind. einem Kand.	3,55	3,07	3,75	5,97	6,62

Tabelle 3.3: Grundlegende Untersuchungsergebnisse zu allen extrahierten Artikel des STZ einschließlich der annotierten Artikel

annotierten Artikeln sind beispielsweise in 98,4% der Artikel Kandidaten enthalten. Auf alle extrahierten Artikel bezogen sind es jedoch nur 93,7%. Bei den annotierten Artikeln ergab sich außerdem ein Schnitt von 3,97 Kandidaten pro Artikel, im Gegensatz zu 6,2 Kandidaten pro Artikel auf alle extrahierten Artikel bezogen.

Kapitel 4

Verwandte Arbeiten

Nachdem im letzten Kapitel auf verschiedene Aspekte der Disambiguierungsproblematik eingegangen wurde, sollen in diesem Kapitel einige verwandte Arbeiten vorgestellt werden.

Hierzu wird zunächst auf eine Arbeit eingegangen, die sich direkt mit der Disambiguierung befasst. Darauf folgend wird eine Arbeit aus dem Bereich Geo-Coding (siehe Abschnitt 4.2) vorgestellt. Außerdem wird auf geographische Suchmaschinen eingegangen, da diese zum Teil ebenfalls die Disambiguierungsproblematik mitbetrachten. Weiter werden Arbeiten aus dem Bereich Eigennamenerkennung vorgestellt, da die Problematik der Eigennamenerkennung eng verwandt mit der Problemstellung der Disambiguierung ist (siehe Kapitel 5). Abgeschlossen wird dieses Kapitel durch eine Zusammenfassung der für diese Arbeit relevanten bzw. interessanten Aspekte aus den vorgestellten Publikationen.

4.1 Disambiguierung

Die Arbeit [GM05] befasst sich damit, in Zeitungsartikeln Ortsnamen zu disambiguieren, wie dies auch im Rahmen dieser Arbeit geschehen soll. In [GM05] wird zum Auflösen der Mehrdeutigkeiten ein statistischer Klassifikator mit Hilfe eines Gazetteers und maschinell getaggten Texten erstellt. Das heißt, dass linguistische Werkzeuge verwendet werden, da die Texte, auf denen gelernt wird, zunächst getaggt werden, um die Tags als Feature beim Lernen zu verwenden. In dieser Arbeit hingegen sollen wenn möglich keine linguistischen Tools verwendet werden. Als weiterer Unterschied zu dieser Arbeit ist zu nennen, dass in [GM05] lediglich geographisch / geographische Mehrdeutigkeiten betrachtet werden, was zudem auf englischen Texten geschieht.

Publikationen, die explizit geographisch / nicht-geographische Mehrdeutigkeiten behandeln, konnten bei den Recherchen zu dieser Arbeit nicht gefunden werden. Es gibt jedoch diverse Arbeiten im Bereich Geo-Coding, in denen Disambiguierung, zum Teil auch geo / nicht-geo Disambiguierung, mitbetrachtet wird. Einige dieser Arbeiten werden im Abschnitt 4.2 vorgestellt. Außerdem gibt es sehr viele Untersuchungen im Bereich Eigennamenerkennung, die bis in den Bereich der Disambiguierung hineingehen oder zumindest für das Auflösen von Mehrdeutigkeiten relevante Verfahren beschreiben. Auf einige dieser wird in Abschnitt 4.4 eingegangen.

4.2 Geo-Coding

Wie schon erläutert gibt es im Bereich Geo-Coding einige Arbeiten, in denen die Disambiguierung, zum Teil sogar die geographische / nicht-geographische Disambiguierung, mitbetrachtet wird.

Unter Geo-Coding wird der Prozess verstanden, Verknüpfungen zwischen geographischen Lokationen und Webseiten zu bestimmen. Dabei soll die Webseite relevante Informationen zu den geographischen Lokation aufweisen. Ein Web-Dokument kann hierbei mit einer oder mehreren Lokationen verknüpft werden (vergleiche [MCS+05] S. 2).

Beispielhaft für Arbeiten aus dem Bereich Geo-Coding soll die Arbeit [AHSS04] vorgestellt werden. In [AHSS04] wird eine Möglichkeit beschrieben, Geo-Coding auf englischsprachigen Internetseiten durchzuführen. Dabei werden zwei Schwerpunkte gesetzt. Zum einen die „Ortserkennung“ und zum anderen die Bestimmung des geographischen Fokus der Internetseite. Im Folgenden soll lediglich der erste Schwerpunkt betrachtet werden, da nur dieser für diese Arbeit relevant ist.

Bei der „Ortserkennung“ wird in [AHSS04] unterschieden zwischen „source geography“ (der Herkunftsort der Internetseite) und „target geography“ (die im Inhalt der Seite beschriebenen Orte). In [AHSS04] ist der Fokus auf das Betrachten der „target geography“ gerichtet.

Der verfolgte Ansatz basiert auf dem Erkennen von eventuell geographischen Kandidaten mit Hilfe eines (weltweiten) Gazetteers, der 40.000 Städtenamen enthält (vergl. [MCS+05] S. 2). Der Gazetteer ist hierarchisch („part-of“ Hierarchie) sowie „Meta“-verknüpft aufgebaut. Er enthält Einwohnerzahlen der Ortschaften, verschiedene Schreibweisen der Namen der geographischen Entitäten, einschließlich Abkürzungen. Außerdem beinhaltet der Gazetteer eine Non-Geo-Liste für Worte, die sehr häufig mit einem nicht-geographischen Sinn vorkommen. Diese brauchen im Disambiguierungsverfahren mehr Anhaltspunkte, um als Ortschaft erkannt zu werden, als Kandidaten, die nicht in dieser Liste stehen.

Diese Non-Geo-Liste wurde anhand eines annotierten Korpus erstellt. Dabei kamen zwei Kriterien zur Anwendung. Namen, die öfter als 100 mal im annotierten Korpus vorkommen und kleingeschrieben sind, werden in die Liste aufgenommen. Außerdem werden die Kandidaten in die Non-Geo-Liste aufgenommen, deren Vorkommen im Korpus größer als die Einwohnerzahl des entsprechend benannten Ortes ist. Anschließend an die maschinelle Erstellung der Non-Geo-Liste wurden Korrekturen von Hand vorgenommen [AHSS04] S. 276.

Ein sogenannter „Geotagger“ sucht alle im Gazetteer und im Text vorkommenden Entitäten und markiert diese, wenn sie mehr als 500.000 Einwohner aufweisen. Die minimale Größe, die positive Auswirkungen hat, sind 5.000 Einwohner (vergl. [MCS+05] S. 2). Beim Heraussuchen der Entitäten wird Groß- und Kleinschreibung nicht beachtet. Im Gazetteer enthaltene abkürzende Schreibweisen kommen in diesem Schritt nicht zur Verwendung (erst beim Auflösen der geo-geo Mehrdeutigkeiten), da diese zu viele Mehrdeutigkeiten hervorrufen würden.

Interessant für diese Arbeit ist die Verwendung eines Gazetteer zum Auffinden der Kandidaten, wie in [AHSS04] durchgeführt. Auch dass die Einwohnerzahlen als Selekti-

onskriterium verwendet werden, stellt einen relevanten Aspekt für diese Arbeit dar. Der Schwerpunkt von [AHSS04] liegt im Gegensatz zu dieser Arbeit auf geo / geo Mehrdeutigkeiten. Außerdem wird auf englischen Texten gearbeitet, wobei interessant ist, dass der „Geotagger“ trotzdem case-insensitiv arbeitet.

4.3 Geographische Suchmaschinen

Im Weiteren werden Publikationen aufgeführt, die Implementierungen von geographischen Suchmaschinen vorstellen. Bei diesen wird das Geo-Coding mitbetrachtet, weshalb auch Aspekte der Disambigierung behandelt werden. Angemerkt sei an dieser Stelle, dass auch Google, Yahoo und weitere Suchmaschinen Lokationssuchoptionen eingeführt haben, genauere Informationen zu den zugrunde liegenden Verfahren aber nicht veröffentlicht sind (vergl. [MCS+05] S. 2).

In der Arbeit [S¨01] wurde versucht, Ortsinformationen aus Meta-Tags und RDF-Beschreibungen (RDF - Resource Descrtiption Framework) von Webseiten zu extrahieren. Dabei wurde auf das Meta-Datenformat Dublin Core zurückgegriffen [S¨01] S. 27. Die geringe Verbreitung von Webseiten mit umfassenden Meta-Informationen führte jedoch lediglich zu begrenztem Erfolg dieses Lösungsansatzes. Quasi als Fortsetzung dieser Arbeit wird in [Mih03] untersucht, welche weiteren Möglichkeiten bestehen, Webseiten mit Ortsbezug aufzufinden und zu sammeln. Eine Implementierung eines entsprechenden Suchmaschinenroboters wird außerdem vorgestellt.

Herausgearbeitet wird in [Mih03], dass die URL, der Titel, die Meta-Tags, der Webseitentext und die Hyperlinks die wichtigsten Teile einer Webseite bei der Suche nach Ortsinformationen darstellen [Mih03] S. 31. Das grundlegende Vorgehen in [Mih03] ist dabei das Herausfiltern von Kandidaten, die eine Ortsinformation darstellen können und das anschließende Überprüfen dieser anhand einer Geo-Datenbank, deren Inhalt auf dem Getty Thesaurus of Geographic Names (TGN) basiert [Mih03] S. 11. Zum Bestimmen der Kandidaten werden je nach Webseitenteil unterschiedliche Methoden verwendet bzw. untersucht, wie im Folgenden aufgeführt (vergl. [Mih03] S. 31):

- Untersuchen aller Eigennamen
- Untersuchen der ersten n oder der häufigsten n Eigennamen
- Testen der Eigennamen auf spezielle Wörter oder Endungen, beispielsweise „Theater“, „70569“ oder „...straße“.
- Anwenden von einfachen, sematischen, kontextbezogenen Regeln

Beim Betrachten des Webseitentextes werden insbesondere die semantischen, kontextbezogenen Regeln verwendet. In Tabelle 4.1 sind einige Regeln wie sie dabei zum Einsatz kommen beispielhaft dargestellt (vergl. [Mih03] S. 28ff).

Festzuhalten ist also, dass im Gegensatz zu dieser Arbeit der Gazetteer in [Mih03] zum Überprüfen und nicht zum Selektieren von Kandidaten verwendet wird. Das Auflösen

Regel	Beispiel
Wort Ortsinfomation	in Stuttgart
Wort Ortsinfo Wort	in Stuttgart lebend
Wort Wort Ortsinformation	Nähe von Stuttgart

Tabelle 4.1: Regeln, wie sie in [Mih03] zum Selektieren von Kandidaten in Webseitentexten herangezogen werden. Zusätzlich ist jeweils ein Beispiel für die Regeln angegeben.

von geographisch / nicht-geographischen Ambiguitäten wird außerdem lediglich als kleine Teilaufgabe betrachtet, da es eher darum geht, überhaupt geographische Informationen aufzufinden und für Suchmaschinen bereit zu stellen.

In [MCS+05] wird eine Suchmaschinenimplementierung vorgestellt, die sich an den Ansätzen von [AHSS04] (siehe Abschnitt 4.2) orientiert. In [MCS+05] wird jedoch nicht wie bei [AHSS04] auf englischen Texten gearbeitet, sondern die .de-Domäne betrachtet.

Ähnlich dem Seitenschwerpunkt in [AHSS04] wird in [MCS+05] ein geographischer Fingerabdruck der Internetseiten erstellt. Hierzu werden geographische Marker extrahiert, wie beispielsweise Adressen, Telefonnummern und Ortsnamen. Bei der Untersuchung von Ortsnamen werden auch geographisch / nicht-geographische Mehrdeutigkeiten mitbetrachtet. Dem Ansatz von [AHSS04] ähnlich wurden Termlisten erstellt, die aussagen, ob ein Städtename öfter oder sehr selten mehrdeutig ist. Anhand dieser Listen werden zunächst die sehr selten mehrdeutigen Kandidaten aufgelöst. Anhand der bereits aufgelösten Kandidaten werden die sogenannten schwachen Kandidaten mit Hilfe von Heuristiken disambiguiert. Zusätzlich werden „Killerterme“ und „Validierungsterme“ eingesetzt, die, wenn sie in der Nähe eines Kandidaten vorkommen, dafür sorgen, dass dieser auf jeden Fall bzw. auf keinen Fall als geographisch betrachtet wird.

Wie bei [AHSS04] und wie es in dieser Arbeit geschehen soll, wird in [MCS+05] ein Gazetteer zum Auffinden der Entitäten verwendet. Beim Heraussuchen der Entitäten wird wie in [AHSS04] case-insensitiv gearbeitet.

4.4 Eigennamenerkennung

Die Problematik der Eigennamenerkennung ist eng verwandt mit der Problematik der Disambiguierung (siehe Kapitel 5). Deshalb soll in diesem Abschnitt auf zwei Arbeiten aus dem Bereich der Eigennamenerkennung eingegangen werden.

In [LBC04] wird ein auf SVM basierendes System (siehe Abschnitt 6.1) zur Informationsextraktion, GATE-SVM, vorgestellt. Eine spezielle Variante von SVM („SVM for uneven margins“) kommt in GATE-SVM zum Einsatz. Dies geschieht, um bei kleinen Trainingsmengen ungleiche Verteilungen von positiven und negativen Beispielen abzufangen. Das System kann sowohl für Slotfilling als auch für die Eigennamenerkennung verwendet werden [LBC04] S. 1.

Zum Erlernen der Klassifikatoren zur Eigennamenerkennung werden zunächst mit Hilfe des Programms ANNIE linguistische Feature berechnet. Die zum Lernen benötigten Vek-

toren (vergl. Abschnitt 6.1) werden dann anhand der Wörter in einem Wortfenster um den Kandidaten und mit den linguistischen Feature, die mit ANNIE bestimmt wurden, gebildet. Dies sind Lemma, Pos, Case, ein Lookup in einem Gazetteer und eine NER-Regelprüfung. Die linguistischen Feature werden alle mit einem Gewicht von eins versehen. Bei den Worten aus dem Wortfenster wurden zwei Varianten verwendet (vergl. [LBC04] S. 4) – zum einen ebenfalls eine Gewichtung von eins bei allen Worten, zum anderen ein Gewicht in Abhängigkeit vom Abstand zum Kandidaten (1/*Anstand*).

Zur Eigennamenerkennung werden anhand der beschriebenen Vektoren zwei SVM-Klassifikatoren für jeden Entitätstyp erlernt. Einer für den Anfang und einer für das Ende des Eigennamens. Laut [LBC04] stellt dies im Vergleich zu anderen Verfahren eine erhebliche Vereinfachung dar. In anderen Systemen werden beispielsweise für den Anfang, das Ende und den Mittelteil eines Eigennamens und zusätzlich für Eigennamen, die ein Wort lang sind, Klassifikatoren berechnet (vergl. [LBC04] S. 3).

Der letzte in [LBC04] zur Bestimmung der Eigennamen durchgeführte Schritt ist das Selektieren von Start-Ende-Paaren aus den Klassifikationsergebnissen. Dies geschieht anhand der Länge der Wortsequenzen. Ergeben sich mehrere Start-Ende-Paare für eine Wortsequenz, wird der Entitätstyp gewählt, dessen Start-Ende-Paar die höchsten SVM-Klassifikationswerte aufweist [LBC04] S. 6.

Für diese Arbeit ist die Gewichtungsvariante der Worte in Abhängigkeit vom Abstand zum Kandidaten interessant. Es bleibt jedoch zu prüfen, ob diese auf englischen Texten funktionierende Gewichtungsmethode auf das Deutsche übertragbar ist. In [LBC04] werden linguistische Feature eingesetzt, was in dieser Arbeit, wenn möglich, vermieden werden soll. Auch wird ein Gazetteer im Unterschied zu dieser Arbeit zum Bilden der Feature und nicht zum Selektieren der Entitäten verwendet.

In [DFM+04] wird ein überwacht lernendes System zur Identifizierung von Genen und Proteinen in englischsprachigen, biomedizinischen Texten vorgestellt. Es basiert auf der „Maximum-Entropy Method“ und lernt sogenannte „Maximum Entropy Markov Models“ anhand einer großen Zahl Feature (vergl. Anschnitt 5).

Unterschieden wird in [DFM+04] dabei eine „Closed Section“ und eine „Open Section“. In der „Closed Section“ werden ausschließlich kontextuelle Eigenschaften verwendet. Hierbei sind die zugrunde liegenden Feature-Typen Worte, Zeichen Substrings, Worteigenschaften, Pos-Tags, Abkürzungen und Eigennamentags bezüglich der Worte in der Umgebung. Der Feature-Typ Worteigenschaften ist eine vereinfachte Repräsentation der Worte und enthält Attribute wie die Wortlänge und ob Groß- / Kleinschreibung, Zahlen oder Sonderzeichen im Wort enthalten sind [DFM+04] S. 1. Alle diese Feature-Typen werden einzeln oder auch kombiniert verwendet, um neue Feature zu bilden [DFM+04] S. 2.

In der „Open Section“ werden zusätzlich zu den beschriebenen Feature-Typen aus der „Closed Section“ externe Feature eingesetzt. Hierbei handelt es sich unter anderem um verschiedene Wörterbücher und um eine Web-Anfrage-Technik, sowie weitere Tagger [DFM+04] S. 2.

Leider wird in [DFM+04] wie schon in den meisten anderen vorgestellten Arbeiten auf englischen Texten gearbeitet. Dennoch stellen die verwendeten Featuer-Typen eine

für diese Arbeit sehr interessante Abbildungsmethode dar. In [DFM+04] wird außerdem geschildert, dass die Klassifikationsergebnisse der „Open Section“ nur minimal bessere Ergebnisse liefert als die „Closed Section“. Diese Ergebnisse bestärken darin, dass ein Ansatz ohne linguistische Feature und ohne Wörterbücher oder ähnliche Werkzeuge gute Klassifikationsergebnisse liefern könnte.

4.5 Auswertung / Zusammenfassung

Wie in den letzten Abschnitten deutlich wurde, konnten im Rahmen der Recherchen zu dieser Arbeit keine Publikationen aufgefunden werden, die explizit die Thematik der geographische / nicht-geographischen Disambiguierung behandeln. In verschiedenen Arbeiten wird das Auflösen von geographischen Mehrdeutigkeiten betrachtet. Dabei wird allerdings nur in den Arbeiten [AHSS04] und [MCS+05] das Auflösen von geographischen / nicht-geographischen Mehrdeutigkeiten explizit als Teilaufgabe dargestellt. Bei [AHSS04] geschieht dies für englische Texte und bei [MCS+05] für Seiten der .de-Domäne, wobei unklar ist, ob dies alles deutsche Texte sind.

[AHSS04] und [MCS+05] stellen auch die einzigen Arbeiten dar, in denen ein Gazetteer zum Auffinden der Kandidaten verwendet wird, wie es in dieser Arbeit erfolgen soll. Bei den anderen Publikationen werden Gazetteer – wenn überhaupt – um Feature zu bilden oder als eine Überprüfungsfunktion eingesetzt.

[AHSS04] und [MCS+05] verwenden keinen rein maschinellen Ansatz, was im Rahmen dieser Arbeit (siehe Abschnitt 5) jedoch versucht werden soll. Ein interessanter Aspekt für diese Arbeit aus [AHSS04] ist, Einwohnerzahlen der Ortschaften als Selektionskriterium zu betrachten.

Ein weiterer für diese Arbeit relevanter Aspekt stellt der Ansatz von [LBC04] dar, mit Hilfe von SVM einen Klassifikator zu berechnen. Dieser Ansatz, in [LBC04] ursprünglich für die Eigennamenerkennung gedacht, könnte eventuell für die Disambiguierung übernommen und angepasst werden. In diesem Zusammenhang sind dann auch die in [LBC04] und [DFM+04] verwendeten Repräsentationen (Feature) als sehr interessant zu bezeichnen. Wobei in dieser Arbeit nach Möglichkeit auf linguistische Feature und die Verwendung von weiteren Lexika verzichtet werden soll. Die in [DFM+04] vorgestellten Ergebnisse bestärken darin, dass ein Ansatz ohne linguistische Feature realistisch ist, und lassen die Hoffnung aufkommen, dass kontextuelle Feature (vergleiche Abschnitt 5.2) zu ausreichend guten Ergebnissen führen können.

Kapitel 5

Lösungsansatz

Nachdem im letzten Kapitel auf verschiedene verwandte Arbeiten eingegangen wurde, soll in diesem Kapitel zunächst die Herangehensweise zur Lösung der Aufgabenstellung geschildert werden. Dabei wird der grundlegende Lösungsansatz vorgestellt, bevor Repräsentationsmöglichkeiten und Verfahrensansätze näher beleuchtet werden.

5.1 Herangehensweise

Bei Betrachtung der Aufgabenstellung ergab sich zunächst der in Abbildung 5.1 dargestellte Ansatz. Aus einem Text sollten mit Hilfe eines Gazetteer durch einfaches Vergleichen von Wortsequenzen die relevanten Textpassagen zur Disambiguierung extrahiert werden (Abb. 5.1 oben). Die Mehrdeutigkeiten dieser Kandidaten sollten dann anhand von einfachen Regeln und – falls notwendig – mit Ausnahmebehandlungen aufgelöst und (mit OpenJUMP) visualisiert werden. Dabei ist zu beachten, dass zur korrekten Visualisierung eigentlich auch ein Auflösen der geo / geo Mehrdeutigkeiten (gestrichelt in Abb. 5.1 dargestellt) notwendig wäre, was jedoch nicht Teil der Aufgabenstellung dieser Arbeit ist.

Um Regeln zum Auflösen von Mehrdeutigkeiten aufzufinden, wurde zu Beginn der Arbeit versucht, Regelmäßigkeiten bei mehrdeutigen sowie nicht mehrdeutigen geographischen Entitäten aufzufinden. Beispielsweise wurde betrachtet, welche Wörter vor einem Kandidaten stehen und ob hieraus direkt auf eine geographische oder nicht-geographische Bedeutung geschlossen werden kann.

Es stellte sich heraus, dass einfache Regeln, wie sie in Abbildung 5.2 aufgeführt sind, zum Auflösen von Mehrdeutigkeiten nicht ausreichen, da immer einige Fälle zu finden waren, bei denen eine solche Regel falsche Zuordnungen erzeugen würde. Bei der Regel „in XY“ beispielsweise würde sich bei „in Zimmern des Hauses“ eine falsche Klassifikation ergeben. Allerdings konnte auch festgestellt werden, dass die Regel „in XY“ neben weiteren Regeln Indikatoren für geographische Entitäten darstellen. Aus den Untersuchungen ließ sich deshalb schließen, dass es sehr wahrscheinlich Regelmäßigkeiten geben würde, anhand derer Mehrdeutigkeiten aufgelöst werden können, welche jedoch keine triviale Form aufweisen. Das heißt, dass sie durch einfaches Betrachten nicht in ausreichendem Maße aufgefunden bzw. erkannt werden können.

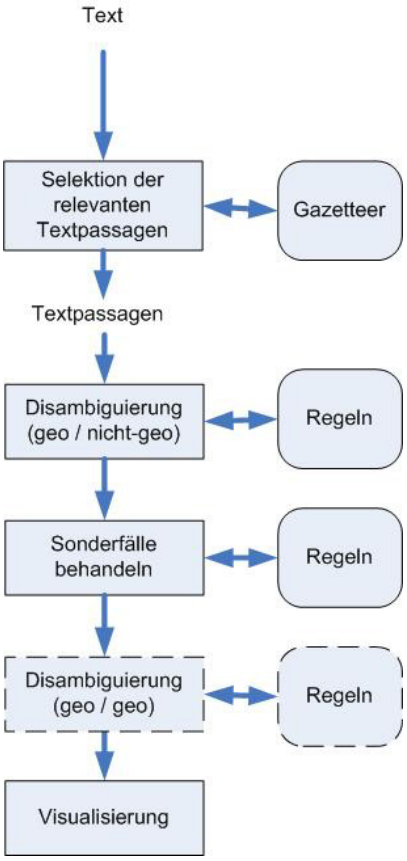

Abbildung 5.1: Grundlegender Ansatz

in XY –> geographisch
zwischen XY und YZ –> geographisch
von XY nach YZ –> geographisch
im XY –> nicht geographisch
auf XY –> nicht geographisch

Abbildung 5.2: Einfache Regeln zur Disambiguierung

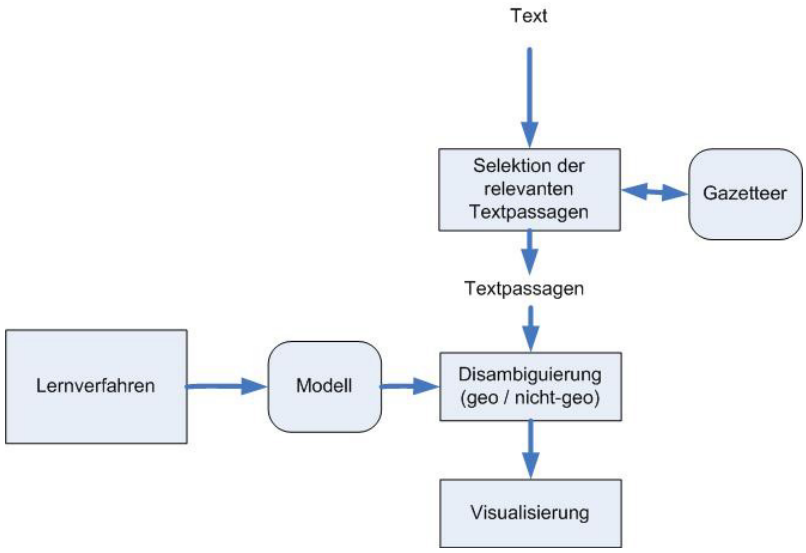

Abbildung 5.3: Ansatz mit maschinellem Lernen

Dementsprechend wurde beschlossen, maschinelles Lernen zum Auffinden von Regelmäßigkeiten heranzuziehen. Hieraus ergibt sich ein etwas veränderter Gesamtansatz. Dieser ist in Abbildung 5.3 dargestellt. Auf das Darstellen der geo / geo Auflösung wurde verzichtet, da sie wie schon erläutert ohnehin nicht Teil der Aufgabenstellung ist. Ebenso wurde darauf verzichtet, das Anwenden von eventuellen Ausnahmeregeln in der Abbildung 5.3 zu visualisieren. Ein Auflösen von Ambiguitäten mit einem maschinell erlernten Modell sollte Ausnahmen schon mit abbilden, wodurch ein gesondertes Erstellen und Abarbeiten von Ausnahmeregeln nicht mehr notwendig sein sollte.

Wie ersichtlich lässt sich dieser neue Ansatz nun in zwei Bereiche aufteilen. Zum einen das Erlernen eines Modells mit dem die Disambiguierung durchführbar wird (Abb. 5.3 links) und zum anderen das Prozedere zum Disambiguieren und Visualisieren eines Textes (Abb. 5.3 rechts). Im Verlauf der Arbeit wurde entschieden, den ersteren Bereich als Schwerpunkt dieser Arbeit zu betrachten.

Wie in Abbildung 5.3 ersichtlich soll mittels eines Gazetteers das Heraussuchen von zu disambiguierenden Entitäten durchgeführt werden. Lässt man sich nicht nur den Kandidaten selbst, sondern eine ganze Textpassage mit den Kandidaten zurückliefern, könnte man anhand der gesamten Textpassage das Auflösen der Ambiguitäten durchführen. Das heißt, dass die Problematik der Disambiguierung sich gewissermaßen auf die der Textklassifikation bzw. der Eigennamenerkennung zurückführen lässt, indem die Kandidaten mit einigen Worten vor ihnen und einigen nach ihnen als zu klassifizierender Text betrachtet werden. In dieser Arbeit untersuchte Ansätze, mit Hilfe eines Klassifikationslernverfahrens zu einem Modell für die Disambiguierung zu gelangen, werden im Abschnitt 5.3 vorgestellt. Im Abschnitt 5.2 wird nun zunächst darauf eingegangen, wie entsprechende Textpassagen gebildet bzw. repräsentiert werden können.

5.2 Repräsentation der Kandidaten

Wie erläutert lässt sich das Auflösen von Mehrdeutigkeiten auf die Textklassifikation zurückführen. Um dies zu bewerkstelligen, müssen die fraglichen Kandidaten und ihr Kontext in eine repräsentierende Form gebracht werden, so dass sie durch Eigenschaften (Feature) beschrieben werden. Dabei muss versucht werden, möglichst viel (relevante) Information abzubilden und gleichzeitig grundlegende Möglichkeiten zur Generalisierung zu schaffen.

In Abbildung 5.4 ist ein kompletter Artikel aus dem STZ dargestellt. Kandidaten zur Disambiguierung sind fett gedruckt. Ebenso in Abbildung 5.5, welche einen Artikelanfang zeigt. Um Missverständnissen vorzubeugen, soll an dieser Stelle nochmals explizit gesagt werden, dass unter einem Kandidaten die Wortsequenz verstanden werden muss, die disambiguiert werden soll. In Abbildung 5.4 sind dies mit „STUTTGART“, „Reuter“ und „Stuttgart“ lediglich Sequenzen der Wortlänge eins. In Abbildung 5.5 ist mit „Bad Reichenbach“ eine Sequenz mit zwei Worten zu finden.

Eine sehr einfache Repräsentation der Kandidaten und ihres textuellen Kontextes wäre nun, eine Anzahl von x Worten vor dem Kandidaten und eine Anzahl von y Worten nach dem Kandidaten heranzuziehen. Bei kleinen Werten für x und y ergäbe sich hierdurch eine Unabhängigkeit von der Länge und der Struktur des gesamten Textes. Dies ist vorteilhaft, da Zeitungsartikel beispielsweise eine andere Textlänge und Textstruktur aufweisen als Internetseiten. Zwei der drei Kandidaten aus Abbildung 5.4 und der Kandidat aus Abbildung 5.5 sind in Tabelle 5.1 mit einem solchen einfachen Kontext für x=4 und y=2 dargestellt.

Porsche-Flitzer werden um 2,2 Prozent teurer **STUTTGART** (**Reuter**) . Der Sportwagenhersteller Porsche AG erhöht mit dem Erscheinen der 1992er-Modelle die Preise . Die Modellreihe 911 werde im Durchschnitt um 2,2 Prozent teurer , sagte ein Sprecher der Firma gestern in **Stuttgart** . Die neue Baureihe 968 sei bereits in die Fertigung gegangen , und am Ende des Jahres beginne auch der Bau der neuen 928 GTs . Außerdem habe Porsche eine Gebrauchtwagen-Garantie für alle Serienmodelle vom Jahrgang 1986 an eingeführt .

Abbildung 5.4: Beispielartikel 1 (Quelle: STZ)

Das Hermann-Traub-Stadion in **Bad Reichenbach** bleibt weiter gesperrt . [...]

Abbildung 5.5: Beispielartikel 2 (Beginn eines Artikel) (Quelle: STZ)

um	2,2	Prozent	teurer	**STUTTGART**	(	Reuter
Prozent	teurer	STUTTGART	(	**Reuter**	)	.
	Das	Hermann-Traub-Stadion	in	**Bad Reichenbach**	bleibt	weiter

Tabelle 5.1: Einfache Repräsentation von Kandidaten für die Disambiguierung mit vier Worten vor und zwei Worten nach den Kandidaten.

b4um	b32,2	b2Prozent	b1teurer	**STUTTGART**	a1(	a2Reuter
b4Prozent	b3teurer	b2STUTTGART	b1(	**Reuter**	a1)	a2.
b4	b3Das	b2Hermann-Traub-Stadion	b1in	**Bad Reichenbach**	a1bleibt	a2weiter

Tabelle 5.2: Repräsentation zur Disambiguierung, bei der die Wortposition mitabgebildet wird.

Bei der Repräsentation des Kontextes und der Kandidaten können allerdings auch Änderungen vorgenommen werden, damit beispielsweise die relative Position der Worte zum Kandidaten mitbetrachtet werden. Hierzu können den Worten beispielsweise Präfixe angehängt werden, wie in Tabelle 5.2 dargestellt. a1 steht dann für after1 und b3 für before3, also für das dritte Wort vor dem Kandidaten. Beim dritten Beispiel in Tabelle 5.2 ist zu beachten, dass es kein b4 gibt und deshalb lediglich der Präfix b4 vorhanden ist, wodurch deutlich wird, dass es sich hier um einen Textanfang handelt.

Unter einem „Feature-Pattern“ versteht man nun in diesem Zusammenhang beispielsweise b1 als immer das Wort vor dem Kandidaten. Dementsprechend sind in den abgebildeten Beispielen die „Wort-Feature-Pattern“ b4 bis b1 sowie a1 und a2 verdeutlicht. Ein Feature, beispielsweise von b1, ist dann b1teuer oder b1in.

Der Kandidat selbst ist in den Beispielen allerdings ebenfalls als ein „Wort-Feature-Pattern“ behandelt worden. Dies ist bei genauerer Betrachtung nicht sinnvoll, da bei unterschiedlicher Bedeutung desselben Kandidaten kein Unterschied entsteht und außerdem keine Generalisierungsmöglichkeit vorhanden ist. Deshalb wurden zwei Feature-Pattern für den Kandidaten in Betracht gezogen. Das „Case-Pattern“ und das „Suffix-Pattern“, da diese mehr Potential zur Verallgemeinerung bieten.

Das Suffix-Feature-Pattern soll alle Suffixe der Länge zwei bis fünf des Kandidaten enthalten. Auf diese Weise wäre beispielsweise eine Verallgemeinerung der Orte Böblingen und Geißlingen durch lingen möglich. Falls der Kandidat kürzer als fünf Buchstaben ist, sollen die entsprechenden Endungen einfach weggelassen werden.

Stuttgart	Xxx
Bad Reichenhall	Xxx Xxx
Stuttgart/Bad Cannstatt	Xxx/Xxx Xxx
MÜNCHEN	XX
weil	xx
Breisgau-Hochschwarzwald	Xxx-Xxx

Tabelle 5.3: Beispielhaft dargestellte Repräsentationsmöglichkeiten des Case-Feature-Pattern.

Das Case-Feature-Pattern setzt den Kandidaten in eine Repräsentation um, die Groß- / Kleinschreibung, mehrere Worte und Sonderzeichen abbildet. Jede Repräsentationsmöglichkeit eines Case-Feature-Pattern wird beispielhaft in Tabelle 5.3 gezeigt. Selbstverständlich kann das Prinzip des Case-Feature-Pattern und des Suffix-Feature-Pattern nicht nur für den Kandidaten angewendet werden.

Wie in Abbildung 5.2 gezeigt und bereits diskutiert, sind spezielle Textkonstruktionen wie beispielsweise „von Stuttgart nach München“ oder „zwischen Stuttgart und München“ ebenfalls Indikatoren für die Disambiguierung. Hieraus ergibt sich eine weitere, zusätzlich sicher sinnvolle Art der Repräsentation – Feature-Pattern, die Wortkombinationen enthalten bzw. abbilden. Für die zwei aufgeführten Beispiele wären zum Beispiel die Kombinationen 1b,1a für den ersten Kandidaten (Stuttgart) und 3b,1b für den zweiten Kandidaten (München) sinnvoll. Die Kombination 2b,1b,1a macht beispielsweise bei „Ludwigsburg,

Kornwestheim und Stuttgart“ zum Disambiguieren von Kornwestheim Sinn. Weitere eventuell hilfreiche Kombinationen wären 2b,Kandidat und Kandidat,2a.

Wie schon diskutiert lassen sich die verschiedenen Feature allgemein für alle Möglichkeiten durch Feature-Pattern beschreiben, deren Ausprägungen für jedes einzelne Beispiel dann die Feature selbst sind. Das Bilden einer Repräsentation mit Hilfe von Feature-Pattern bringt den Vorteil mit sich, diese allgemein implementieren zu können. Dementsprechend sind sie für verschiedenste Verfahren leicht zu verwenden und auf einzelne Anforderungen der Verfahren anpassbar. Diesen Aspekten entsprechend wurden im Rahmen dieser Arbeit die vorgestellten Feature-Pattern allgemein, für alle Verfahren anwendbar implementiert.

5.3 Verfahrensansätze

In der Textklassifikation werden verschiedenste Lernverfahren verwendet. Die einfachste Variante dieser Verfahren ist das überwachte Lernen bzw. Klassifizieren, wie schon in Abschnitt 2.4 erläutert. Wie in Abschnitt 2.4.1 beschrieben ist SVM beim überwachten Lernen zum Standardverfahren geworden. Daher ist es naheliegend, einen überwachten Ansatz mit SVM zu betrachten und für die Disambiguierungsproblematik zu untersuchen. Näheres zu diesem überwachten Verfahren mit SVM findet sich in Kapitel 6.

Das überwachte Lernen bringt zwei Probleme mit sich. Zum einen wird eine relativ große annotierte Lernmenge für gute Ergebnisse benötigt. Das bedeutet wie schon in Abschnitt 2.4 diskutiert einen großen Aufwand, da diese von Hand erstellt werden muß. Zum anderen muß diese Lernmenge repräsentativ sein, da sonst manche Texte nicht oder falsch klassifiziert werden können.

Im Rahmen dieser Arbeit ist das Annotieren einer großen Lernmenge nicht möglich. Wie eine repräsentative Lernmenge aussehen würde, ist zudem unbekannt. Außerdem wäre es ohnehin wünschenswert, ein Verfahren zu finden, das ohne annotierte Beispiele oder zumindest mit einer lediglich sehr kleinen Bootstrap-Lernmenge auskommt. Denn das würde bedeuten, dass die Erkenntnisse und Verfahren aus dieser Arbeit ohne großen Aufwand auf andere Bereiche, wie z.B. das Disambiguieren von Bergen oder Straßen übernommen werden könnten. In Abschnitt 2.4.2 wurden halb-überwachte Lernverfahren insbesondere das Self-Training als eine Ausprägung dieser Verfahren vorgestellt, die die Problematik der Trainingsmenge adressieren. Im Rahmen dieser Arbeit sollen einige Ansätze dieser Art verfolgt und ihre Brauchbarkeit untersucht werden. Dabei werden zwei Ansatzarten näher untersucht. Zum einen zwei Self-Training-Ansätze, zum anderen ein komplett maschinell arbeitendes Bootstrap-Verfahren.

Das erste Self-Training-Verfahren basiert auf SVM, das zweite auf einem Mehrheitsverfahren. Diese beiden Ansätze werden in den Kapiteln 7 und 8 näher vorgestellt. Das komplett maschinelle Bootstrap-Verfahren basiert auf einem zweistufigen SVM-Verfahren, bei dem jedoch nicht mit einer handannotierten Lernmenge gearbeitet wird, sondern mit einer maschinell erzeugten Lernmenge. Auf diesen Bootstrap-Ansatz wird in Kapitel 9 näher eingegangen.

Kapitel 6

Überwachtes Verfahren

In der Textklassifikation werden verschiedenste Lernverfahren verwendet. Die einfachste Variante dieser Verfahren ist das überwachte Lernen bzw. Klassifizieren, wie schon in Abschnitt 2.4 erläutert. Da SVM ein Standardverfahren ist und unter anderem bei der Eigennamenerkennung sehr gute Ergebnisse liefert, ist die Umsetzung eines überwachten Ansatzes mit SVM naheliegend. In den folgenden Abschnitten soll deshalb zunächst kurz darauf eingegangen werden, was SVM ist bzw. wie es funktioniert. Darauf folgend wird SVMlight, die in dieser Arbeit verwendete SVM-Implementierung, kurz vorgestellt und deren Schnittstellen erläutert. Es schließen sich Erläuterungen zur Implementierung der Umsetzung des Disambiguierungsproblems auf die SVM-Klassifikation an. Abschließend werden einige experimentelle Untersuchungen vorgestellt und diskutiert.

6.1 SVM

Zur Klassifikation von Objekten betrachtet man stets deren Eigenschaften. Oftmals bedient man sich deshalb der Betrachtungsweise, dass jedes Objekt einem Vektor, bestehend aus den Eigenschaften des Objektes, entspricht. Stellt man sich nun die Eigenschaften aller Objekte als Dimensionen eines Raumes vor, so kann man jedes Objekt als einen Punkt in diesem Raum ansehen. Bei der Textklassifikation werden häufig solche sogenannten Vektormodelle und darauf basierende Verfahren zum Klassifizieren angewendet. Dabei wird jeder Text durch einen Vektor der enthaltenen Worte repräsentiert.

Betrachtet man nun den Fall, dass die in einem solchen n-dimensionalen Raum abgebildeten Dokumente zwei verschiedenen Klassen angehören, so kann man annehmen, dass die Dokumente wegen Ähnlichkeiten innerhalb jeder Klassen nicht völlig gemischt, sondern in irgendeiner Weise getrennt voneinander in dem Raum auftreten. Diese Annahme ist stark vereinfacht und zweidimensional in Abbildung 6.1 dargestellt.

Viele Klassifikationslernverfahren versuchen wegen dieser Annahme eine lineare, die zwei Klassen trennende Funktion zu bestimmen. Wie in Abbildung 6.2 angedeutet gibt es hierfür viele Möglichkeiten. Der Ansatz der Support-Vector-Machine (engl. Support Vector Machine - SVM) ist ebenfalls, nach einer Hyperebene zu suchen, die die Objekte der Klassen im Raum teilt [MRS07] S. 238f.

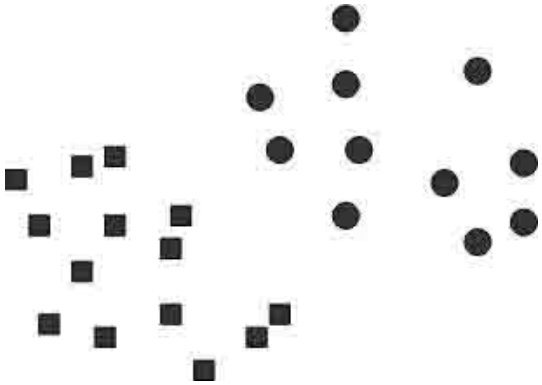

Abbildung 6.1: Objekte aus zwei verschiedenen Klassen als Punkte in einem Vektorraum, stark vereinfacht und zweidimensional dargestellt.

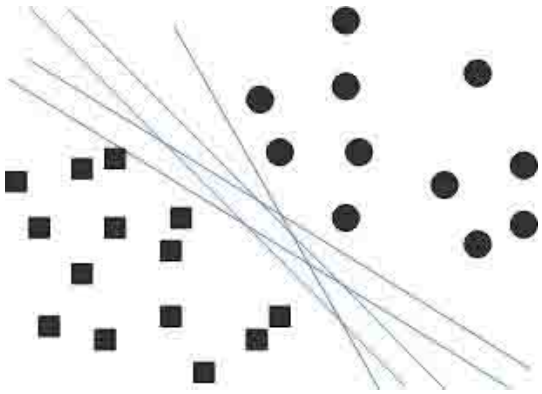

Abbildung 6.2: Möglichkeiten, Klassen im Vektorraum zu trennen – mit den Geraden sind verschiedene Hyperebenen vereinfacht dargestellt, um zu zeigen, dass es sehr viele Möglichkeiten gibt, die dargestellten Klassen zu trennen.

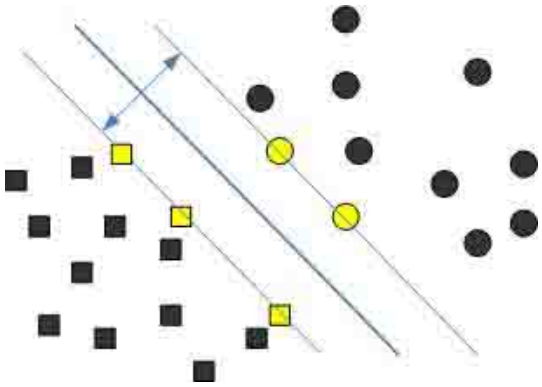

Abbildung 6.3: Optimale Hyperebene – durch Maximierung des Abstandes zu allen Datenpunkten bzw. durch Maximierung des Randes (Doppelpfeil) wird mit SVM eine optimale Hyperebene (dicke Gerade) berechnet. Die hell dargestellten Datenpunkte bilden die Support Vektoren.

Manche Verfahren versuchen anhand von einigen Kriterien einen optimalen linearen Separator zu berechnen. SVM verfolgt zur Bestimmung einer optimalen Trennung den Ansatz, die Hyperebene zu berechnen, die den größten Abstand zu allen Datenpunkten hat. Es entsteht ein leerer Rand um die Klassengrenze herum, dessen Breite maximiert wird. Man spricht daher auch von einem large-margin-Klassifikationsverfahren (engl. „großer Rand“) [*W4]. In Abbildung 6.3 ist die Hyperebene durch die dicke Linie dargestellt. Die Maximierung des margin (Randes) ist angedeutet durch den Doppelpfeil zwischen den mit dünnen Linien dargestellten Grenzen des margin.

Eine SVM maximiert den Rand um die trennende Hyperebene. Das bedeutet, dass die Entscheidungsfunktion einer SVM komplett durch eine (relativ kleine) Teilmenge der Datenpunkte spezifiziert werden kann [MRS07] S. 239. In Abbildung 6.3 sind die Datenpunkte dieser Teilmenge für das Beispielproblem hell dargestellt. Diese Datenpunkte werden Support-Vectoren genannt [MRS07] S. 240. Es ist zu beachten, dass die anderen Datenpunkte keine Rolle bezüglich der gewählten Entscheidungsfunktion spielen [MRS07] S. 241. Die Entscheidungsfunktion bzw. die Stützvektoren können nun also gespeichert und zum Klassifizieren neuer Beispiele herangezogen werden.

Auf eine mathematische bzw. formale Darstellung der Funktionsweise von SVM soll an dieser Stelle verzichtet werden. Bei Interesse kann eine formale Betrachtung und weiterführende Informationen zu SVM [MRS07] S. 239-251 entnommen werden.

6.2 SVMlight

Es gibt einige Implementierungen von SVM. Für diese Arbeit wurde SVMlight von Thorsten Joachims gewählt, welche unter [*SV] zum Herunterladen zur Verfügung steht. SVMlight wurde bereits für diverse Problemstellungen angewendet. Unter anderem zur Textklassifikation, zur Bilderkennung und für bioinformatische sowie medizinische Applikationen.

SVMlight ist in C implementiert und für die nicht-kommerzielle Nutzung kostenlos. In dieser Arbeit wurde die Version 6.01 verwendet.

Im Folgenden soll die allgemeine Nutzung sowie die Schnittstellen von SVMlight kurz erläutert werden. Das in SVMlight enthaltene Klassifikationslernverfahren wird mit

```
svm_learn train.dat model
```

aufgerufen. Die zwei Parameter des Aufrufes „train.dat“ und „model“ stellen dabei Pfad und Dateiname der Datei mit den Trainingsdaten (train.dat) und des Speicherpfades des Modelles dar. Das Klassifikationsverfahren wird mit

```
svm_classify test.dat model predictions
```

aufgerufen, wobei diesmal drei Parameter übergeben werden. Als erstes Pfad und Dateiname der zu klassifizierenden Daten. Dann Pfad und Dateiname des Modells, das zur Klassifikation verwendet werden soll. Als Letztes wird übergeben, in welche Datei in welchem Verzeichnis das Ergebnis der Klassifikation gespeichert werden soll.

Die Dateien „train.dat“ und „test.dat“ müssen zeilenweise die Beispiele enthalten. Jedes Beispiel wird dabei repräsentiert durch ein Label (entweder +1 oder -1), gefolgt von

durch eindeutige Zahlenschlüssel repräsentierten Wörtern. Diesen Schlüsseln muss mit einem Doppelpunkt abgetrennt eine Gewichtung mitgegeben werden. Zu beachten ist außerdem, dass die Wörter nach ihren Schlüsseln sortiert angegeben werden müssen. Im Folgenden ist ein positiv und ein negativ klassifiziertes Beispiel in der für SVMlight benötigten Repräsentation angegeben.

```
+1 3:1 257:1 606:1 711:1 712:1 713:1 # Beispiel 1
-1 3:1 27:1 436:1 3336:1 3337:1 3338:1 3339:1 # Beispiel 2
```

Um die verwendete Zahlenrepräsentation der Wörter zu ermöglichen, muss die Datei „words“ zur Verfügung gestellt werden. Diese enthält in jeder Zeile ein Wort, wobei die Zeilenzahl dem Schlüssel des Wortes entspricht. Jedes Wort darf selbstverständlich nur einmal in der Datei vorkommen.

Die Datei „model“ enthält wie schon erwähnt die Repräsentation des Klassifikationsmodelles. In der Datei „prediction“ werden die Klassifikationsergebnisse abgelegt, wie bereits erörtert . Dies wird wie abgebildet durch Zahlenwerte durchgeführt. Jede Zeile mit einem Zahlenwert steht für die Klassifikation des entsprechenden Beispieles in derselben Zeile in der Datei „test.dat“. Ein positiver Wert entspricht einer positiven Klassifizierung, ein negativer einer negativen Klassifizierung. Der Betrag des Wertes gibt die Gewichtung bzw. Sicherheit der Klassifikation an (je sicherer, um so höher der Betragswert). Beispielhaft sind im Folgenden vier Zeilen einer „prediction“-Datei abgebildet.

```
-0.13296831
-0.26533666
0.54931681
1.0642551
```

6.3 Implementierung

Wie schon in Abschnitt 3.5 beschrieben ist OpenJUMP in Java implementiert. Daher wurde im Rahmen dieser Arbeit auch alles Weitere in Java implementiert. Da SVMlight als ausführbare Datei bzw. als Applikation vorliegt, wird diese aus Java heraus aufgerufen.

Hierfür wurde die Java-Klasse ProcessBuilder verwendet, so dass die Ausgaben von SVMlight mit einem BufferedReader in Java verfügbar sind. Im folgenden Quellcodeauszug ist der Aufruf von SVMlight zum Klassifizieren abgebildet.

```
ProcessBuilder classifyBuilder = new ProcessBuilder( "cmd", "/C",
       "svm_classify", testPathFile, modelPathFile, predictionFilePath );
classifyBuilder.directory( new File(svmPath) );
Process cp = classifyBuilder.start();
BufferedReader cbr =
       new BufferedReader(new InputStreamReader(cp.getInputStream()));
```

Wie in Abschnitt 6.2 bereits geschildert, müssen beim Aufruf von SVMlight verschiedene Dateien übergeben werden. Dies geschieht im Quellcodeauszug in der ersten bzw. zweiten Zeile. Wie zu sehen ist, wurde SVMlight mit den Standardeinstellungen verwendet.

Vor dem Aufruf von SVMlight müssen die benötigten Dateien erzeugt werden. Hierfür wurden mit den Gazetteer (vergl. Abschnitt 3.4) die Kandidaten case-insensitiv gesucht und mit ihren Kontexten extrahiert. Die selektierten Textabschnitte wurden, wie in Abschnitt 5.2 diskutiert, in eine passende Repräsentation umgewandelt, indem die diskutierten Feature-Pattern alle implementiert wurden. Beispielhaft hier nochmal eine Repräsentation eines Kandidaten und dessen Kontextes.

```
b2Jahren b1nach WöCaXxx WöSuen WöSuden WöSurden WöSuorden Norden
a1führende a2und wCb3b1siebzigernach wC2b1b1aJahrennachführende
b3siebziger wCb1a1nachführende b4den wCb2wJahrenNorden wCa2wNordenund
```

In Abschnitt 6.2 wurde bereits erläutert, dass SVMlight eine Repräsentation der Beispiele mit Hilfe von eindeutigen Schlüsselwerten benötigt. Um eine Repräsentation wie dargestellt in die Schlüsselrepräsentation umsetzen zu können, wurde eine Klasse Lexikon implementiert. Diese enthält die Funktion getNumber, welche für ein übergebenes Feature einen eindeutigen Schlüssel zurückgibt. Wird ein Feature übergeben, dem bisher kein Schlüssel zugeordnet ist, wird ein neuer erzeugt und gespeichert. Nachdem alle Feature nachgeschlagen wurden, enthält das Lexikon somit alle Feature. Diese können dann in der Reihenfolge der Schlüsselnummern zeilenweise in die Datei „words“ geschrieben werden.

Bei den jeweiligen Beispielrepräsentationen werden die vom Lexikon zurückgegebenen Schlüsselnummern aufsteigend sortiert und die Beispiele zeilenweise in die jeweilige Trainings- bzw. Analysedatei geschrieben. Passend zu der angegebenen Beispielrepräsentation ist im Folgenden eine für SVMlight passende Repräsentation angegeben.

```
-1 3:1 4:1 156:1 157:1 158:1 159:1 160:1 161:1 162:1 163:1 164:1
165:1 166:1 167:1 168:1 169:1 170:1
```

Am Label „-1“ ist zu sehen, dass es sich um ein Beispiel eines mehrdeutigen Kandidaten (nicht-geo Kandidaten) handelt. Die Gewichtung der Feature ist, wie zu sehen ist, bei allen Feature-Pattern mit einem Gewicht von eins gleich.

6.4 Analyse

Um ein überwachtes Lernverfahren anwenden zu können, wird eine annotierte Lernmenge benötigt. Wie in Abschnitt 3.6 bereits geschildert, wurde eine Menge von Beispielen annotiert. Diese annotierten Beispiele wurden zufällig in 3 Mengen unterteilt (durch Bilden der 3-er-Restklassen der Artikelnummern). Hierdurch wurde erreicht, dass eine Lernmenge, eine Menge zum Analysieren und eine Menge zum Evaluieren vorhanden sind. Um

	Lernmenge	Analysemenge	Evaluierungsmenge
Artikel	159	146	145
Kandidaten	564	560	663
geo von Kandidaten	352	345	373
nicht-geo von Kandidaten	212	215	290

Tabelle 6.1: Daten zu den annotierten Artikeln und ihrer Aufteilung in die Lern-, Analyse- und Evaluierungsmenge.

vergleichbare Ergebnisse zu erhalten, wurden diese drei Mengen bei allen Verfahren entsprechend eingesetzt. In Tabelle 6.1 sind einige Angaben zu den drei Mengen annotierter Beispiele zu finden.

Wie beschrieben wurde sind verschiedene Feature-Pattern auf SVM umgesetzt worden. Um herauszufinden, wie hilfreich diese Feature-Pattern zur Disambiguierung sind, wurden verschiedene Feature-Pattern-Kombinationen zum Erlernen eines Klassifikators herangezogen und die Ergebnisse analysiert.

b3		X	X	X	X	X	X	X
b2	X		X	X	X	X	X	X
b1	X	X		X	X	X	X	X
case	X	X	X		X	X	X	X
suff	X	X	X	X		X	X	X
a1	X	X	X	X	X		X	X
a2	X	X	X	X	X	X		X
Acc (%)	**95,00**	94,29	93,57	92,86	86,25	94,64	94,82	94,46
Pre (%)	94,40	94,08	91,64	94,46	83,33	94,12	94,38	94,10
Rec (%)	97,68	96,81	98,55	93,91	97,10	97,39	97,39	97,10
F_1 (%)	**96,01**	95,43	94,97	94,18	89,69	95,73	95,86	94,10

Tabelle 6.2: Analyse zu sieben Feature-Pattern

Zunächst wurde angenommen, dass eine größere Anzahl von Feature-Pattern zu besseren Ergebnissen führen würde. In den Tabelle 6.3 und 6.4 sind die Werte mit allen 19 implementierten Feature-Pattern sowie bei jeweiligem Weglassen eines Feature-Pattern dargestellt. Bei weiterem Untersuchen stellte sich jedoch heraus, dass sich bei der Kombination der sechs Pattern b2, b1, case, suff, a1 und a2 mit 95% Acc bzw. 96,01% F_1 ein besseres Ergebnis ergab, wie in Tabelle 6.2 zu sehen ist. In Tabelle 6.5 sind weitere Werte für 12 Feature-Pattern dargestellt.

Betrachtet man die Untersuchungsergebnisse, so kann vermutet werden, dass ein geringerer F_1-Wert bei einer Kombination bedeutet, dass das fehlende Pattern ein zur Disambiguierung wichtiges Pattern darstellt. Bei den Kombinationen in Tabelle 6.5 sind die Werte der Kombinationen, in denen case bzw. suff weggelassen wurden, erheblich schlechter. Dies würde auch dazu passen, dass diese Pattern in der mit einem F_1-Wert von 96,01% besten Kombination enthalten sind.

Wie diese Untersuchungen gezeigt haben, ist also die Anzahl der Feature-Pattern nicht proportional zum Klassifikationsergebnis. Dies ist vermutlich darauf zurückzuführen, dass die Feature-Pattern nicht alle unabhängig voneinander sind. Auch wegen der relativ kleinen Trainings- sowie Analysemenge ist sicherlich an mancher Stelle eine Unter- oder Überbewertung einzelner Feature-Pattern bzw. Beispiele möglich. Das heißt, bei der Betrachtung der Ergebnisse muss beachtet werden, dass davon auszugehen ist, dass mit steigender Größe der Lernmenge auch die implizit oder explizit gemachten Schätzungen der zugrunde liegenden Verteilung und somit der Klassifikator besser werden müssten [Lan06] S. 12. Ein Ergebnis mit 96,01% F_1 könnte somit mit einer größeren Lernmenge eventuell verbessert werden. Dies gilt insbesondere, da nur textuelle Feature verwendet werden, bei denen es sehr viele Verschiedene gibt. Wie schon erläutert soll deshalb im Rahmen dieser Arbeit versucht werden, mit Self-Training auf bessere Ergebnisse zu kommen und eventuell sogar lediglich eine kleinere annotierte Lernmenge zu benötigen. Näheres hierzu in den Kapiteln 7 und 8.

Zu der innerhalb der Untersuchungen am besten abschneidenden Pattern-Kombination sind noch folgende Werte zu nennen – von den 560 Beispielen, die zur Analyse herangezogen wurden, sind 532 korrekt und 28 falsch klassifiziert worden. Von den falsch klassifizierten waren 8 als nicht-geo und 20 als geo klassifiziert.

Bezüglich der Feature-Pattern ist festzuhalten, dass mit der Kombination der sechs Feature-Pattern b2, b1, suff, case, a1 und a2 sicher auch bei anderen Verfahren mit guten Ergebnissen zu rechnen ist. Ob eine andere Feature-Pattern-Kombination zu noch besseren Ergebnissen führen würde, kann nicht ausgeschlossen werden, da vom Überprüfen aller Kombinationen wegen des hohen Aufwandes abgesehen wurde. Auch ist nicht auszuschließen, dass bei anderen Verfahren andere Pattern-Kombinationen zu besseren Ergebnissen führen könnten.

b4		X	X	X	X	X	X	X	X	X
b3	X		X	X	X	X	X	X	X	X
b2	X	X		X	X	X	X	X	X	X
b1	X	X	X		X	X	X	X	X	X
b1case	X	X	X	X		X	X	X	X	X
case	X	X	X	X	X		X	X	X	X
suff	X	X	X	X	X	X		X	X	X
a1case	X	X	X	X	X	X	X		X	X
a1	X	X	X	X	X	X	X	X		X
a2	X	X	X	X	X	X	X	X	X	
a3	X	X	X	X	X	X	X	X	X	X
a4	X	X	X	X	X	X	X	X	X	X
b1, Kand	X	X	X	X	X	X	X	X	X	X
b2, Kand	X	X	X	X	X	X	X	X	X	X
a1, Kand	X	X	X	X	X	X	X	X	X	X
a2, Kand	X	X	X	X	X	X	X	X	X	X
b3, b1	X	X	X	X	X	X	X	X	X	X
b1, a1	X	X	X	X	X	X	X	X	X	X
b2, b1, a1	X	X	X	X	X	X	X	X	X	X
Acc (%)	93,57	93,57	93,04	91,96	93,21	92,14	84,46	93,39	93,21	93,75
Pre (%)	92,33	92,33	91,58	89,47	91,60	93,62	81,16	92,08	91,83	92,82
Rec (%)	97,68	97,69	97,68	98,55	97,97	93,62	97,39	97,68	97,68	97,39
F_1 (%)	94,93	94,93	94,53	93,79	94,68	93,62	88,54	94,80	94,66	95,05

Tabelle 6.3: Teil 1 – Analyse zu allen (19) im Rahmen dieser Arbeit implementierten Feature-Pattern

b4	X	X	X	X	X	X	X	X	X	X
b3	X	X	X	X	X	X	X	X	X	X
b2	X	X	X	X	X	X	X	X	X	X
b1	X	X	X	X	X	X	X	X	X	X
b1case	X	X	X	X	X	X	X	X	X	X
case	X	X	X	X	X	X	X	X	X	X
suff	X	X	X	X	X	X	X	X	X	X
a1case	X	X	X	X	X	X	X	X	X	X
a1	X	X	X	X	X	X	X	X	X	X
a2	X	X	X	X	X	X	X	X	X	X
a3		X	X	X	X	X	X	X	X	X
a4	X		X	X	X	X	X	X	X	X
b1, Kand	X	X		X	X	X	X	X	X	X
b2, Kand	X	X	X		X	X	X	X	X	X
a1, Kand	X	X	X	X		X	X	X	X	X
a2, Kand	X	X	X	X	X		X	X	X	X
b3, b1	X	X	X	X	X	X		X	X	X
b1, a1	X	X	X	X	X	X	X		X	X
b2, b1, a1	X	X	X	X	X	X	X	X		X
Acc (%)	93,21	93,57	93,21	93,21	93,21	93,21	93,57	93,39	93,39	93,21
Pre (%)	91,83	92,33	91,83	91,83	91,83	91,83	92,10	91,85	91,85	91,83
Rec (%)	97,68	97,68	97,68	97,68	97,68	97,68	97,97	97,97	97,97	97,68
F_1 (%)	94,66	94,93	94,66	94,66	94,66	94,66	94,94	94,81	94,81	94,66

Tabelle 6.4: Teil 2 – Analyse zu allen (19) im Rahmen dieser Arbeit implementierten Feature-Pattern

b3		X	X	X	X	X	X	X	X	X	X	X	X
b2	X		X	X	X	X	X	X	X	X	X	X	X
b1	X	X		X	X	X	X	X	X	X	X	X	X
b1case	X	X	X		X	X	X	X	X	X	X	X	X
case	X	X	X	X		X	X	X	X	X	X	X	X
suff	X	X	X	X	X		X	X	X	X	X	X	X
a1case	X	X	X	X	X	X		X	X	X	X	X	X
a1	X	X	X	X	X	X	X		X	X	X	X	X
a2	X	X	X	X	X	X	X	X		X	X	X	X
b3, b1	X	X	X	X	X	X	X	X	X		X	X	X
b1, a1	X	X	X	X	X	X	X	X	X	X		X	X
b2, b1, a1	X	X	X	X	X	X	X	X	X	X	X		X
Acc (%)	94,64	93,93	93,75	94,64	92,68	84,64	94,29	94,46	94,46	94,46	94,64	94,46	94,11
Pre (%)	93,63	93,07	91,89	93,39	93,68	81,36	93,59	93,37	93,85	93,61	93,39	93,61	93,09
Rec (%)	97,97	97,39	98,55	98,26	94,49	97,39	97,39	97,97	97,39	97,68	98,26	97,68	97,97
F_1 (%)	95,75	95,18	95,10	95,76	94,08	88,66	95,45	95,61	95,59	95,60	95,76	95,60	95,47

Tabelle 6.5: Analyse zu 12 Feature-Pattern

Kapitel 7

Self-Training-Verfahren (SVM)

Bei statistischen Klassifikationsverfahren muss davon ausgegangen werden, dass mit steigender Größe der Lernmenge auch die implizit oder explizit gemachten Schätzungen der zugrunde liegenden Verteilung und somit der Klassifikator besser werden müssten (vergl. [Lan06] S. 12). Ein Ergebnis von 96,01 F_1, wie beim überwachten Verfahren im letzten Kapitel erreicht, könnte somit mit einer größeren Lernmenge eventuell verbessert werden. Wie schon erläutert soll deshalb im Rahmen dieser Arbeit versucht werden, mit Self-Training (siehe Abschnitt 2.4.2) auf bessere Ergebnisse zu kommen und eventuell sogar lediglich eine kleinere annotierte Lernmenge zu benötigen.

Klassifiziert man mit SVM, erhält man für die klassifizierten Beispiele neben der Klassenzuordnung ein Gewicht. Dieses sagt aus, wie sicher sich SVM mit der Klassenzuordnung dieses Beispieles ist. Bei dem in diesem Kapitel vorgestellten Verfahren wurde der Ansatz verfolgt, diese Gewichte zum Selektieren neuer Beispiele heranzuziehen, um diese iterativ zur Trainingsmenge hinzuzufügen. Auf diese Weise sollte versucht werden, die Ergebnisse des überwachten Verfahrens zu verbessern.

7.1 Algorithmus

Im Abschnitt 2.4.2 wurde die Methodik des Self-Trainings bereits allgemein vorgestellt. Im Folgenden soll nun darauf eingegangen werden, wie diese Methodik mit Hilfe von SVM für die Problematik der Disambiguierung implementiert wurde. Wie im Pseudocode ersichtlich müssen zunächst einige Beispielmengen erstellt werden. Dies wurde auf dieselbe Weise durchgeführt, wie schon beim überwachten Verfahren (siehe Kapitel 6). Im Unterschied zum überwachten Ansatz wird nun in einer Schleife iterativ gelernt, wie im Pseudocode in den Zeilen 6 bis 14 dargestellt.

Das heißt, mit Hilfe von SVM wird ein Modell erlernt, beim ersten Mal basierend auf der handannotierten Lernmenge. Mit diesem Modell werden nun die Klassifikationsdaten klassifiziert. Wie im letzten Abschnitt bereits erläutert liefert SVM hierbei „Sicherheits-Gewichte“ zurück. Anhand dieser können die „am sichersten“ klassifizierten X Beispiele selektiert werden. Wie im abgebildeten Pseudocode ersichtlich, ist es dadurch möglich, diese zur Trainingsmenge hinzuzufügen und den Lernvorgang mit einer nun vergrößerten

Lernmenge erneut zu starten.

```
( 1)  Aufbauen der Trainingsdaten
( 2)  Aufbauen der Analysedaten
( 3)  Aufbauen der Klassifikationsdaten
( 4)  Erstellen des Wörterbuches
( 5)
( 6)  wiederhole {
( 7)      Modell lernen mit SVM anhand Trainingsdaten
( 8)      Klassifikationsdaten mit SVM
( 9)          anhand gelerntem Modell klassifizieren
(10)      besten X (mit höchstem SVM-Betrag) heraussuchen
(11)      zur Trainingsmenge besten X und
(12)          gleiche in Artikel hinzunehmen
(13)      Analysieren der Ergebnisse
(14)  }
```

Wie im Pseudocode dargestellt werden nicht nur die besten X Beispiele, sondern zusätzlich „gleiche im Artikel“ hinzugefügt. Im Abschnitt 3.6 wurde erläutert, dass innerhalb eines Artikels alle Vorkommen einer Entität der gleichen Klasse angehören. Beispielsweise gehören alle Vorkommen von Achim innerhalb eines Artikels zur nicht-geographischen Klasse und alle Vorkommen von Singen zur geographischen Klasse. Wird nun eines dieser Vorkommen sicher klassifiziert, können also alle anderen Vorkommen innerhalb des Artikels ebenfalls klassifiziert werden und zur Trainingsmenge hinzugenommen werden. Dies führt dazu, dass eine größere Anzahl von Kontexten zur Trainingsmenge hinzugefügt werden können.

Wie im Pseudocode ersichtlich gibt es einige Parameter, die am Algorithmus variiert werden können, z.B. die Feature-Pattern, wie viele Beispiele pro Lernrunde zur Lernmenge hinzugefügt werden sollen, usw.. Deshalb wurden einige Untersuchungen durchgeführt, die im Abschnitt 7.2 diskutiert werden.

7.2 Analyse

Das in den letzten Abschnitten vorgestellte Self-Training-Verfahren beruht, wie schon geschildert, direkt auf dem bereits vorgestellten überwachten Verfahren. Deshalb wurde vor allem mit der Feature-Pattern-Kombination b2, b1, case, suff, a1 und a2 gearbeitet, da zu erwarten war, das dies die besten Ergebnisse liefern würde. Anhand von Stichproben mit anderen Feature-Pattern-Kombinationen bestätigte sich diese Annahme.

Eine natürliche Grenze für die Anzahl der Lernrunden gibt es in diesem Verfahren leider nicht. Daher wurde für die Untersuchungen frei gewählt, mit 60 Lernrunden zu arbeiten. In Abbildung 7.1 sind die Acc-Werte in jeder Runde für mehrere Analyseläufe dargestellt. Für die Analyse ist dieselbe Analysemenge wie beim überwachten Verfahren verwendet worden. Als Boot-Menge ist beim Lauf 1 ebenfalls die gleiche Boot-Menge wie beim überwachten

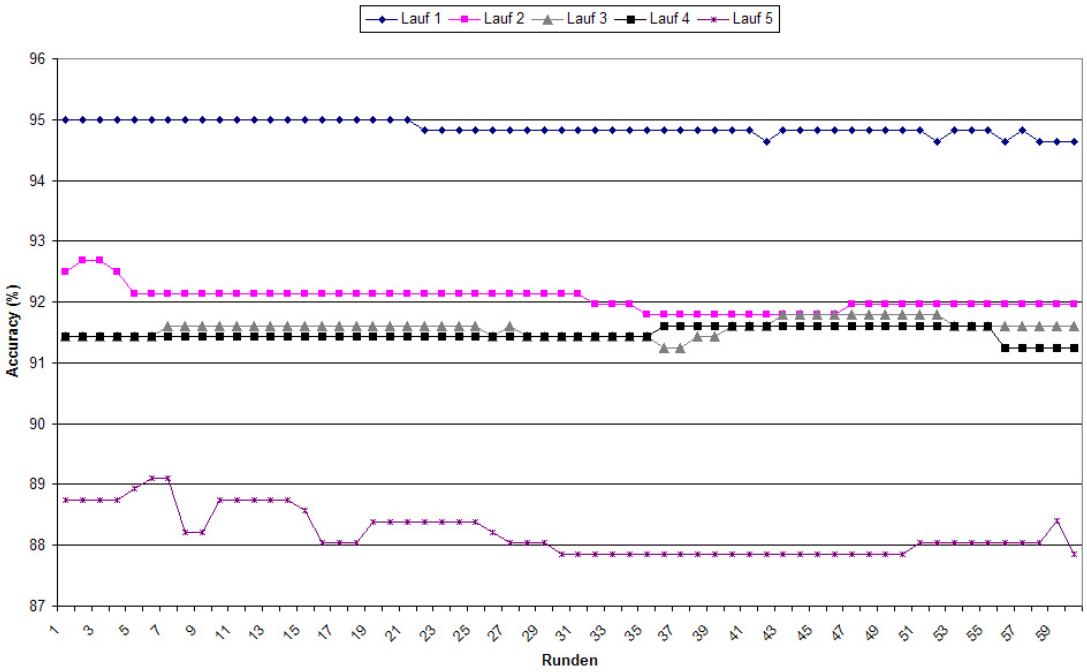

Abbildung 7.1: Analyseläufe – Self-Training mit SVM; die Läufe wurden mit 564, 288, 211, 182 und 116 Beispielen in der Boot-Menge durchgeführt.

Verfahren (siehe 6.4) verwendet worden. Bei den Läufen 2 bis 5 wurden Teilmengen der Boot-Menge aus Lauf 1 verwendet. Die zum Erstellen der neu hinzuzunehmenden Beispiele herangezogene Menge bestand bei allen Läufen aus denselben 5000 zufällig bestimmten Artikeln aus dem STZ. Es wurden in jeder Runde die besten fünf Beispiele bestimmt und diese zusammen mit gleichen im jeweiligen Artikel zur Lernmenge hinzugefügt.

Wie in Abbildung 7.1 zu sehen ist können beim Lauf 1 keine Verbesserungen verzeichnet werden. Bei den Läufen 2, 4 und 5 sind minimale kurzzeitige Verbesserungen vorhanden. Bei Lauf 3 ist eine minimale Gesamtverbesserung zu beobachten.

Beispielhaft ist in Abbildung 7.2 für die Läufe 1 und 5 die Anzahl der Beispiele angegeben, die in jeder Runde zum Erlernen des Klassifikators zur Verfügung stehen. Zusätzlich ist jeweils die Gerade angegeben, die den Verlauf der Beispielzahl ohne das Hinzunehmen von gleichen innerhalb des Artikels ergäbe. In der Abbildung 7.3 ist ebenfalls zu den Läufen 1 und 5 jeweils der höchste und der niederste SVM-Wert der besten fünf Beispiele jeder Lernrunde dargestellt.

Wie in den Abbildungen 7.1 und 7.2 beispielsweise für den Lauf 1 dargestellt ist, wird die Lernmenge stetig vergrößert. Die Klassifikationsergebnisse bleiben dabei zunächst relativ lange Zeit stabil und sinken dann langsam ab. Hieraus lässt sich die Vermutung ableiten, dass bei diesem Verfahren eine Spezialisierung anstelle einer Generalisierung und Erweiterung des Klassifikators stattfindet. Diese Vermutung wird dadurch gestützt, dass beim Lauf 1 in fast allen Lernrunden lediglich Beispiele für Stuttgart hinzugefügt werden. Eine entsprechende Tendenz ist auch bei den übrigen Läufen zu beobachten.

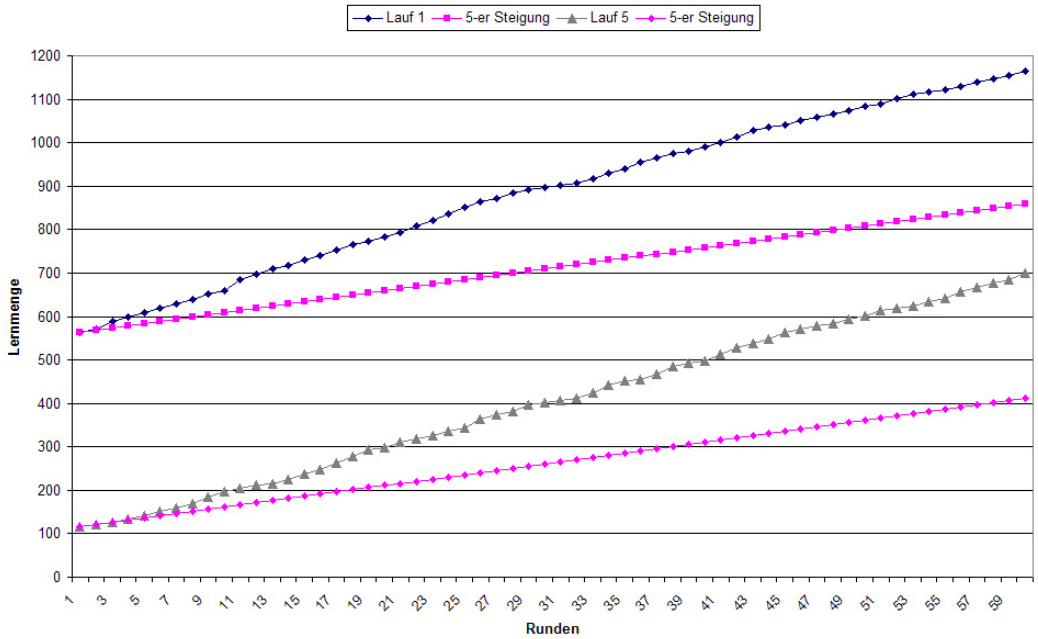

Abbildung 7.2: Analyseläufe – Self-Training mit SVM; die Anzahl der Trainingsbeispiele für alle Lernrunden der Läufe 1 und 5 sind im Vergleich zur Größe der Trainingsmengen bei einer minimalen Selektion von fünf Beispielen pro Runde dargestellt.

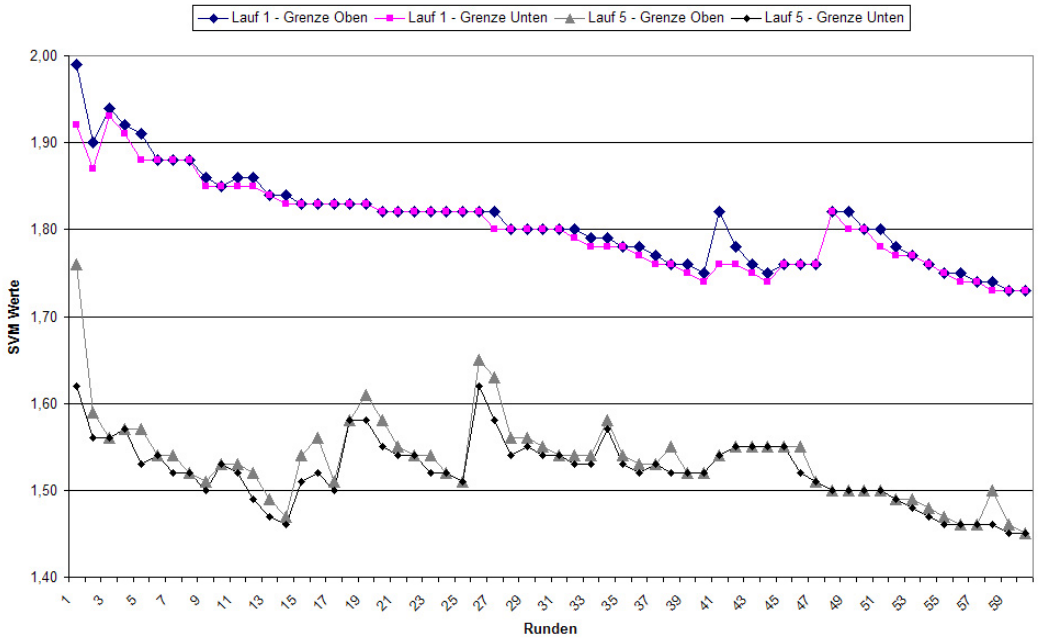

Abbildung 7.3: Analyseläufe – Self-Training mit SVM; für die Läufe 1 und 5 sind die größten und kleinsten Werte der in jeder Runde zur Selektion bestimmten besten fünf Beispiele dargestellt.

Bei den bisher vorgestellten Läufen werden pro Lernrunde die besten fünf Beispiele bestimmt. In weiteren Testläufen wurde dieser Parameter in den Schritten 5, 10, 20 und 30 variiert, um zu überprüfen, ob sich beim Hinzufügen von einer größeren Zahl von Beispielen eine Verbesserung des Verfahrens abzeichnet. Dabei wurde die Boot-Menge aus Lauf 5 verwendet.

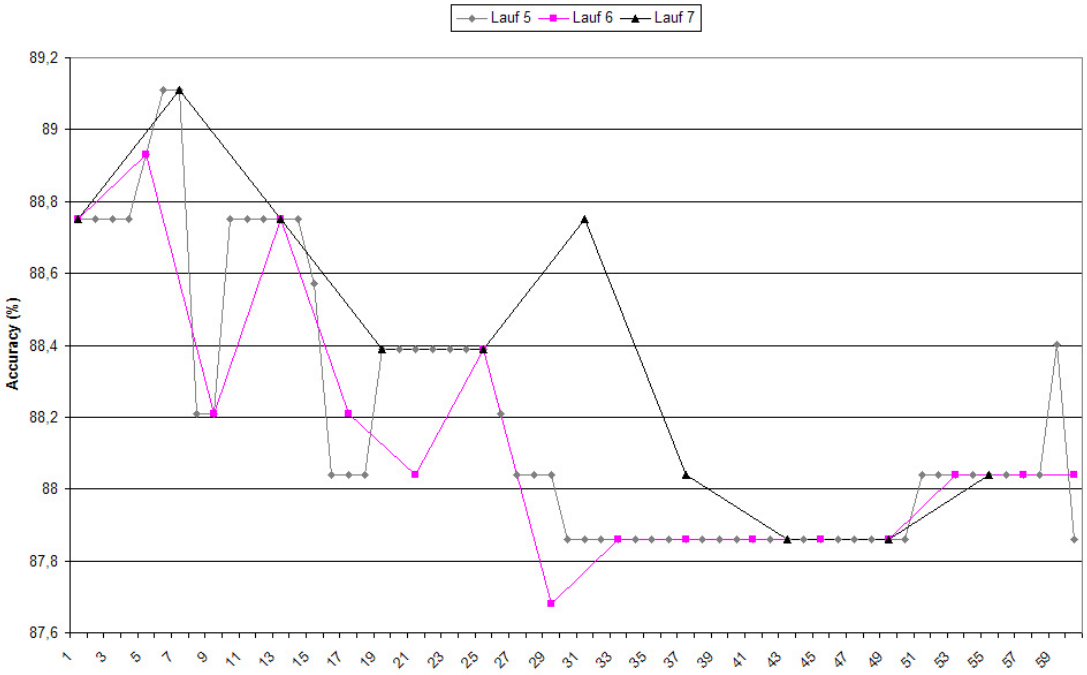

Abbildung 7.4: Analyseläufe – Self-Training mit SVM; dargestellt sind die Acc-Werte der Läufe 5, 6 und 7, wobei die Darstellung der letzten beiden bezüglich der Runden gestreckt ist. Hierdurch soll eine bessere Vergleichbarkeit, basierend auf ähnlichen Größen der Lernmengen, erreicht werden.

In Abbildung 7.4 sind nochmal die Acc-Werte von Lauf 5 dargestellt. Zusätzlich wurden die Acc-Werte der Läufe 6 und 7 dargestellt. Bei diesen wurden die Runden allerdings in gestreckter Form abgetragen. So ist bei Lauf 6 die Runde zwei auf die Runde vier von Lauf 5 abgetragen. Die Runde zwei von Lauf 7 ist auf die Runde acht von Lauf 5 verschoben. Hierdurch sind die Werte mit Bezug auf die Größe der Lernmengen in den jeweiligen Runden besser vergleichbar. Auf die gleiche Weise sind in Abbildung 7.5 die SVM-Werte (Mittelwerte aus den größten und kleinsten Werte der in jeder Runde zur Selektion bestimmten besten Beispiele) zu den Läufen 5, 6 und 7 dargestellt.

Wie zu beobachten ist, ähneln sich sowohl die Verläufe der Acc-Werte sowie der SVM-Werte. Das bedeutet, dass ein vergrößerter Wert für die besten zu wählenden Beispiele nicht zu besseren Ergebnissen führt. Stattdessen kann angenommen werden, dass quasi dasselbe in einer geringeren Rundenzahl durchgeführt wird.

Es kann nicht mit Sicherheit ausgeschlossen werden, dass bei einer höheren Anzahl von Lernrunden das Verfahren noch Lernzuwächse liefert. Dies ist aber unwahrscheinlich,

da die SVM-Werte, alle Lernrunden im Gesamten betrachtet, stark fallen und somit bei weiteren Lernrunden lediglich sehr unsicher klassifizierte Beispiele hinzugenommen werden können.

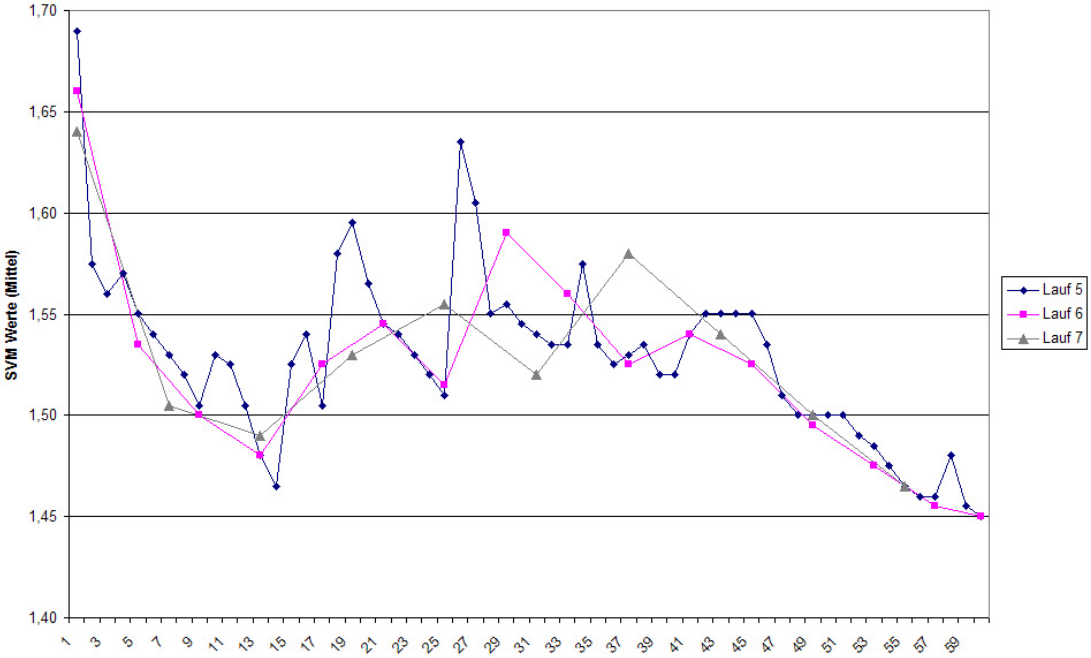

Abbildung 7.5: Analyseläufe – Self-Training mit SVM; dargestellt sind zu den Läufe 5, 6 und 7 die Mittelwerte aus den größten und kleinsten SVM-Werten der in jeder Runde zur Selektion bestimmten besten fünf Beispiele. Dabei wurde die Darstellung der Läufe 6 und 7 bezüglich der Runden gestreckt. Hierdurch soll eine bessere Vergleichbarkeit, basierend auf ähnlichen Größen der Lernmengen, erreicht werden.

Kapitel 8

Self-Training-Verfahren (Mehrheitsverfahren)

Wie schon bei dem im Kapitel 7 vorgestellten Verfahren soll bei diesem Verfahren ebenfalls versucht werden, einen verbesserten Klassifikator zu erhalten, indem durch Self-Training iterativ die Trainingsmenge vergrößert wird.

Betrachtet man verschiedene Feature-Pattern als voneinander unabhängige Indikatoren bezüglich eines zu klassifizierenden Beispieles, so können die einzelnen Feature entweder geographisch oder unentscheidbar bzw. unbekannt oder nicht-geographisch zurückliefern. Ein Feature-Pattern alleine, beispielsweise b1, wäre, wie schon in Abschnitt 5.1 diskutiert, zu unsicher und kann keine Ausnahmen abbilden.

Betrachtet man jedoch mehrere Feature-Pattern und verknüpft ihre Aussagen, müsste die Wahrscheinlichkeit steigen, eine richtige Gesamtindikation zu erlangen. Eine einfache und oft verwendete Möglichkeit, die Voten mehrerer Feature-Pattern miteinander zu verbinden, ist eine Art Mehrheitsentscheid durchzuführen.

Ob ein Feature-Pattern in einem konkreten Fall geographisch oder unbekannt oder nicht-geographisch zurückgeben kann, hängt davon ab, ob das in diesem Fall vorliegende Feature als Indikator für geo oder nicht-geo betrachtet werden kann. Beispielsweise ist „in“ als Wort vor dem Kandidaten ein Indikator für eine geographische Entität.

Welche konkreten Feature der Feature-Pattern Indikatoren sind, kann durch einen Unabhängigkeitstest bestimmt werden. Ist ein Feature nicht unabhängig, so stellt es einen Indikator dar. Chi-Square ist ein sehr einfacher Unabhängigkeitstest. Er wurde deshalb zur Bestimmung der Indikatoren ausgewählt und wird im Abschnitt 8.1 näher vorgestellt.

8.1 Chi-Square

Chi-Square oder auch χ^2 ist eine populäre Methode zur Feature-Selektion [MRS07] S. 193 + 194. Im Statistikbereich wird der χ^2-Test verwendet, um die Unabhängigkeit zweier zufälliger Variablen zu überprüfen. Bei der Feature-Selektion sind unter diesen zwei Variablen das Feature (W -word) und die Klasse (C - class) zu verstehen. Auf diese Weise lassen sich dann Feature unter der Beachtung des in Formel 8.1 gegebenen Maßes gewichten.

$$X^2(W,C) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(O_{e_w e_c} - E_{e_w e_c})^2}{E_{e_w e_c}} \tag{8.1}$$

$O_{e_w e_c}$ stellt die beobachtete relative Frequenz der gemeinsamen Vorkommen von w und c dar. In dem im Rahmen dieser Arbeit betrachteten Fall also die Anzahl der Kandidaten, die geographische Entitäten (c) darstellen und das Feature in (w) beinhalten. $E_{e_w e_c}$ beschreibt die erwartete Frequenz unter der Annahme, dass das Feature und die Klasse unabhängig sind.

Es kann gezeigt werden, dass, wenn W und C unabhängig sind, $X^2 \sim \chi^2$ ist, wobei χ^2 die χ^2-Verteilung darstellt [MRS07] S. 193 + 194.

Ein X^2-Wert, der unerwartet hoch ist, sagt aus, dass die Hypothese der Unabhängigkeit sehr wahrscheinlich falsch ist. Unter Verwendung der Rangordnung, einem heuristischen Schwellwert, kann man somit entscheiden, ob ein Feature unabhängig ist oder nicht. Wenn zwei Events abhängig sind, ergibt sich aus dem Vorkommen eines der beiden eine größere (bzw. kleinere) Wahrscheinlichkeit des Eintreffens des anderen. Aus diesem Grund kann X^2 zur Feature-Selektion herangezogen werden.

Eine einfacher berechenbare, jedoch zu Formel 8.1 äquivalente Berechnungsmöglichkeit der Signifikanz ist in Gleichung 8.2 gegeben.

$$X^2(W,C) = \frac{(O_{11} + O_{10} + O_{01} + O_{00})(O_{11}O_{00} - O_{10}O_{01})^2}{(O_{11} + O_{01})(O_{11} + O_{10})(O_{10} + O_{00})(O_{01} + O_{00})} \tag{8.2}$$

Die O_{xy} Werte ergeben sich dabei, bezogen auf die Verwendung in dieser Arbeit für jedes Feature, durch das Ausfüllen der in Tabelle 8.1 dargestellten Chi-Square-Tafel.

	Feature kommt vor	Feature kommt nicht vor
Klasse = geo	O_{11}	O_{01}
Klasse = nicht geo	O_{10}	O_{00}

Tabelle 8.1: Chi-Square-Tafel

8.2 Algorithmus

Wie schon erläutert soll bei diesem Self-Training-Verfahren (Mehrheitsverfahren) die Trainingsmenge ebenfalls iterativ vergrößert werden. Dabei ergibt sich im Gegensatz zu dem auf SVM basierenden Self-Training-Verfahren ein „natürliches“ Abbruchkriterium, wie dem Pseudocode entnommen werden kann.

Zunächst muss in jeder Lernrunde berechnet werden, welche Feature einen Indikator für geo bzw. für nicht-geo darstellen. Dies muss für jedes Feature-Pattern einzeln erfolgen. Dabei ist wichtig, dass in jeder Lernrunde alle Indikatoren neu bestimmt werden, weshalb zu Beginn einer Lernrunde zunächst alle Feature-Informationen gelöscht werden, die eventuell aus vorherigen Lernrunden noch vorhanden sind (vergleiche Pseudocode Zeilen 3-5). Zu beachten ist, dass dieselben Feature-Pattern wie bei den auf SVM basierenden Verfahren

verwendet werden. Außerdem wurde bei diesem Verfahren die case-insensitive Verwendung der Gazetteer genauso wie bei den bisher vorgestellten Verfahren implementiert.

Um mit Hilfe von Chi-Square bestimmen zu können, welches Feature ein Indikator darstellt, muss quasi die in Abschnitt 8.1 vorgestellte Chi-Square-Tafel ausgefüllt werden. Dies muss für jedes Feature von jedem Feature-Pattern geschehen. Deshalb wird, wie im Pseudocode in den Zeilen 7 bis 11 dargestellt, über alle Trainingsbeispiele und Feature-Pattern iteriert und die Funktion „inkrementiere Feature“ aufgerufen.

```
( 1)  solange bis keine neuen Bsp. hinzugenommen werden {
( 2)
( 3)    mit allen Feature-Pattern {
( 4)        alle Feature löschen
( 5)    }
( 6)
( 7)    mit allen Trainingsbeispielen {
( 8)        mit allen Feature-Pattern {
( 9)            inkrementiere Feature
(10)        }
(11)    }
(12)
(13)    mit allen Feature-Pattern {
(14)        mit allen Feature {
(15)            berechne Chi-Square Wert
(16)            bestimme geo/non-geo
(17)        }
(18)    }
(19)
(20)    (Analyse anhand Analysemenge)
(21)
(22)    mit allen Beispielen (die nicht annotiert und
(23)        bisher nicht in der Trainingsmenge sind){
(24)
(25)        selectiere neue gute Beispiele
(26)        füge diese zur Trainingsmenge hinzu
(27)    }
(28)
(29)  }
```

Beim Aufruf von „inkrementiere Feature“ (Zeile 9) wird gezählt, wie oft ein Feature in einem geographischen bzw. nicht-geographischen Beispiel vorkommt. Anhand der Gesamtanzahl der Beispiele lassen sich dann alle vier Felder der Chi-Square-Tafel (siehe Tabelle 8.1) für die Feature ausfüllen. Dadurch ist es, wie schon erläutert, mit der Formel 8.2 möglich, die Signifikanz eines Feature zu berechnen. Dies wird für jedes Feature von

jedem Feature-Pattern, wie in den Zeilen 13 bis 18 dargestellt, durchgeführt. Ob ein Feature ein Indikator für geographisch oder für nicht-geographisch ist, lässt sich durch einen Vergleich der Anzahl der geographischen und der nicht-geographischen Vorkommen des Feature bestimmen, was in der Pseudocode-Zeile 16 angedeutet wurde.

In Zeile 20 im Pseudocode liegt nun sozusagen der Klassifikator vor und es können neue Beispiele klassifiziert werden. Falls erwünscht, kann also an dieser Stelle eine Analyse oder Evaluierung auf einer entsprechenden Menge durchgeführt werden.

In den Zeilen 22 bis 27 wird über alle Beispiele iteriert, die nicht in der Lernmenge oder einer der annotierten Mengen enthalten sind. In Zeile 25 wird bestimmt, wie viele Feature-Pattern für ein Beispiel ein Feature liefern, das ein Indikator für geographisch bzw. für nicht-geographisch ist. Als relevant werden dabei Feature mit einem Chi-Square-Wert größer 3,84 betrachtet.

Bei der Klassifikation für die Selektion von Beispielen ist es nicht so wichtig, möglichst viele Beispiele zu klassifizieren, sondern möglichst viele korrekt klassifizierte Beispiele zu erhalten. Deshalb wird für die Selektion nicht ein einfacher Mehrheitsentscheid verwendet, sondern eine „strengere Variante“. Zur Trainingsmenge wird ein Beispiel als geographisch hinzugefügt, wenn die Anzahl der Feature, die als geo-Indikator gelten, größer als die Hälfte der insgesamt verwendeten Feature-Pattern +1 ist und weniger als ein Fünftel der Feature ein Indikator für nicht-geographisch sind. Zum Hinzufügen von nicht-geographischen Beispielen wurde das Selektionskriterium von geographischen Beispielen symmetrisch angewendet.

Als endgültiger Klassifikator kann selbstverständlich sowohl das strenge zu Selektion herangezogene Kriterium als auch ein weniger strenges Mehrheitskriterium herangezogen werden. Hierdurch kann erreicht werden, dass nach Möglichkeit alle zu klassifizierenden bzw. zu disambiguiernden Beispiele klassifiziert werden können, wenn auch nicht mit so hoher Sicherheit.

8.3 Analyse

Da die Feature-Pattern-Kombination b2, b1, suff, case, a1 und a2 sich, wie beschrieben, bei den SVM basierten Verfahren bereits als die beste Kombination erwiesen hatte, wurde neben einigen anderen insbesondere diese für das Self-Training mit Mehrheitsverfahren verwendet.

Leider ergaben sich auch bei diesem Self-Training-Verfahren keine nennenswerten Verbesserungen durch das iterative Vergrößern der Lernmenge. Wie in Tabelle 8.2 zu sehen ist, war eine leichte Verbesserung in der zweiten Lernrunde zu verzeichnen. Ansonsten fielen die Werte ab. Die verwendete Analysemenge ist dieselbe wie bei den SVM-Verfahren (siehe Abschnitt 6.4). Auffällig ist, dass insbesondere die fälschlicherweise als geographisch klassifizierten Fälle stark zunehmen. Bei den Werten in Tabelle 8.2 ist zu beachten, dass zur Klassifikation eine einfache Mehrheit der als Indikator zu betrachtenden Feature genügt.

Zur Selektion wurde, wie bereits in Abschnitt 8.2 geschildert, ein strengerer Klassifikator herangezogen. In Tabelle 8.3 ist gezeigt, wie gut der Klassifikator zum Hinzunehmen

von Beispielen ist, und wie viele Beispiele pro Runde hinzugenommen wurden. Dabei ist zu beobachten, dass bei diesem Klassifikator die Acc-Werte stetig steigen (mit Ausnahme der dritten Lernrunde). Bei den F_1-Werten ergaben sich allerdings keine Verbesserungen. Dieser Unterschied ist darauf zurückzuführen, dass die richtig klassifizierten nicht-geo Beispiele zunehmen, während die richtig klassifizierten geo Beispiele stark abnehmen. Wie durch die Rec-Werte deutlich wird, ergab sich auch eine sehr große Menge von Beispielen, die überhaupt nicht klassifiziert werden konnten (vergl. Tab. 8.3).

Weitere Experimente wurden mit anderen Feature-Pattern-Kombinationen und mit kleineren Boot-Mengen durchgeführt. Leider konnten auch bei diesen keine besseren Ergebnisse erzielt werden. Daher wurde von tiefer gehenden Untersuchungen von Varianten abgesehen, zumal der „überwachte“ F_1-Wert des Verfahrens (Klassifikation in der ersten Runde) lediglich bei 89,76% anstelle von 96,01% beim überwachten Lernen mit SVM liegt.

Runde	korrekt			falsch			ohne	gleich	ges	Acc	falsch	ohne	gleich	Pre	Rec	F_1
		geo	nicht-geo		geo	nicht-geo				(%)	(%)	(%)	(%)	(%)	(%)	(%)
1	443	333	110	66	2	64	2	49	560	79,11	11,79	0,36	8,75	83,88	96,52	89,76
2	455	322	133	48	5	43	2	55	560	81,25	8,57	0,36	9,82	88,22	93,33	90,70
3	444	326	118	56	2	54	0	60	560	79,29	10,00	0,0	10,71	85,79	94,50	89,93
4	430	294	136	53	13	40	0	77	560	76,79	9,46	0,0	13,75	88,88	85,23	87,02
5	427	294	133	60	12	48	0	73	560	76,25	10,71	0,0	13,04	85,96	85,22	85,59
6	427	286	141	68	21	47	0	65	560	76,25	12,14	0,0	11,61	85,89	82,90	84,37
7	428	287	141	70	21	49	0	62	560	76,43	12,50	0,0	11,07	85,42	83,19	84,29
8	428	287	141	70	21	49	0	62	560	76,43	12,50	0,0	11,07	85,42	83,19	84,29
9	428	287	141	70	21	49	0	62	560	76,43	12,50	0,0	11,07	85,42	83,19	84,29
10	427	287	140	70	21	49	0	63	560	76,25	12,50	0,0	11,25	85,42	83,19	84,29
11	428	288	140	70	21	49	0	62	560	76,43	12,50	0,0	11,07	85,46	83,48	84,46
12	428	288	140	71	22	49	0	61	560	76,43	12,68	0,0	10,89	85,46	83,48	84,46
13	428	288	140	71	22	49	0	61	560	76,43	12,68	0,0	10,89	85,46	83,48	84,46
14	428	288	140	71	22	49	0	61	560	76,43	12,68	0,0	10,89	85,46	83,48	84,46
15	428	288	140	71	22	49	0	61	560	76,43	12,68	0,0	10,89	85,46	83,48	84,46

Tabelle 8.2: Klassifikationsergebnisse – Self-Training mit Mehrheitsverfahren; mit einfacher Mehrheitsbildung

Runde	Acc (%)	falsch (%)	ohne (%)	Pre (%)	Rec (%)	F_1 (%)	nicht-geo	geo	ges	ohne	möglich
1	13,39	0,36	86,25	97,14	19,71	32,77	256	726	982	95910	96892
2	13,75	0,54	85,71	95,77	19,71	32,69	1211	149	1360	94550	95910
3	11,79	0,00	88,21	100,00	10,43	10,43	865	1110	1975	92575	94550
4	14,29	0,00	85,71	100,00	11,59	20,77	3315	394	3709	88866	92575
5	15,18	0,00	84,82	100,00	8,99	16,50	1472	975	2447	86419	88866
6	15,71	0,00	84,29	100,00	8,12	15,02	1329	245	1574	84845	86419
7	16,79	0,00	83,21	100,00	8,70	16,01	440	86	526	84319	84845
8	16,96	0,00	83,04	100,00	8,99	16,50	72	23	95	84224	84319
9	16,96	0,00	83,04	100,00	8,95	16,43	22	2	24	84200	84224
10	17,14	0,00	82,86	100,00	9,28	16,98	26	65	91	84109	84200
11	17,14	0,00	82,86	100,00	9,28	16,98	0	15	15	84094	84109
12	17,14	0,00	82,86	100,00	9,28	16,98	19	8	27	84067	84094
13	17,14	0,00	82,86	100,00	9,28	16,98	22	0	22	84045	84067
14	17,14	0,00	82,86	100,00	9,28	16,98	13	0	13	84032	84045
15	17,14	0,00	82,86	100,00	9,28	16,98	0	0	0	84032	84032

Tabelle 8.3: Klassifikationsergebnisse – Self-Training mit Mehrheitsverfahren; wie in Tabelle 8.2, jedoch mit strenger Mehrheitenbildung. Zusätzlich sind Daten zur Selektion von neu klassifizierten Beispielen angegeben; „nicht-geo“, „geo“ und „ges“ gibt die Anzahl der selektierten Beispiele an; „ohne“ gibt die Anzahl der Bsp. an, die ohne bekannte Feature sind und „möglich“ bezeichnet die zur Selektion vorhandenen Beispiele.

Kapitel 9

Bootstrap-Verfahren

Nachdem in den letzten zwei Kapiteln Self-Training-Verfahren vorgestellt und untersucht wurden, soll in diesem Kapitel auf ein Bootstrap-Verfahren eingegangen werden. Neben dem iterativen Vergrößern einer handannotierten Trainingsmenge gibt es die Möglichkeit, eine Trainingsmenge anhand von Heuristiken komplett maschinell zu generieren.

Im nächsten Abschnitt wird daher zunächst diskutiert, welche Möglichkeiten es zur Berechnung einer Trainingsmenge gibt. Darauf folgend wird im Abschnitt 9.2 ein Verfahren vorgestellt, das auf den in Abschnitt 9.1 diskutierten Ansätzen beruht. Im Abschnitt 9.3 werden einige Untersuchungen zu diesem Verfahren vorgestellt.

9.1 Trainingsmenge maschinell erzeugen

Im Rahmen dieser Arbeit ist versucht worden, eine möglichst gute Bootstrap-Menge zu bilden, ohne von Hand Kandidaten zu annotieren. Hierfür mussten Beispiele für geographische sowie für nicht-geographische Vorkommen gebildet werden, die möglichst repräsentativ sind.

Alfred Hitchcocks	Theo Waigel
Nutzen	Jahreslauf
Jahr	Professor
Otto	Dienstag
beschlossen	mutigen
Schäden	Dollar
Juli	WDR-Hauszeitschrift
Eiche	Grün
Baum	

Tabelle 9.1: Liste nicht-geographischer Worte für die maschinelle Erzeugung einer Lernmenge

Ein erster einfacher Ansatz war, eine kurze Liste von Worten zusammenzustellen (siehe Tabelle 9.1), die ziemlich sicher keine Orte sind. Diese wurden in 9000 zufällig bestimmten Artikeln gesucht und als Beispiele für nicht-geographische Vorkommen gespeichert.

Ebenso wurde zur Erstellung von geographischen Beispielen vorgegangen. In Tabelle 9.2 sind die verwendeten Ortsnamen aufgelistet, die sich bei Untersuchungen als geographisch und nicht mehrdeutig erwiesen. Diese wurden ebenfalls in den beschriebenen Artikeln des STZ gesucht und als geographische Trainingsbeispiele abgespeichert.

Stuttgart	STUTTGART
Reichenbach/Fils	Rems-Murr-Kreis
München	FRANKFURT/MAIN
Schleswig-Holstein	Bad Rothenfelde
Bad Urach	

Tabelle 9.2: Liste geographischer Worte für die maschinelle Erzeugung einer Lernmenge

Mit Hilfe des überwachten Ansatzes mit SVM (siehe Abschnitt 6.1) wurden im Vergleich zu handannotierten Trainingsbeispielen untersucht, wie gut oder schlecht die Ergebnisse ausfallen, wenn die beschriebene Trainingsmenge verwendet wird.

Lernmenge	Lernbsp.	korrekt	falsch	Acc (%)	Pre (%)	Rec (%)	F_1 (%)
annotiert	575	532	28	95,00	94,40	97,68	96,01
Listen generiert	10267	240	320	42,86	55,06	39,42	45,95

Tabelle 9.3: Vergleich der Ergebnisse überwachter SVM-Klassifikationen mit unterschiedlichen Lernmengen – einer handannotierten Lernmenge und einer mit Hilfe von Listen maschinell generierten Lernmenge

In Tabelle 9.3 sind noch mal die Ergebnisse aufgeführt, die sich mit der handannotierten Lernmenge ergeben. Darunter ist zum Vergleich das Ergebnis der mit den Wortlisten generierten Lernmenge dargestellt. Wie ersichtlich ergibt sich bei der mittels Wortlisten generierten Boot-Menge mit einem F_1-Wert von 45,95% ein erheblich schlechteres Ergebnis im Vergleich zu 96,01% (F_1) bei der handannotierten Lernmenge. Deshalb wurde ein weiterer Ansatz betrachtet.

Selektiert man mit Hilfe der Gazetteer alle Kandidaten und deren Kontext, so erhält man eine Beispielmenge, in der ca. 60% (vergl. Abschnitt 3.6) der Beispiele geographisch sein müssten. Nachdem Beispiele mit kleingeschriebenen Kandidaten aussortiert wurden, ist sogar mit 75% (vergl. Abschnitt 3.6) geographischen Beispielen zu rechnen. Betrachtet man diese Beispielmenge nun komplett als geographische Beispiele, so erhält man relativ einfach eine recht gute Annäherung.

Schwieriger ist es jedoch mit den nicht-geographischen Beispielen. Um nicht-geo Beispiele aufzufinden, wurde zunächst in Betracht gezogen, jegliche Worte als Kandidaten zu verwenden, die nicht in den Gazetteer vorhanden sind. Dies wirft jedoch das Problem auf, dass ein Nichtvorhandensein nicht heißen muss, dass es sich dann nicht um eine geographische Entität handelt, da Ortsnamen anderer Länder nicht in den Gazetteer stehen. Dies ließe sich jedoch durch eine Gegenprobe zum gesamten Gazetteer [*WG] beheben. Eine weitere Verbesserung ergibt sich auch für die nicht-geograpischen Trainingsbeispiele wiederum durch ausschließliches Verwenden von großgeschriebenen Kandidatenwörtern.

Ungelöst bleibt jedoch trotzdem das Problem, dass Bezeichnungen wie beispielsweise „Ulmer Münster“ wegen der Endung „er“ nicht vom Gazetteer gefunden werden, was bedeutet, dass sie als nicht-geographische Beispiele beim Erstellen der Lernmenge betrachtet werden, obwohl sie geographische Beispiele sind.

Eine weitere Verbesserungsmöglichkeit der geographischen Lernbeispiele könnte in der Verwendung einer Selektionsgrenze liegen. Hierzu könnten zum Beispiel Orte erst ab einer bestimmte Größe bzw. einer bestimmten Einwohnerzahl zur Beispielselektion herangezogen werden. Kleinere Orte, die vermutlich öfter in einem nicht-geographischen Sinn vorkommen, würden dann einfach ignoriert und nicht als geographisches Beispiel aufgenommen, ähnlich wie dies in [AHSS04] schon verwendet wurde (siehe Kapitel 4).

Auf der Basis der in diesem Abschnitt vorgestellten Überlegungen wurde im Rahmen dieser Arbeit ein Bootstrap-Verfahren implementiert. Dieses Verfahren wird in Abschnitt 9.2 vorgestellt. Im Abschnitt 9.3 sind einige experimentelle Untersuchungen zu diesem Verfahren zu finden.

9.2 Algorithmus

Wie im letzten Abschnitt diskutiert soll bei diesem Verfahren eine Lernmenge maschinell erzeugt werden. Hierfür soll, wie im letzten Abschnitt bereits erläutert, die Menge aller mit einem Großbuchstaben beginnenden Worte im STZ betrachtet werden. Aus dieser Menge sollen, wie in Abbildung 9.1 dargestellt, drei Teilmengen gebildet werden. Die Teilmenge 1 enthält alle Worte, die nicht im weltweiten Gazetteer stehen. Teilmenge 2 enthält alle Worte, die im Gazetteer stehen und im Gazetteer einen Eintrag mit einer Einwohnerzahl größer X haben. Die restlichen Worte gehören der Teilmenge 3 an und werden im Rahmen dieses Verfahrens ignoriert.

Um, wie beschrieben, die Teilmengen bilden zu können, muss der Gazetteer in den Speicher geladen werden. Da der weltweite Gazetteer relativ groß ist, wurde entschieden, lediglich die Namen und anderen Schreibweisen sowie eine Markierung, die Einträge kennzeichnet, die eine Einwohnerzahl größer X haben, aufzunehmen. Die anderen Daten wie beispielsweise Längengrad und Breitengrad werden nicht in den Speicher geladen.

Ist der Gazetteer geladen, können die Texte des Korpus sequentiell durchlaufen werden. Trifft man auf ein großgeschriebenes Wort, so wird überprüft, ob dieses (unter Umständen mit den Folgewörtern zusammen) im Gazetteer steht. Ist dies der Fall, liegt ein geographisches Trainingsbeispiel vor, sofern die Größenmarkierung im Gazetteer gesetzt ist. Ist diese nicht gesetzt, wird das Beispiel ignoriert, wie oben schon anhand Abbildung 9.1 erläutert. Steht das Wort nicht im Gazetteer, wird es als nicht-geographisches Beispiel in die Trainingsmenge aufgenommen.

Bei diesem Durchlauf durch das STZ werden die Beispiele zeilenweise als Textausschnitt des Originaltextes in eine Datei geschrieben. Erst im nächsten Schritt wird die Datei mit den Trainingsbeispielen in die für SVM benötigte Repräsentation überführt. Außerdem werden die Analysedaten und das Wörterbuch für SVM generiert. Durch das Überführen der Trainingsbeispiele in zwei Schritten ist es möglich, den Gazetteer aus dem Speicher

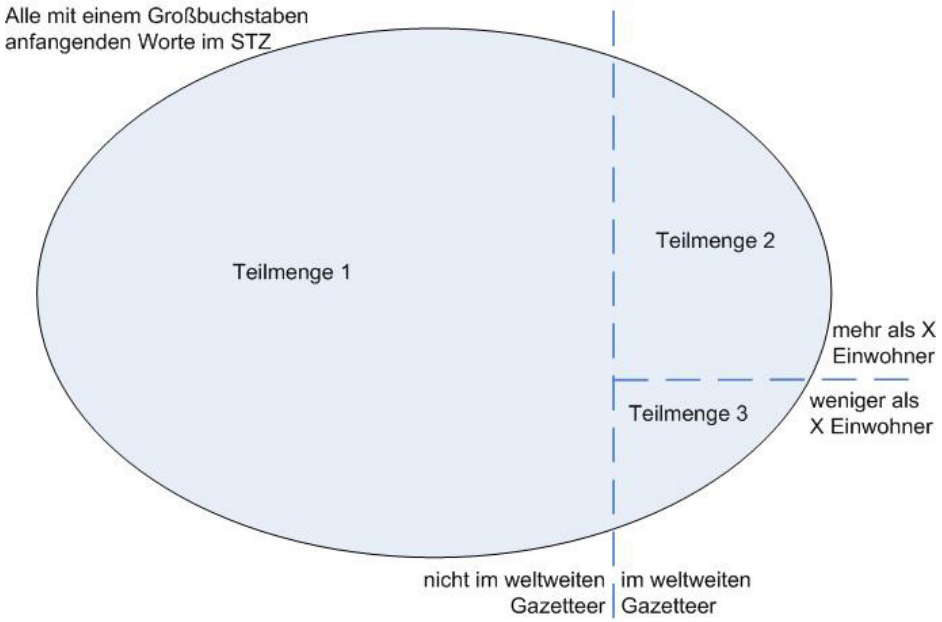

Abbildung 9.1: Mengenbildung zum maschinellen Erzeugen der Trainingsmenge

zu nehmen, bevor das Wörterbuch während dem Erzeugen der SVM-Repräsentation im Speicher aufgebaut wird.

Wie schon in Abbildung 9.1 angedeutet, liegt eine erheblich größere Zahl von nicht-geographischen Beispielen als von geographischen Beispielen vor. Daher ist zu vermuten, dass sehr viele der nicht-geographischen Beispiele einen Kontext haben, der niemals bei geographischen bzw. mehrdeutigen Entitäten vorkommen kann. Auch könnte ein starkes Ungleichgewicht der geographischen / nicht-geographischen Beispiele bei der Klassifikation mit SVM zu ungenaueren Ergebnissen führen.

In Abbildung 9.2 ist die gesamte Menge der Trainingsbeispiele dargestellt. Erlernt man nun mit SVM ein Modell unter Verwendung dieser Beispiele und klassifiziert sie mit diesem erlernten Modell gleich wieder, erhält man von SVM Klassifikationswerte für die Trainingsbeispiele.

Hierbei ist zu hoffen, dass die Beispiele, die nie in einem geographischen Kontext vorkommen können, einen hohen negativen SVM-Wert erhalten und somit, wie in Abbildung 9.2 angedeutet, in einem separaten Bereich der gesamten Beispielmenge liegen. In Abbildung 9.2 also rechts von der Strichpunktlinie. Angesichts dieser Überlegungen wäre ein Aussortieren der Teilmenge 3 (Abb. 9.2) sicher sinnvoll.

In Ermangelung eines bekannten Grenzwertes für die in Abbildung 9.2 angedeutete Grenze wurde entschieden, eine Grenze so zu berechnen, dass sich ungefähr gleich viele geographische sowie nicht-geographische Beispiele ergeben. Da dies nicht unbedingt die benötigte Grenze ist, wird es notwendig, neue Beispiele in zwei Stufen zu klassifizieren.

Dementsprechend wird im implementierten Bootstrap-Verfahren zunächst wie beschrieben die Menge der Lernbeispiele gebildet. Mit dieser Menge – Teilmenge 1 als geographische

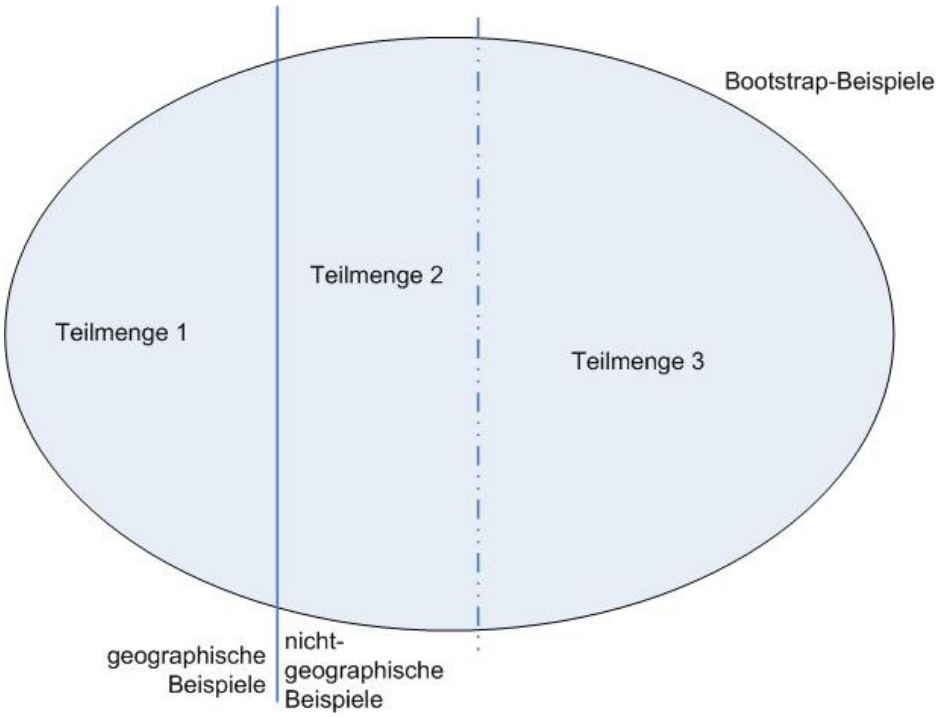

Abbildung 9.2: Erhoffte Aufteilung der Trainingsmenge durch SVM-Gewichte

und Teilmenge 2 und 3 als nicht-geographische Beispiele (vergl. Abb 9.2) – wird ein Modell (Modell 1) erlernt. Mit dem Modell 1 werden die Beispiele in der Lernmenge klassifiziert und mit den SVM-Werten eine Grenze berechnet, so dass ungefähr gleich viele geographische und nicht-geographiche Beispiele vorliegen. Mit diesen Beispielen – Teilmenge 1 als geographische und Teilmenge 2 als nicht-geographische Beispiele (vergl. Abb. 9.2) – wird dann ein weiteres Modell (Modell 2) berechnet.

Beim Klassifizieren von einem neuen Beispiel muss dieses zunächst mit SVM unter Anwendung des Modell 1 klassifiziert werden. Ist der SVM-Klassifikationswert größer als der Grenzwert, wird es mit dem zweiten Modell noch mal klassifiziert. Ansonsten ist es bereits als ein nicht-geographisches bzw. mehrdeutiges Vorkommen zu betrachten.

9.3 Analyse

Zu dem im Abschnitt 9.2 vorgestellten Verfahren wurden verschiedene Untersuchungen durchgeführt. In den Tabellen 9.5, 9.6 und 9.7 sind die Klassifikationsergebnisse bezogen auf die Analysemenge (siehe Abschnitt 6.4) dargestellt. Im jeweils ersten Tabellenteil wurden die Feature-Pattern variiert. Unter der Bezeichnung „base“ verbergen sich die Feature-Pattern b2, b1, suff, case, a1 und a2, wie sie schon beim überwachten Verfahren und den Self-Training-Verfahren verwendet wurden.

Bei den anderen bereits beschriebenen Verfahren ergaben sich bei dieser Feature-Pattern-Kombination die besten Ergebnisse. Beim hier untersuchten Bootstrap-Verfahren

musste jedoch festgestellt werden, dass sich bei der Feature-Pattern-Kombination „more“ bessere Ergebnisse abzeichnen. Die Kombination „more“ enthält zusätzlich zu den Feature-Pattern in „base“ das Wort-Feature-Pattern b3 und die Wortkombinations-Feature-Pattern b3,b1 und b2,b1,a1. In „more2“ sind zusätzlich zu den Feature-Pattern in „more“ das Wort-Feature-Pattern b4 sowie die Wortkombinations-Feature-Pattern b1,a1; b2,Kandidat und Kandidat,a2 enthalten.

Neben der Variation der Feature-Pattern-Kombination wurde untersucht, welche Auswirkungen die Verwendung der Einwohnerzahl als Grenze beim Selektieren der geographischen Beispiele hat. Im jeweils zweiten Teil der Tabellen 9.5, 9.6 und 9.7 sind drei der durchgeführten Analyseläufe dargestellt. Zu beachten ist, dass die Einträge des Nav (vergl. Abschnitt 3.4) alle als groß genug markiert wurden, die des gesamten Gazetteers [*WG] jedoch nur als groß genug markiert wurden, wenn sie eine angegebene Einwohnerzahl mit der Mindestgröße hatten. Obwohl bei einer kleineren Einwohnerzahl sich natürlich eine größere Lernmenge ergibt, waren die Ergebnisse bei einer Grenze von mindesten 5000 Einwohnern am besten.

Neben den Auswirkungen der Einwohnerzahl und verschiedenen Feature-Pattern wurde außerdem untersucht, ob ein Verändern der Gewichte der Feature-Pattern eine positive Auswirkung hat. So wurden beispielsweise für die Suffixe ein geringeres Gewicht gesetzt. Auch für die Wort-Feature-Pattern wurden geringere Gewichte in Abhängigkeit vom Abstand zum Kandidaten untersucht ($1/\mathit{Abstand}$). Diese Veränderungen ergaben jedoch keine Verbesserungen.

Bei den bisher vorgestellten Untersuchungen wurden lediglich 100 Artikel verwendet, um die Trainingsmenge zu bilden. Wie bereits diskutiert werden nur Feature-Pattern verwendet, die auf Text basieren. Deshalb ist anzunehmen, dass sich bei steigender Artikelzahl eine weitere Verbesserung ergeben müsste. Da die Feature-Pattern-Kombination „more“ in Verbindung mit einer Einwohnerzahl größer 5000 die besten Ergebnisse ergab, sollte für diese Parameter ein Analyselauf mit allen extrahierten Artikeln als Quelle für Trainingsbeispiele durchgeführt werden.

SVM weist einen quadratischen Aufwand in Abhängigkeit von den verwendeten Beispielen auf. Deshalb wurde für den Lauf mit allen extrahierten Artikeln das Bootstrap-Verfahren auf einem UNIX-Server ausgeführt. Hierbei musste festgestellt werden, dass bei der Ausführung von SVM auf einem UNIX-System im Vergleich zu einem Windows-System andere precision-Werte berechnet werden. Das liegt daran, dass ein UNIX-System eine andere interne Zahlenrepräsentation verwendet. Bei einfacher Ausführung von SVM ergeben sich hierdurch keine Klassifikationsunterschiede. Durch das Berechnen der Selektionsgrenze für die zweite Klassifikationsstufe im verwendeten Bootstrap-Verfahren ergeben sich durch die unterschiedlichen precision-Werte jedoch unterschiedliche Trainingsmengen für die zweite Stufe und somit auch unterschiedliche Klassifikationsergebnisse.

Um den Unterschied zu verdeutlichen, wurde für 100 Artikel, der Feature-Pattern-Kombination „more“ und einer Grenze von 5000 Einwohnern sowohl auf einem Windows- als auch auf einem UNIX-System eine Klassifikation durchgeführt. In Tabelle 9.4 sind jeweils der berechnete Grenzwert, die Anzahl der korrekt und der falsch klassifizierten Bei-

spiele in der zweiten Klassifikationsstufe sowie der Acc-Wert für die gesamte Klassifikation angegeben.

System	Grenzwert	korrekt	falsch	gesamt Acc (%)
Windows	-0.99960464	422	99	80,89
UNIX	-0.99963883	401	120	77,14

Tabelle 9.4: Unterschiede bei der Klassifikation auf einem UNIX- und einem Windows-System

Diese großen Unterschiede in den Klassifikationsergebnissen sind darauf zurückzuführen, dass unter Windows einige wenige andere Beispiele zur Trainingsmenge der zweiten Stufe hinzugefügt werden, die für die Klassifikation sehr wichtig sind. Daher ist anzunehmen, dass der extreme Unterschied sich bei einem Lauf mit allen extrahierten Artikeln relativieren würde. Deshalb wurde trotz der erheblichen Unterschiede in den Ergebnissen ein Analyselauf wie beschrieben auf dem UNIX-Server mit allen extrahierten Artikeln durchgeführt.

In den Tabellen 9.5, 9.6 und 9.7 sind die Ergebnisse zu diesem Analyselauf im jeweils letzten Abschnitt dargestellt. Im Vergleich zu 80,89% Acc (77,14% Acc unter UNIX) ergibt sich mit 85,89% Acc und einem F_1-Wert von 89,16% wie erhofft ein verbessertes Ergebnis. Leider ist das Ergebnis weiterhin schlechter, als die 96,01% F_1 des überwachten Verfahrens.

Art.	Feature Pattern	mindest Einwohner	Lernm. 1. Stufe geo	Lernm. 1. Stufe nicht-geo	Analyse 1. Stufe korrekt	Analyse 1. Stufe falsch	Analyse 1. Stufe ges
100	base	5000	654	5637	35	8	43
100	more	5000	654	5637	31	8	39
100	more2	5000	654	5637	26	7	33
100	more	0	1020	5636	11	8	19
100	more	5000	654	5637	31	8	39
100	more	7000	618	5637	28	8	36
15976	more	5000	167268	1451777	121	3	124

Tabelle 9.5: Teil 1 – Klassifikationsergebnisse des Bootstrap-Verfahrens

Lernm. 2.Stufe geo	Lernm. 2.Stufe nicht-geo	Analyse 2. Stufe korrekt	Analyse 2. Stufe falsch	Analyse 2. Stufe falsch geo	Analyse 2. Stufe falsch nicht-geo	Analyse 2. Stufe ges
654	653	407	110	66	44	517
654	653	422	99	56	43	521
654	653	398	129	47	82	527
1020	1019	403	138	68	70	541
654	653	422	99	56	43	521
616	617	419	105	63	42	524
167268	167497	360	76	17	59	436

Tabelle 9.6: Teil 2 – Klassifikationsergebnisse des Bootstrap-Verfahrens

Gesamt Analyse korrekt	falsch	falsch geo	falsch nicht-geo	ges	Acc (%)	Pre (%)	Rec (%)	F_1 (%)
442	118	74	44	560	78,93	86,03	78,55	82,12
453	107	64	43	560	80,89	86,73	81,45	84,01
424	136	54	82	560	75,71	78,02	84,35	81,06
414	146	76	70	560	73,93	79,35	77,97	78,65
453	107	64	43	560	80,89	86,73	81,45	84,01
447	113	71	42	560	79,82	86,71	79,42	82,91
481	9	20	59	560	85,89	84,64	94,20	89,16

Tabelle 9.7: Teil 3 – Klassifikationsergebnisse des Bootstrap-Verfahrens

Kapitel 10

Validierung der Lernverfahren

In den letzten Kapiteln wurde ein überwachtes Lernverfahren, zwei Self-Training-Verfahren und ein Bootstrap-Verfahren vorgestellt. In diesem Kapitel sollen die Varianten dieser Verfahren, die bei der Analyse die besten Ergebnisse lieferten, mit der bisher unbenutzten Evaluierungsmenge (siehe Abschnitt 6.4) validiert werden.

10.1 Überwachtes und Self-Training-Verfahren (SVM)

Wie in den Abschnitten 6.4 und 7.2 beschrieben ergeben sich bei der Feature-Pattern-Kombination b2, b1, suff, case, a1 und a2 die besten Ergebnisse. Daher wurde diese Kombination zur Evaluierung ebenfalls verwendet.

case	Acc (%)	Pre (%)	Rec (%)	F_1 (%)	korrekt	falsch	ges	falsch geo	falsch nicht-geo
insensitiv	92,16	90,63	95,98	93,23	611	52	663	15	37
sensitiv	90,00	90,63	95,98	93,23	468	52	520	15	37

Tabelle 10.1: Ergebnisse der Klassifikation mit der Evaluierungsmenge

Wie in Tabelle 10.1 ersichtlich ergibt sich beim überwachten Lernverfahren mit 92,16% korrekt klassifizierten Kandidaten wieder ein recht gutes Ergebnis. Da die Baseline für case-insensitives Vorgehen bei 60% liegt, ist eine Verbesserung von 32,16% zu verzeichnen. Allerdings muss angemerkt werden, dass 143 kleingeschriebene, nicht-geographische Beispiele in der Evaluierungsmenge enthalten sind. Diese werden in diesem Fall alle korrekt klassifiziert. Das heißt, bei case-sensitiver Betrachtung ergibt sich mit 90 % Acc lediglich eine Verbesserung von 15% (Baseline für case-sensitiv ist 75% Acc bzw. 75% F_1). Bezogen auf den F_1-Wert von 93,32% wird mit dem überwachten Verfahren also eine Verbesserung von 18,32 % erreicht.

Zur Validierung des Self-Training-Verfahrens werden pro Lernrunde die besten fünf bestimmt. Diese und gleiche in den jeweiligen Artikeln werden zur Lernmenge hinzugenommen. Wie schon bei der Analyse wird mit 60 Runden gelernt und 5000 Artikel werden nach neuen Beispielen durchsucht.

In allen Lernrunden ergab sich keine Veränderung der Klassifizierung. Somit blieben die in Tab 10.1 aufgeführten Ergebnisse konstant. Auch ergab sich keine Änderung der falsch klassifizierten Beispiele, obwohl die Trainingsmenge von anfänglich 564 auf 1164 Beispiele vergrößert wurde. Das heißt, insgesamt wurden 864 Beispiele, die mehrfach vorkommen selektiert. Bis auf wenige Ausnahmen werden allerdings nur Beispiele zu Stuttgart und zu Baden-Württemberg zur Trainingsmenge hinzugefügt. Hierbei sanken die SVM-Werte der Besten stetig von anfangs 1,99 auf 1,73.

Das auf SVM basierende Self-Training-Verfahren erwies sich somit bei der Evaluierung wie schon bei der Analyse als nicht brauchbar.

10.2 Self-Training-Verfahren (Mehrheitsverfahren)

Zur Evaluierung des Self-Training-Ansatzes mit Mehrheitsverfahren wurden dieselben Parameter wie beim in Abschnitt 8.3 vorgestellten Analyselauf verwendet.

In Tabelle 10.2 sind die Klassifikationsergebnisse bei einfacher Mehrheitsbildung bezüglich der Evaluierungsmenge dargestellt. Wie schon beim überwachten Verfahren ergibt sich auch hier eine Verschlechterung der Ergebnisse bezogen auf die Evaluierungsmenge im Vergleich zur Analysemenge. Außerdem ist wie schon beim Analyselauf zu beobachten, dass sich in der zweiten Lernrunde mit einem F_1-Wert von 87,32% eine minimale Verbesserung im Vergleich zu 86,40% in der ersten Lernrunde abzeichnet. In weiteren Lernrunden wird das Gesamtergebnis jedoch schlechter (vergl. Tab. 10.2).

Interessant ist, dass die fälschlicherweise als geographisch klassifizierten Kandidaten im Verlauf des Lernprozesses fast halbiert werden. Diese Verbesserung wird jedoch durch eine Zunahme von fälschlicherweise als nicht-geo klassifizierten Beispielen und von Kandidaten mit gleich vielen Feature für geo und für nicht-geo zunichte gemacht.

Wie schon bei der Analyse zeigt sich auf der Evaluierungsmenge wieder, dass der strenge Klassifikator die Acc-Werte betreffend insgesamt zu einer Verbesserungen führt, wie in Tabelle 10.3 dargestellt. Betrachtet man jedoch die F_1-Werte, wird eine Verschlechterung sichtbar. Außerdem ergeben sich wieder sehr viele Beispiele, die überhaupt nicht klassifiziert werden können, wie anhand der Rec-Werte in Tabelle 10.3 zu sehen ist. Auch ist die Fehlerrate des strengen Klassifikators auf der Evaluierungsmenge im Vergleich zur Analysemenge etwas höher.

Insgesamt ist das mehrheitsbasierte Self-Training-Verfahren mit einem maximalen F_1-Wert von 87,32% und einem (überwachten) F_1-Wert von 86,40% erheblich schlechter als das überwachte Verfahren mit einem F_1-Wert von 93,23%.

Runde	korrekt			falsch			ohne	gleich	ges	Acc	falsch	ohne	gleich	Pre	Rec	F_1
		geo	nicht-geo		geo	nicht-geo				(%)	(%)	(%)	(%)	(%)	(%)	(%)
1	498	359	139	100	1	99	1	64	663	75,11	15,08	0,15	9,65	78,38	96,25	86,40
2	501	341	160	74	7	67	0	88	663	75,57	11,16	0,00	13,27	83,58	91,42	87,32
3	500	350	150	86	3	83	0	77	663	75,41	12,97	0,00	11,61	80,83	93,83	86,85
4	494	326	168	67	11	56	0	102	663	74,51	10,11	0,00	15,39	85,34	87,40	86,36
5	496	327	169	72	12	60	0	95	663	74,81	10,86	0,00	14,33	84,50	87,67	86,06
6	486	311	175	69	15	54	0	108	663	73,30	10,41	0,00	16,29	85,21	83,38	84,29
7	493	316	177	70	14	56	0	100	663	74,36	10,56	0,00	15,08	84,95	84,72	84,83
8	491	315	176	69	14	55	0	103	663	74,06	10,41	0,00	15,54	85,14	84,45	84,79
9	491	315	176	68	14	54	0	104	663	74,06	10,26	0,00	15,69	85,37	84,45	84,91
10	491	315	176	68	14	54	0	104	663	74,06	10,26	0,00	15,69	85,37	84,45	84,91
11	491	315	176	69	14	55	0	103	663	74,06	10,41	0,00	15,59	85,14	84,45	84,79
12	491	315	176	68	14	54	0	104	663	74,06	10,26	0,00	15,69	85,37	84,45	84,91
13	491	315	176	68	14	54	0	104	663	74,06	10,26	0,00	15,69	85,37	84,45	84,91
14	491	315	176	68	14	54	0	104	663	74,06	10,26	0,00	15,69	85,37	84,45	84,91
15	491	315	176	68	14	54	0	104	663	74,06	10,26	0,00	15,69	85,37	84,45	84,91

Tabelle 10.2: Klassifikationsergebnisse auf der Evaluierungsmenge – Self-Training mit Mehrheitsverfahren; mit einfacher Mehrheitsbildung

Runde	Acc (%)	falsch (%)	ohne (%)	Pre (%)	Rec (%)	F_1 (%)	nicht-geo	geo	ges	ohne	möglich
1	15,69	0,15	84,16	98,95	25,20	40,17	256	726	982	95910	96892
2	15,84	0,15	84,01	98,99	24,93	39,83	1211	149	1360	94550	95910
3	15,99	0,00	84,01	100,00	16,09	27,72	865	1110	1975	92575	94550
4	15,54	0,00	84,47	100,00	14,21	24,88	3315	394	3709	88866	92575
5	17,20	0,00	82,81	100,00	10,99	19,80	1472	975	2447	86419	88866
6	17,65	0,00	82,35	100,00	10,72	19,36	1329	245	1574	84845	86419
7	17,95	0,15	81,90	97,56	10,72	19,32	440	86	526	84319	84845
8	17,95	0,15	81,90	97,56	10,72	19,32	72	23	95	84224	84319
9	17,95	0,15	81,90	97,56	10,72	19,32	22	2	24	84200	84224
10	17,95	0,30	81,75	97,56	10,72	19,32	26	65	91	84109	84200
11	17,95	0,30	81,75	97,56	10,72	19,32	0	15	15	84094	84109
12	17,95	0,30	81,75	97,56	10,72	19,32	19	8	27	84067	84094
13	17,95	0,30	81,75	97,56	10,72	19,32	22	0	22	84045	84067
14	17,95	0,30	81,75	97,56	10,72	19,32	13	0	13	84032	84045
15	17,95	0,30	81,75	97,56	10,72	19,32	0	0	0	84032	84032

Tabelle 10.3: Klassifikationsergebnisse auf der Evaluierungsmenge – Self-Training mit Mehrheitsverfahren; wie in Tabelle 10.2, jedoch mit strenger Mehrheitenbildung. Zusätzlich sind Daten zur Selektion neu klassifizierter Beispiele angegeben; „nicht-geo“, „geo“ und „ges“ gibt die Anzahl der selektierten Beispiele an; „ohne“ gibt die Anzahl der Bsp. an, die ohne bekannte Feature sind und „möglich“ bezeichnet die zur Selektion vorhandenen Beispiele.

10.3 Bootstrap-Verfahren

Wie in Abschnitt 9.3 erläutert ergibt sich beim Bootstrap-Verfahren im Gegensatz zu den anderen Verfahren das beste Ergebnis bei der Feature-Pattern-Kombination b3, b2, b1, suff, case, a1, a2, b3,b1 und b2,b1,a1.

Deshalb wurde diese auch bei der Evaluierung verwendet. Als Grenzwert zur Selektion wurde 5000 Einwohner gewählt, da dies bei der Analyse die besten Ergebnisse lieferte. Auch wurde zur Evaluierung mit allen extrahierten Artikeln gearbeitet, weshalb die Berechnungen wieder auf einem UNIX-Server durchgeführt wurden. Die Klassifikationsergebnisse sind in den Tabellen 10.4, 10.5 und 10.6 dargestellt.

Im Gegensatz zu den anderen Verfahren zeigte sich beim Bootstrap-Verfahren eine minimale Verbesserung der Ergebnisse im Vergleich zur Analyse. Mit einem F_1-Wert von 89,61 ergibt sich eine Verbesserung von 14,61% im Vergleich zur Baseline. Allerdings ist das Bootstrap-Verfahren im Vergleich zum überwachten Verfahren, welches einen F_1-Wert von 93,23% lieferte, immer noch weit entfernt.

Wie schon in Abschnitt 10.1 beschrieben sind in der Evaluierungsmenge 143 kleingeschriebene nicht-geo Beispiele enthalten. Von diesen werden wieder alle korrekt klassifiziert. Das heißt, bei case-sensitivem Vorgehen werden von 520 Beispielen in der Evaluierungsmenge 409 korrekt klassifiziert. Es ergibt sich also lediglich eine Acc von 78,65%, wobei der F_1-Wert bei 89,61% bleibt.

Art.	Feature Pattern	mindest Einwohner	Lernm. 1. Stufe		Analyse 1. Stufe		
			geo	nicht-geo	korrekt	falsch	ges
15976	more	5000	167268	1451777	142	0	142

Tabelle 10.4: Teil 1 – Klassifikationsergebnis der Bootstrap-Verfahrens auf der Evaluierungsmenge

Lernm. 2.Stufe		Analyse 2. Stufe				
geo	nicht-geo	korrekt	falsch	falsch geo	falsch nicht-geo	ges
167268	167497	410	111	14	97	521

Tabelle 10.5: Teil 2 – Klassifikationsergebnis der Bootstrap-Verfahrens auf der Evaluierungsmenge

Gesamt Analyse								
korrekt	falsch	falsch geo	falsch nicht-geo	ges	Acc (%)	Pre (%)	Rec (%)	F_1 (%)
552	111	14	97	663	83,26	78,73	96,25	89,61

Tabelle 10.6: Teil 3 – Klassifikationsergebnis der Bootstrap-Verfahrens auf der Evaluierungsmenge

10.4 Analyse der Klassifikationsbeispiele

In den letzten Abschnitten wurde auf die Klassifikationsergebnisse bei der Evaluierung eingegangen. Das überwachte Verfahren und das Bootstrap-Verfahren liefern die besten Ergebnisse, weshalb auf sie noch etwas genauer eingegangen werden soll.

In Tabelle 10.9 sind die falsch klassifizierten geographischen Beispiele aus der Evaluierungsmenge aufgeführt. Dabei ist angegeben, wie oft sie beim überwachten bzw. beim Bootstrap-Verfahren falsch klassifiziert wurden. Auffällig ist, dass beim überwachten Verfahren mit 15 Fehlern sogar ein Beispiel mehr falsch klassifiziert wurde. Betrachtet man lediglich unterschiedliche Kandidaten, so sind es jedoch weniger als beim Bootstrap-Verfahren. Relativ überraschend ist, dass lediglich vier Beispiele von beiden Verfahren falsch klassifiziert werden, und ansonsten bei unterschiedlichen Beispielen Fehlklassifikationen auftreten.

Bei den falsch klassifizierten nicht-geo Beispielen ist der Anteil der von beiden Verfahren falsch klassifizierten mit 25 Beispielen hingegen schon höher (vergl. Tab 10.10). Bei den nicht-geo Beispielen zeigt sich außerdem, warum das überwachte Verfahren erheblich besser ist. Dieser Umstand muss wahrscheinlich darauf zurückgeführt werden, dass beim Bootstrap-Verfahren in der Trainingsmenge mehrdeutige Beispiele als geo Beispiele enthalten sind.

Soll mit dem Viesenhäuser **Hof** eine Bresche in die Regionalplanung
die Bebauung des Gebietes Viesenhäuser **Hof** zu keiner gütlichen Einigung zwischen
des Flächennutzungsplanes im Bereich Viesenhäuser **Hof** vorausgegangen sind . Als
, daß die Bebauung Viesenhäuser **Hof** in der Öffentlichkeit immer mit
Rechtsanwalt Bulling ist der Viesenhäuser **Hof** „ ein einmaliger Fall "
die Landesplaner mit dem Viesenhäuser **Hof** ein Exempel mit Signalwirkung

Tabelle 10.7: Bei der Evaluierung falsch klassifizierte Beispiele zu „Hof“.

Wie in der Tabelle 10.10 zu sehen ist, ergaben sich bei beiden Verfahren bei allen Vorkommen von „Hof“ Klassifikationsfehler. Dies lässt zunächst die Vermutung aufkommen, dass „Hof“ mit den angewendeten Verfahren nicht disambiguiert werden kann. Betrachtet man jedoch die Beispiele genauer (vergleiche Tabelle 10.7), so wird klar, dass es sich um sehr ähnliche, wenn nicht gar gleiche Beispiele handelt.

Eine weitere Auffälligkeit bei den falsch klassifizierten nicht-geo Beispielen ist, dass bei 13 Beispielen (überwachtes Verfahren) bzw. bei 27 Beispielen (Bootstrap-Verfahren) Personennamen als geo klassifiziert wurden. Das sind immerhin 35% bzw. 28% der falsch klassifizierten nicht-geo Beispiele.

Das überwachte Verfahren und das Bootstrap-Verfahren basieren beide auf SVM, welches für den Nutzer eine Blackbox darstellt. Das heißt, dass lediglich bekannt ist, was SVM übergeben wird und was SVM zurückliefert. Hierdurch wird es sehr schwierig für einzelne Beispiele zu analysieren, warum sie als geo bzw. nicht-geo klassifiziert werden. Auch kann nicht bestimmt werden, welche Feature für die Klassifikation ausschlaggebend sind.

In Tabelle 10.8 sind alle in der Evaluierungsmenge vorkommenden Beispiele zu Wiesbaden aufgeführt. Zusätzlich ist zu jedem Text die Feature-Repräsentation für das überwach-

Bundesregierung für Fehlbelegungsabgabe **WIESBADEN** (dpa) . Die b2für b1Fehlbelegungsabgabe XX EN DEN ADEN BADEN SBADEN a1(a2dpa	k
Sozialhilfe um 9,7 Prozent gestiegen **WIESBADEN** (AP) . Die b2Prozent b1gestiegen XX EN DEN ADEN BADEN SBADEN a1(a2AP	k
berichtete das Statistische Bundesamt in **Wiesbaden** . Die Ausgaben für die b2Bundesamt b1in Xxx en den aden baden sbaden a1. a2Die	k
stellvertretende Schauspieldirektor am Hessischen Staatstheater **Wiesbaden** , Michael W. Schlicht , b2Hessischen b1Staatstheater Xxx en den aden baden sbaden a1, a2Michael	f

Tabelle 10.8: Klassifikationsbeispiele und ihre Repräsentation durch Feature für das überwachte Verfahren; k steht für korrekt, f für falsch klassifiziert.

te Verfahren mitaufgeführt. Die ersten drei Beispiele wurden vom überwachten Verfahren korrekt, das letzte Beispiel jedoch falsch klassifiziert.

Sichere Aussagen darüber, warum die Beispiele dem Ergebnis entsprechend klassifiziert wurden, können aus dem genannten Grund nicht getroffen werden. Vermutet werden kann jedoch, dass bei den ersten beiden Beispielen case (XX) und suff (EN DEN ADEN BADEN SBADEN) die ausschlaggebenden Feature waren. Beim dritten Beispiel wird vermutlich das Feature b1 (in) ausschlaggebend gewesen sein. Diese Vermutung kann aufgestellt werden, da im vierten Beispiel eine ähnliche Konstellation vorliegt. Anstelle von „in“ steht jedoch ein viel selteneres Wort vor dem Kandidaten, weshalb es im Gegensatz zum dritten Beispiel als nicht-geo klassifiziert wurde.

Wie schon geschildert können nur Vermutungen, wie die aufgeführten, über einzelne Beispiele angestellt werden, daher wird an dieser Stelle auf weitere Interpretationen verzichtet. Um dem Leser zumindest ein Gefühl für die Beispielkontexte und die Klassifikationsergebnisse zu vermitteln, sind in den Tabellen 10.12 und 10.11 einige weitere Kontextbeispiele für falsch und für korrekt klassifizierte Beispiele aufgeführt.

Baltmannsweiler		1	Hessen	2		Saarlouis		1
Darmstadt-Dieburg		1	Hofen	1		Sillenbuch	1	
Fehmarn	1	1	Messel	2	2	Stölten		1
Filderstadt		1	Platz	1		Wiesbaden	1	
Freudenstadt		1	Remshalden		1	Winnenden		1
Grube	1		Saarbrücken		2			
Hallschlag	1	1	Saarland	4				

Tabelle 10.9: Bei der Evaluierung durch das überwachte und das Bootstrap-Verfahren falsch klassifizierte geo Beispiele im Vergleich. Hinter den Beispielen ist jeweils die Anzahl der falsch klassifizierten Bsp. beim überwachten Verfahren und dahinter beim Bootstrap-Verfahren angegeben. Insgesamt wurden 15 Bsp. beim überwachten und 14 Bsp. beim Bootstrap-Verfahren bzw. 10 und 12 verschiedene Bsp. falsch klassifiziert.

Achim		3	P.	Haft		1		Regen	1	4	
Alfred	1	1	P.	Hart	1	1		Reichenbach	3	3	P.
Aura	1			Häusern	1			Reuter		2	P.
Boden		3		Hilden		7	P.	Schlauch	2	2	P.
Brand		4		Hütten		1		Schön		2	
Bus	1	1		Hof	6	6		Sinn		1	
Bühler	1	1	P.	Horst		2	P.	Sorgen	1		
Cronenberg	1	1	P.	Karin	1		P.	Stelle		3	
Damm	1			Kolonie	1	1		Stock	1	1	
Drogen	2			Lage		2		Wasser		2	
Eck	1	1		Lohr	2	3	P.	Weg		12	
Ernst		4	P.(2)	Norden		3		Weil		3	
Gräben	1			Ostheim	1	1		Weng	1		P.
Grube	1			Platz		8		Wetter		3	
Grund		1		Pöhl	2		P.	Wüstenrot	2	3	A.

Tabelle 10.10: Bei der Evaluierung durch das überwachte und das Bootstrap-Verfahren falsch klassifizierte nicht-geo Beispiele im Vergleich. Hinter den Beispielen ist jeweils die Anzahl der falsch klassifizierten Bsp. beim überwachten Verfahren und dahinter beim Bootstrap-Verfahren angegeben. Insgesamt wurden 37 Bsp. beim überwachten und 97 Bsp. beim Bootstrap-Verfahren bzw. 25 und 35 verschiedene Bsp. falsch klassifiziert. Personennamen wurden mit „P.“ markiert. „A.“ markiert falsch annotierte Beispiele.

Kontext	überw.	boot.
das aus der ehemaligen niederländischen **Kolonie** Surinam stammt . Ein großer	f	f
und Chorzow ist das schmutzigste **Eck** Polens . Zabrze wurde inzwischen	f	f
Der frühere SEL-Chef **Lohr** bleibt in Haft STUTTGART (	f	f
Übereinstimmung mit dem Landgericht bei **Lohr** sowohl den dringenden Tatverdacht als	f	f
(SEL) , Helmut **Lohr** , bleibt im Gefängnis .	k	f
Traub-Stadion wegen Kupferschlacke weiter zu **REICHENBACH** , Kreis Esslingen . Das	k	k
Esslingen . Das Hermann-Traub-Stadion in **Reichenbach** bleibt weiter gesperrt . Eine	k	k
des sächsischen CDU-Landesverbandes , Klaus **Reichenbach** , will auf dem nächsten	f	f
. Einen entsprechenden Verzicht hatte **Reichenbach** auf einer Präsidiumssitzung des Landesverbandes	f	f
des Gremiums . Die von **Reichenbach** vorgetragenen persönlichen Gründe , er	f	f
Nachfolger von Bundesbank-Präsident Karl Otto **Pöhl** soll sein bisheriger Stellvertreter Helmut	f	k
Vorentscheidung über die Nachfolge von **Pöhl** fallen . Zuvor befasse sich	f	k
im Jahr zuvor . Intendant **Achim** Plato führte dies auf die	k	f
Shakespeares Sommernachtstraum (Regie : **Achim** Plato) mit der Musik	k	f
diesmal regennasse Zehn-Kilometer-Strecke Fotos : **Achim** Zweygarth	k	f
Gefährlicher **Ernst** Zum Thema „ Geplante Schließung	k	f
, es sieht nach gefährlichem **Ernst** aus . Kein Geld mehr	k	f
die Verantwortung der Frauen ganz **ernst** nimmt und ihnen auch nicht	k	k
In diesem Punkt sieht Oberbürgermeister **Ernst** Fischer den „ Pferdefuß ”	k	f
die Arbeit der Ausländerbeauftragten nicht **ernst** genug nehme . Die jetzige	k	k
MAX **ERNST** : „ Der Hausengel ”	k	f

Tabelle 10.11: Nicht-geographische Kontextbeispiele; k steht für korrekt, f für falsch klassifiziert.

Kontext	überw.	boot.
die neue rot-grüne Regierung in **Hessen** die Pläne ihrer Landesbank und	k	k
mit den übermächtigen , finanzstarken **Hessen** gut beraten sind oder nur	f	k
Girozentrale ausgemacht hätte . Für **Hessen** ist die Ausdehnung des Tätigkeitsfelds	f	k
(43 Hektar) , **Filderstadt** (4,8 Hektar) und	k	f
, der 1940 beim Bahnhof **Freudenstadt** seine Eisenbahnerlaufbahn begann , Bundesbahnamtsrat	k	f
auf 4,09 Meter , in **Karlsruhe** um einen halben Meter auf	k	k
Nebelwarnungen für die A 8 **Karlsruhe** – Stuttgart zwischen dem Dreieck	k	k
– Stuttgart zwischen dem Dreieck **Karlsruhe** und Heimsheim und für die	k	k
Monaten in einem Kaufhaus in **Karlsruhe** einen 42jährigen Niedersachsen festnehmen ,	k	k
Diebe klauen Solarzellen der Verkehrszählstation **KARLSRUHE** (lsw) . Geradezu	k	k
Verkehrszählanlage an der Autobahn bei **Karlsruhe** abmontiert haben . Wie die	k	k
haben . Wie die Autobahnpolizei **Karlsruhe** mitteilte , dienen die zwischen	k	k
Böblingen und Esslingen . In **Karlsruhe** dagegen liege die Obergrenze bei	k	k
Institut für Regionalwissenschaft der Universität **Karlsruhe** . Nach Ansicht von Bulling	k	k
des französischen Kosmetikherstellers L'Oréal (**Karlsruhe**) hat das gesamte Aktienkapital	k	k
strebt Musterklage gegen Sonderregelung im **Saarland** an Um eines können Arbeitnehmer	f	k
. Und es gibt im **Saarland** eine weitere einmalige soziale Errungenschaft	f	k
Davon ausgenommen sind nur im **Saarland** beschäftigte Bundesbedienstete und beamtete Lehrkräfte	f	k
schließlich gibt es ja im **Saarland** beispielsweise auch ein Bildungsurlaubsgesetz mit	f	k

Tabelle 10.12: Geographische Kontextbeispiele; k steht für korrekt, f für falsch klassifiziert.

Kapitel 11

Disambiguierungs- und Visualisierungs-Prototyp

Wie in Abschnitt 1.2 bereits erläutert soll im Rahmen dieser Arbeit ein Prototyp erstellt werden, der die als geographische Entitäten erkannten Kandidaten eines Textes mit Hilfe von OpenJUMP visualisiert. Wie im letzten Kapitel deutlich wurde, ergaben sich beim überwachten Verfahren die besten Resultate. Außerdem lieferte das Bootstrap-Verfahren ebenfalls eine Verbesserung. Daher wurde die Implementierung des Prototypen für diese beiden Verfahren vorgenommen.

Im nächsten Abschnitt wird die Implementierung des Prototypen vorgestellt. Im Abschnitt 11.2 wird daraufhin anhand eines Beispiels die Funktionsweise weiter verdeutlicht.

11.1 Implementierung

In Abschnitt 5.1 wurde erläutert und in Abbildung 5.3 dargestellt, welcher Ansatz im Rahmen dieser Arbeit verfolgt wurde. Bis auf geringe Teile der prototypischen Implementierung wurden alle benötigten Komponenten für den Prototypen bereits beim Erstellen der Lernverfahren implementiert und in den entsprechenden Abschnitten erläutert.

Daher soll an dieser Stelle nur noch auf einige Schnittstellen eingegangen werden. Außerdem wird kurz erläutert, wie die benötigten Komponenten zusammengeführt wurden.

Wie in Abschnitt 3.5 schon beschrieben verfügt OpenJUMP zur Erweiterung über ein Plugin-System. Im Rahmen der Arbeit [BKN+06] wurde bereits ein OpenJUMP-Plugin zur Visualisierung von Straßen erstellt. Hierbei wurden ebenfalls die Daten aus der Datenbasis der Firma Navteq verwendet. Daher konnte das Plugin für die Aufgabenstellung in dieser Arbeit angepasst und zur Implementierung des Prototypen herangezogen werden.

Die eigentliche Logik des Prototypen wurde in der Klasse VisuDisambi implementiert. Diese Klasse enthält einen Konstruktor, der bei der Initialisierung der Klasse dazu führt, dass die Gazetteer Nav und Deu und die SVM-Funktionalitäten geladen werden. Außerdem sind in VisuDisambi neben dem beschriebenen Konstruktor die Prozeduren getDisambiguated und getDisambiguatedSup enthalten. Sie beinhalten die Implementierung des Disambiguierungsvorgangs für das Bootstrap-Verfahren bzw. für die überwachte Variante. Ihnen

wird ein Text übergeben, aus dem mit Hilfe der Gazetteer die Kandidaten extrahiert werden. Im Fall von getDisambiguatedSup werden diese mit SVM und einem bereitgestellten Modell einfach klassifiziert. Im Fall von getDisambiguated kommt das zweistufige Verfahren zum Einsatz, für das entsprechend zwei Modelle zur Verfügung gestellt werden müssen.

Die Ergebnisse der Disambiguierung werden in beiden Fällen in einer Liste von Ergebnisobjekten (der Klasse Result) zurückgegeben, welche dann mit OpenJUMP visualisiert werden. Diese Objekte enthalten den Kandidaten als String und die Klassifikation als geo oder nicht-geo (in einem Boolean). Außerdem ist der SVM-Klassifikationswert enthalten, wobei bei der Bootstrap-Variante mitangegeben ist, in welcher Klassifikationsstufe der Kandidat klassifiziert wurde.

Auf diese Weise kann der Prototyp für weitere Analysen und Experimente genutzt werden. Er kann außerdem als Anwendung verwendet werden, um zum Beispiel graphisch zu verdeutlichen an welchen Orten eine als Text übergebene Geschichte spielt.

Angemerkt sei an dieser Stelle noch, dass im Prototypen keine geo / geo Mehrdeutigkeiten aufgelöst werden, da dies nicht Teil der Aufgabenstellung dieser Arbeit ist. Auch können lediglich geographische Entitäten aus Baden-Württemberg visualisiert werden, da die Datenbasis (der Firma Navteq) lediglich Daten zu Baden-Württemberg enthält.

Um dem Leser den Prototypen weiter vorzustellen, wird im nächsten Abschnitt ein Beispiele anhand eines Screenshots vorgestellt.

11.2 Beispiel

In Abbildung 11.1 ist ein Screenshot des Prototypen dargestellt. Das obere Fenster in Abbildung 11.1 ist das Applikations-Fenster von OpenJUMP und zeigt die Karte von Baden-Württemberg, in die die Landkreise eingezeichnet sind. Wie ersichtlich wird der Landkreis Böblingen durch Einfärbung und die Ortschaften Böblingen, Bohndorf, Deckenpfronn, Waldenbuch, Oberjettingen und Jettingen durch etwas dunklere Einfärbung als gefundene geographische Entität hervorgehoben. Wie bereits erläutert wird durch den Prototypen keine geo / geo Auflösung durchgeführt. Daher werden sowohl der Landkreis als auch die Ortschaft Böblingen markiert.

Im unteren Teil des in Abbildung 11.1 gezeigten Screenshots ist das Fenster zur Bedienung des Prototypen zu sehen. Im oberen Teil dieses Fensters kann ausgewählt werden, welches Verfahren zum Einsatz kommen soll. Darunter kann in ein Textfeld ein Text zur Disambiguierung und Visualisierung eingegeben werden, was in Abbildung 11.1, wie ersichtlich, beispielhaft durchgeführt wurde.

Der beispielhaft eingegebene Text wurde disambiguiert, wobei, wie im Bedienungsfenster rechts oben zu sehen ist das Bootstrap-Verfahren zum Einsatz kam. Nach dem Disambiguierungsvorgang werden im Textfeld alle Kandidaten zur Disambiguierung markiert. Im Beispieltext in Abbildung 11.1 wurde „Weissach“ nicht markiert, weil es nicht in den Gazetteer vorkommt, und deshalb nicht als Kandidat erkannt wird.

Im unteren Teil des Bedienungsfensters ist eine Tabelle zu sehen, in der alle Kandidaten und ihre Zuordnung zu geo bzw. nicht-geo aufgeführt sind. Außerdem werden die SVM-

Klassifikationswerte in der Tabelle mitaufgeführt. Beim Bootstrap-Verfahren wird dabei die Klassifikation in der ersten Stufe (c1) und in der zweiten Stufe (c2) unterschieden. „Osten“ beispielsweise wurde in der ersten Stufe mit einem SVM-Wert von -1,264 korrekt als nicht-geographisch bestimmt. „Norden“ hingegen wurde in der zweiten Stufe fälschlicherweise als geo klassifiziert, wie in Abbildung 11.1 zu sehen ist.

Die Ortschaften Böblingen, Bondorf, Deckenpfronn, Waldenbuch, Oberjettingen und Jettingen werden im dargestellten Beispiel korrekt klassifiziert und, wie bereits beschrieben, durch Einfärbung auf der Karte visualisiert.

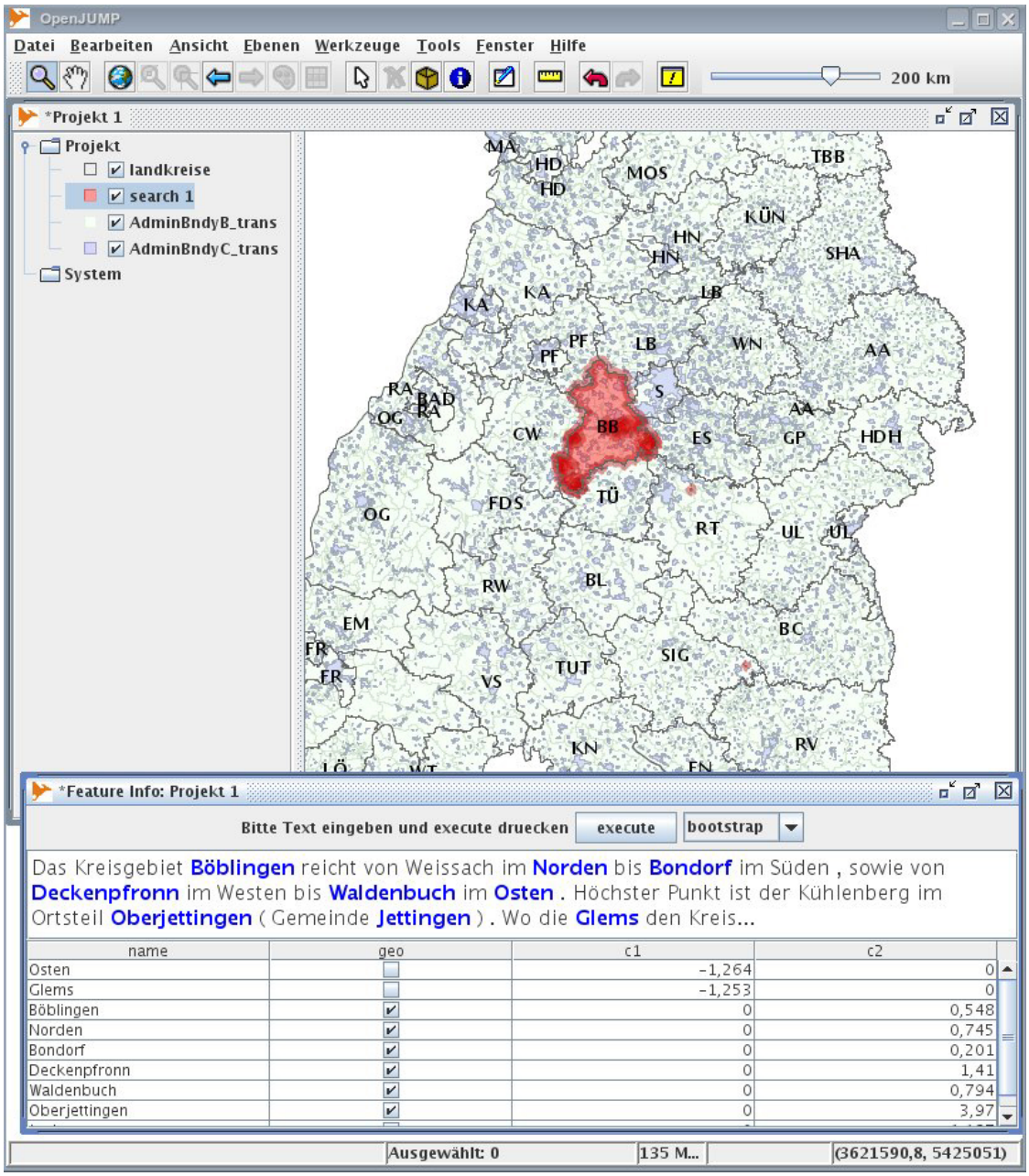

Abbildung 11.1: Screenshot des Disambiguierungs- und Visualisierungs-Prototypen.

Kapitel 12

Zusammenfassung und Ausblick

Abschließend soll in diesem Kapitel zusammengefasst werden, welche Ergebnisse in der vorliegenden Diplomarbeit erzielt wurden. Außerdem sollen im Ausblick einige Möglichkeiten aufgezeigt werden, wie diese Arbeit weitergeführt werden kann und welche weiteren Untersuchungen sinnvoll wären.

12.1 Zusammenfassung

Ziel dieser Arbeit war es, Möglichkeiten zur Auflösung von geo / nicht-geo Ambiguitäten im Deutschen zu erarbeiten.

Hierzu wurden grundlegende Untersuchungen durchgeführt. Dabei ergab sich, dass bei der Verwendung von Nav und Deu und einer case-insensitiven Suche eine Verteilung von ca. 60% geo und 40% nicht-geo Kandidaten zu erwarten ist. Bei case-sensitiver Suche hingegen liegt eine Verteilung von 75% geo und 25% nicht-geo zugrunde. Außerdem zeigte sich bei den Untersuchungen, dass Kandidaten, die in einem Artikel mehrfach vorkommen, in aller Regel entweder alle geo oder alle nicht-geo sind. Das heißt, dass die Annahme, dass Kandidaten eine Bedeutung pro Artikel aufweisen, eine gute Heuristik darstellt.

Auf der Basis dieser Untersuchungen wurde zunächst versucht, einfache Regeln zum Auflösen der Mehrdeutigkeiten zu finden. Hierbei stellte sich heraus, dass ein Disambiguieren anhand einfacher Regeln nicht möglich ist, weil die vorhandenen Regelmäßigkeiten im Kontext nicht trivial sind. Daher wurde der Ansatz verfolgt, mit Hilfe von Methoden des maschinellen Lernens, ähnlich wie bei der Eigennamenerkennung bzw. der Textklassifikation, ein Modell zur Disambiguierung zu berechnen. Hierzu wurden verschiedene rein kontextuelle Repräsentationsmöglichkeiten der Kandidaten entworfen und untersucht.

Unter Verwendung eines überwachten Lernverfahrens auf der Basis von SVM ergab sich, dass die Feature-Pattern-Kombination (vergl. Abschnitt 6.4) b2, b1, suff, case, a1 und a2 mit einem F_1-Wert von 93,23% und einer Acc von 90,00% die besten Disambiguierungsergebnisse liefert. Ein Auflösen von geo / nicht-geo Ambiguitäten ohne linguistische Feature zu verwenden ist, wie dieses Ergebnis zeigt, mit recht hoher Erfolgsrate möglich.

Im Rahmen dieser Arbeit wurden außerdem verschiedene Ansätze verfolgt, mit einer möglichst kleinen handannotierten Lernmenge, oder sogar ohne eine solche auszukommen.

Hierfür wurde ein auf dem überwachten SVM-Verfahren und ein auf einem Mehrheitsverfahren basierendes Self-Training-Verfahren implementiert. Beim auf SVM basierenden Self-Training-Verfahren konnte zwar die Trainingsmenge vergrößert werden, hierdurch ergab sich jedoch keine Verbesserung der Disambiguierungsergebnisse im Vergleich zum überwachten Lernverfahren bzw. der ersten Lernrunde.

Beim Self-Training mit Mehrheitsverfahren ergab sich eine minimale Verbesserung auf einen F_1-Wert von 87,32% im Vergleich zur ersten Lernrunde mit 86,40% F_1. Insgesamt ist das Ergebnis des Self-Trainings somit weit schlechter, als das überwachte, auf SVM basierende Verfahren mit einem F_1-Wert von 93,23%.

Neben den Self-Training-Verfahren wurde ein Bootstrap-Verfahren implementiert, bei welchem die Trainingsmenge rein maschinell generiert wird. Auf dieser wird mit Hilfe eines zweistufigen auf SVM basierenden Verfahren gelernt. Hierbei ergab sich bei der Verwendung der Feature-Pattern-Kombination b3, b2, b1, suff, case, a1, a2, b3,b1 und b2,b1,a1 und casesensitiver Betrachtung mit einem F_1-Wert von 89,61 und einer Acc von 78,65% ein relativ gutes Ergebnis, auch wenn es noch weit hinter dem überwachten Verfahren zurückbleibt.

Wie beschrieben wurde sowohl für das überwachte Verfahren als auch für das Bootstrap-Verfahren, welches die besten Ergebnisse der nicht überwachten Verfahren lieferte, eine prototypische Implementierung zur Visualisierung der Ergebnisse mit OpenJUMP erstellt. Diese kann beispielsweise für weitere Analysen und Experimente eingesetzt werden.

12.2 Fazit

Im Rahmen dieser Arbeit konnte gezeigt werden, dass es mit recht hoher Erfolgsrate möglich ist, ohne linguistische Feature, lediglich mit einer einfachen textuellen Repräsentation des Kontextes und des Kandidaten als Feature, geo / nicht-geo Ambiguitäten im Deutschen aufzulösen. Mit dem überwachten Verfahren wurde ein einfaches, mit einer Verbesserung des F_1-Wertes von 18,23%, recht gutes Verfahren zur Disambiguierung gefunden.

Grundlegende Möglichkeiten zur Disambiguierung und zur Repräsentation der Kandidaten konnten mit der Entwicklung der Feature-Pattern aufgezeigt werden.

Es konnte zwar kein Self-Training- bzw. Bootstrap-Verfahren erarbeitet werden, dessen Ergebnisse mit den Ergebnissen des überwachten Verfahrens vergleichbar sind, jedoch ist mit dem Bootstrap-Verfahren, welches bereits eine Verbesserung von 14,61% des F_1-Wertes liefert, ein grundlegendes Verfahren gefunden worden, dass Potential zur weiteren Verbesserung birgt. Daher konnte für weitere Untersuchungen und das Bilden eines nicht überwachten Verfahrens mit den in dieser Arbeit durchgeführten Untersuchungen und dem Entwickeln der Self-Training-Verfahren, insbesondere aber dem Bootstrap-Verfahren, ein guter Grundstein gelegt werden.

Mit dem Prototypen zur Disambiguierung und Visualisierung ist zudem ein Werkzeug erstellt worden, das für weitere Experimente und Untersuchungen hilfreich ist und als einfache Applikation bereits verwendet werden kann.

12.3 Ausblick

Das überwachte Verfahren lieferte, wie schon geschildert, die besten Disambiguisierungsergebnisse. Da die Beispiele für die Lernmenge zufällig ausgewählt wurden und die Lernmenge recht klein war, ist zu vermuten, dass die Lernmenge und somit die Ergebnisse des Verfahrens weiter verbessert werden können. Daher wäre es sicherlich sinnvoll zu untersuchen, wie eine repräsentative Lernmenge zur Disambiguierung aussehen müsste, so dass anhand einer relativ kleinen handannotierten Lernmenge eventuell ein noch besseres Klassifikationsergebnis erreicht werden könnte.

Eine weitere Möglichkeit das überwachte Verfahren, aber auch das Bootstrap-Verfahren weiter zu verbessern, wäre zu analysieren, ob die Feature-Pattern noch verbessert werden können. Hierzu sollte eine Möglichkeit gefunden werden zu analysieren, welche Feature in welchem Maß zur Klassifikation eines Beispieles beigetragen haben. Ist eine solche Möglichkeit – trotz der Blackbox SVM – gefunden, könnte beispielsweise analysiert werden, ob wegen einzelner Feature einzelne Beispiele falsch klassifiziert werden, oder lediglich weil ein Kontext nicht ausreichend bekannt ist.

Wie schon erläutert sollte außerdem geprüft werden, ob eine im Rahmen dieser Arbeit nicht analysierte Feature-Pattern-Kombination möglicherweise zu besseren Ergebnissen führt.

Das Bootstrap-Verfahren liefert mit einer Verbesserung des F_1-Werte um lediglich 14,61% im Vergleich zur Baseline kein überwältigendes Ergebnis. Wie sich anhand der Berechnungsunterschiede zwischen UNIX und Windows zeigte, wirkt sich eine Änderung des Grenzwertes zum Aussortieren extremer nicht-geo Beispiele recht stark auf das gesamte Klassifikationsergebnis aus. Daher erscheint es sinnvoll, weiterführende Untersuchungen zum Berechnen eines optimalen Grenzwertes durchzuführen, so dass eventuell ein Näherungsverfahren implementiert werden kann.

Wie in Abschnitt 10.4 diskutiert ist zu vermuten, dass das schlechtere Ergebnis des Bootstrap-Verfahrens darauf zurückzuführen ist, dass die mehrdeutigen Beispiele als geo Beispiele in der Lernmenge sind. Daher empfiehlt sich zu untersuchen, ob diese Beispiele nach einem Klassifizieren durch SVM nahe der geo / nicht-geo Grenze in der Lernmenge liegen, ähnlich wie die extremen nicht-geo Beispiele in einem Extrabereich liegen. Ist dies der Fall, könnte durch eine Verschiebung der Grenze bzw. der SVM-Werte eventuell eine weitere Verbesserung des Bootstrap-Verfahrens herbeigeführt werden. Außerdem macht es Sinn zu überprüfen, ob eine weitere Vergrößerung der Beispielmenge zum Erzeugen der Lernmenge zu weiter verbesserten Ergebnissen führt.

Werden auch Straßen, Berge, etc. disambiguiert, wäre eine weitere Möglichkeit eventuell Verbesserungen bei der Disambiguierung zu erzielen, Querverbindungen zwischen den verschiedenen Entitäten zu nutzen. Wird beispielsweise in einem Kontext von „Süßen“ eine Straße erkannt, die in „Süßen“ liegt, wäre es sicher sinnvoll, diese Information zur Disambiguierung von „Süßen“ miteinzusetzen.

Ein weiterer Aspekt, dessen genauere Betrachtung sicher lohnend ist, ist das Auflösen von geo / geo Ambiguitäten. Wie im Zuge dieser Arbeit bereits diskutiert, hängen die geo / geo und die geo / nicht-geo Auflösung zum Teil eng zusammen. Zum anderen muss die

geo / geo Disambiguierung durchgeführt werden, um ein Kontextmodell korrekt aktualisieren zu können. Auch zur Visualisierung wäre es sinnvoll, die geo / geo Disambiguierung durchzuführen.

Abbildungsverzeichnis

2.1 Beispiel Informationsextraktion 6
2.2 Entitäten und ihre Entitätstypen 7
2.3 Überwachtes Lernen 10
2.4 Technik des Self-Trainings 11

3.1 Gazetteer Datei Spalten (vergl. [*WG]) 18

5.1 Grundlegender Ansatz 29
5.2 Einfache Regeln zur Disambiguierung 29
5.3 Ansatz mit maschinellem Lernen 30
5.4 Beispielartikel 1 (Quelle: STZ) 31
5.5 Beispielartikel 2 (Beginn eines Artikel) (Quelle: STZ) 31

6.1 Objekte aus zwei verschiedenen Klassen als Punkte in einem Vektorraum, stark vereinfacht und zweidimensional dargestellt. 36
6.2 Möglichkeiten, Klassen im Vektorraum zu trennen – mit den Geraden sind verschiedene Hyperebenen vereinfacht dargestellt, um zu zeigen, dass es sehr viele Möglichkeiten gibt, die dargestellten Klassen zu trennen. 36
6.3 Optimale Hyperebene – durch Maximierung des Abstandes zu allen Datenpunkten bzw. durch Maximierung des Randes (Doppelpfeil) wird mit SVM eine optimale Hyperebene (dicke Gerade) berechnet. Die hell dargestellten Datenpunkte bilden die Support Vektoren. 36

7.1 Analyseläufe – Self-Training mit SVM; die Läufe wurden mit 564, 288, 211, 182 und 116 Beispielen in der Boot-Menge durchgeführt. 47
7.2 Analyseläufe – Self-Training mit SVM; die Anzahl der Trainingsbeispiele für alle Lernrunden der Läufe 1 und 5 sind im Vergleich zur Größe der Trainingsmengen bei einer minimalen Selektion von fünf Beispielen pro Runde dargestellt. 48
7.3 Analyseläufe – Self-Training mit SVM; für die Läufe 1 und 5 sind die größten und kleinsten Werte der in jeder Runde zur Selektion bestimmten besten fünf Beispiele dargestellt. 48

7.4 Analyseläufe – Self-Training mit SVM; dargestellt sind die Acc-Werte der Läufe 5, 6 und 7, wobei die Darstellung der letzten beiden bezüglich der Runden gestreckt ist. Hierdurch soll eine bessere Vergleichbarkeit, basierend auf ähnlichen Größen der Lernmengen, erreicht werden. 49
7.5 Analyseläufe – Self-Training mit SVM; dargestellt sind zu den Läufe 5, 6 und 7 die Mittelwerte aus den größten und kleinsten SVM-Werten der in jeder Runde zur Selektion bestimmten besten fünf Beispiele. Dabei wurde die Darstellung der Läufe 6 und 7 bezüglich der Runden gestreckt. Hierdurch soll eine bessere Vergleichbarkeit, basierend auf ähnlichen Größen der Lernmengen, erreicht werden. 50

9.1 Mengenbildung zum maschinellen Erzeugen der Trainingsmenge 61
9.2 Erhoffte Aufteilung der Trainingsmenge durch SVM-Gewichte 62

11.1 Screenshot des Disambiguierungs- und Visualisierungs-Prototypen. 78

Tabellenverzeichnis

3.1 Beispiele mehrdeutiger Ortschaftsnamen bei Ortschaften aus Baden-Württemberg. 15

3.2 Grundlegende Untersuchungsergebnisse zu den 450 annotierten Artikeln . . 20

3.3 Grundlegende Untersuchungsergebnisse zu allen extrahierten Artikel des STZ einschließlich der annotierten Artikel 20

4.1 Regeln, wie sie in [Mih03] zum Selektieren von Kandidaten in Webseitentexten herangezogen werden. Zusätzlich ist jeweils ein Beispiel für die Regeln angegeben. 25

5.1 Einfache Repräsentation von Kandidaten für die Disambiguierung mit vier Worten vor und zwei Worten nach den Kandidaten. 32

5.2 Repräsentation zur Disambiguierung, bei der die Wortposition mitabgebildet wird. 32

5.3 Beispielhaft dargestellte Repräsentationsmöglichkeiten des Case-Feature-Pattern. 33

6.1 Daten zu den annotierten Artikeln und ihrer Aufteilung in die Lern-, Analyse- und Evaluierungsmenge. 40

6.2 Analyse zu sieben Feature-Pattern 40

6.3 Teil 1 – Analyse zu allen (19) im Rahmen dieser Arbeit implementierten Feature-Pattern 42

6.4 Teil 2 – Analyse zu allen (19) im Rahmen dieser Arbeit implementierten Feature-Pattern 43

6.5 Analyse zu 12 Feature-Pattern 44

8.1 Chi-Square-Tafel 52

8.2 Klassifikationsergebnisse – Self-Training mit Mehrheitsverfahren; mit einfacher Mehrheitsbildung 56

8.3 Klassifikationsergebnisse – Self-Training mit Mehrheitsverfahren; wie in Tabelle 8.2, jedoch mit strenger Mehrheitenbildung. Zusätzlich sind Daten zur Selektion von neu klassifizierten Beispielen angegeben; „nicht-geo“, „geo“ und „ges“ gibt die Anzahl der selektierten Beispiele an; „ohne“ gibt die Anzahl der Bsp. an, die ohne bekannte Feature sind und „möglich“ bezeichnet die zur Selektion vorhandenen Beispiele. 57

9.1 Liste nicht-geographischer Worte für die maschinelle Erzeugung einer Lernmenge 58
9.2 Liste geographischer Worte für die maschinelle Erzeugung einer Lernmenge 59
9.3 Vergleich der Ergebnisse überwachter SVM-Klassifikationen mit unterschiedlichen Lernmengen – einer handannotierten Lernmenge und einer mit Hilfe von Listen maschinell generierten Lernmenge 59
9.4 Unterschiede bei der Klassifikation auf einem UNIX- und einem Windows-System 64
9.5 Teil 1 – Klassifikationsergebnisse des Bootstrap-Verfahrens 65
9.6 Teil 2 – Klassifikationsergebnisse des Bootstrap-Verfahrens 65
9.7 Teil 3 – Klassifikationsergebnisse des Bootstrap-Verfahrens 65

10.1 Ergebnisse der Klassifikation mit der Evaluierungsmenge 66
10.2 Klassifikationsergebnisse auf der Evaluierungsmenge – Self-Training mit Mehrheitsverfahren; mit einfacher Mehrheitsbildung 68
10.3 Klassifikationsergebnisse auf der Evaluierungsmenge – Self-Training mit Mehrheitsverfahren; wie in Tabelle 10.2, jedoch mit strenger Mehrheitenbildung. Zusätzlich sind Daten zur Selektion neu klassifizierter Beispiele angegeben; „nicht-geo“, „geo“ und „ges“ gibt die Anzahl der selektierten Beispiele an; „ohne“ gibt die Anzahl der Bsp. an, die ohne bekannte Feature sind und „möglich“ bezeichnet die zur Selektion vorhandenen Beispiele. 69
10.4 Teil 1 – Klassifikationsergebnis der Bootstrap-Verfahrens auf der Evaluierungsmenge 70
10.5 Teil 2 – Klassifikationsergebnis der Bootstrap-Verfahrens auf der Evaluierungsmenge 70
10.6 Teil 3 – Klassifikationsergebnis der Bootstrap-Verfahrens auf der Evaluierungsmenge 70
10.7 Bei der Evaluierung falsch klassifizierte Beispiele zu „Hof“. 71
10.8 Klassifikationsbeispiele und ihre Repräsentation durch Feature für das überwachte Verfahren; k steht für korrekt, f für falsch klassifiziert. 72
10.9 Bei der Evaluierung durch das überwachte und das Bootstrap-Verfahren falsch klassifizierte geo Beispiele im Vergleich. Hinter den Beispielen ist jeweils die Anzahl der falsch klassifizierten Bsp. beim überwachten Verfahren und dahinter beim Bootstrap-Verfahren angegeben. Insgesamt wurden 15 Bsp. beim überwachten und 14 Bsp. beim Bootstrap-Verfahren bzw. 10 und 12 verschiedene Bsp. falsch klassifiziert. 73

10.10 Bei der Evaluierung durch das überwachte und das Bootstrap-Verfahren falsch klassifizierte nicht-geo Beispiele im Vergleich. Hinter den Beispielen ist jeweils die Anzahl der falsch klassifizierten Bsp. beim überwachten Verfahren und dahinter beim Bootstrap-Verfahren angegeben. Insgesamt wurden 37 Bsp. beim überwachten und 97 Bsp. beim Bootstrap-Verfahren bzw. 25 und 35 verschiedene Bsp. falsch klassifiziert. Personennamen wurden mit „P.“ markiert. „A.“ markiert falsch annotierte Beispiele. 73
10.11 Nicht-geographische Kontextbeispiele; k steht für korrekt, f für falsch klassifiziert. 74
10.12 Geographische Kontextbeispiele; k steht für korrekt, f für falsch klassifiziert. 75

Literaturverzeichnis

[*AG] *Alexandria Digital Library Project - Gazetteer.* Website, University of California, Santa Barbara.
unter: `http://www.alexandria.ucsb.edu/gazetteer/#name-list` (07. Dez 2006).

[AHSS04] Einat Amitay, Nadav Har'El, Ron Sivan, and Aya Soffer. *Web-a-Where: Geotagging Web Content.* In *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conferende on Research and Development in Information Retrival, Sheffield, UK, July 25-29, 2004*, pages 273–280. ACM, 2004.
unter: `http://doi.acm.org/10.1145/1008992.1009040` (28. März 2007).

[BKN+06] André Blessing, Stefan Klatt, Daniela Nicklas, Steffen Volz, and Hinrich Schütze. *Language-Derived Information and Context Models.* In *PerCom Workshops (4th IEEE Conference on Pervasive Computing and Communications Workshop (PerCom 2006 Workshops), 13-17 March 2006, Pisa, Italy))*, pages 24–29. IEEE Computer Society, 2006.
unter: `http://doi.ieeecomputersociety.org/10.1109/PERCOMW.2006.72` (28. März 2007).

[Blo02] Hans Heinrich Blotevogel. *Geographie.* In *Lexikon der Geographie.*, page 15. Spektrum, Heidelberg, E. Brunotte, H. Gebhardt, M. Meurer, P. Meusberger, J. Nipper (Hrsg.), 2002.

[*CW] *IMS Corpus Workbench (CWB).* Website, Institut für Maschinelle Sprachverarbeitung; Universität Stuttgart.
unter: `http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/` (01. Dez 2006).

[DFM+04] S. Dingare, J. Finkel, C. Manning, M. Nissim, and B. Alex. *Exploring the Boundaries: Gene and Protein Identification in Biomedical Text.* In *Proceedings of the BioCreative Workshop, Granada.*, 2004.
unter: `http://www.stanford.edu/~jrfinkel/papers/BioCreativeTask1A.pdf` (04. Jan 2007).

[Eve02] Stefan Evert. *Corpus Encoding Tutorial: First Steps.* Tutorial, Juni 2002.
unter: `http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/CWBTutorial/cw%b-tutorial.pdf` (01. Dez 2006).

[GM05] Eric Garbin and Inderjeet Mani. *Disambiguating Toponyms in News.* In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical in Natural Language Processing, Proceedings of the Conference 6-8 Oct 2005, Vancouver, British Columbia, Canada.* The Association for Computational Linguistics, 2005.
unter: `http://complingone.georgetown.edu/~linguist/papers/hlt05.pdf` (22. März 2007).

[Lan06] Joel Lang. *Named Entity Recognition - „Knowledge Discovery and Text Mining“.* Seminararbeit, Universität Karlsruhe, 2006.
unter: `http://www.aifb.uni-karlsruhe.de/Lehre/Sommer2006/kdtm/stuff/NER_Ausarb%eitung.pdf` (08. Feb 2007).

[LBC04] Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. *SVM Based Learning System For Information Extraction.* In *Deterministic and Statistical Methods in Machine Learning, First Internal Workshop, Sheffild, UK, Sep 7-10, 2004, Revised Lectures*, pages 319–339. Springer, 2004.
unter: `http://dx.doi.org/10.1007/11559887_19` (9. April 2007).

[MCS+05] Alexander Markowetz, Yen-Yu Chen, Torsten Suel, Xiaohui Long, and Bernhard Seeger. *Design and Implementation of a Geographic Search Engine.* In *Proceedings of the Eight International Workshop on the Web & Databases (WebDB 2005), Baltimore, Maryland, USA, Collocated with ACM SIGMOD/PODS 2005, June 16-17, 2005*, pages 19–24, 2005.
unter: `http://cis.poly.edu/suel/papers/geo.pdf` (22. März 2007).

[Mih03] Jakob Mihály. *Ortsbezug für Web-Inhalte.* Diplomarbeit 2049, Universität Stuttgart, März 2003.
unter: `http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id%=DIP-2049&engl=` (23. März 2007).

[MRS07] C. Manning, P. Raghavan, and H. Schütze. Introduction to Information Retrieval. Cambridge University Press – DRAFT, 2007.
DRAFT is Web publicated at `http:\\www.informationretrieval.org` (12. März 2007).

[Neu04] Günther Neumann. *Informationsextraktion.* In *Computerlinguistik und Sprachtechnologie - Eine Einführung (2. überarbeitete und erweiterte Auflage)*, pages 502–510. Spektrum Akademischer Verlag (ELSEVIER), Klabunde (et al.), 2004.

[*OJ] *OpenJUMP.* Website.
unter: `http://openjump.org/wiki/show/HomePage` (28. Feb 2007).

[*PL] OpenJUMP – How to make your own plugin. Website.
unter: `http://openjump.org/wiki/show/How+to+use+and+make+own+Plugins` (02. Apr 2007).

[RBB03] K. Rothermel, M. Bauer, and C. Becker. *Digitale Weltmodelle - Grundlage kontextbezogener Systeme.* In *Friedemann Mattern (ed.): Total vernetzt - Szenarien einer informatisierten Welt*, pages 123–141. Xpert.press, Berlin, Heiderlberg, New York: Springer-Verlag, Universität Stuttgart : Sonderforschungsbereich SFB 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme), Mai 2003.

[Rot02] Jeanette Roth. *Der Stand der Kunst in der Eigennamen-Erkennung - Mit einem Fokus auf Produktnamen-Erkennung.* Lizentiatsarbeit, Universität Zürich, Dezember 2002.
unter: `http://scholar.google.com/url?sa=U&q=http://www.ifi.unizh.ch/cl/study/l%izarbeiten/lizjeannetteroth.pdf` (10. Jan 2007).

[S¨01] Sütö Mihály. *Ortsbasierter Web-Zugriff.* Studienarbeit 1834, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, Anwendersoftware, April 2001.
unter: `http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id%=STUD-1834&engl=` (23. März 2007).

[*SV] *SVMlight - Support Vector Maschine - von Thorsten Joachims.* Website, Cornell University, Department of Computer Science.
unter: `http://svmlight.joachims.org/` (15. März 2007).

[*W1] *EI - Wikipedia.* Website.
unter: `http://en.wikipedia.org/wiki/Named_entity_recognition` (19. Feb 2007).

[*W2] *Entity - Wikipedia.* Website.
unter: `http://de.wikipedia.org/wiki/Entit\%C3\%A4t_(Informatik)` (11. Jan 2007).

[*W3] *JUMP GIS - Wikipedia.* Website.
unter: `http://en.wikipedia.org/wiki/JUMP_GIS#Features` (28. Feb 2007).

[*W4] *SVM - Wikipedia.* Website.
unter: `http://de.wikipedia.org/wiki/Support-Vector-Machine` (01. März 2007).

[*WG] *World Gazetteer.* Website. unter: `\\http://bevoelkerungsstatistik.de/wg.php?x=&lng=en&dat=32&srt=npan&col%=hdq&men=stdl&lng=de` (22. Okt 2006).

[Zhu06] Xiaojin Zhu. *Semi-Supervised Learning Literature Survey.* Literature Survey 1530, Computer Sciences, University of Wisconsin-Madison, Dezember 2006.
unter: `http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf` (12. Feb 2007).

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift:

Stuttgart, 17. April 2007