Universität Zürich[UZH]

Institut für Computerlinguistik

Seminararbeit im Seminar
'Sprachtechnologie für wenig beachtete Sprachen'
Frühjahrssemester 2012

Part-of-Speech Tagging für Schweizerdeutsch

Betreuer:
Prof. Dr. Martin Volk
lic.phil. Annette Rios

Vorgelegt von:
Noëmi Aepli & Nora Hollenstein
Matrikelnummern 10-719-250 & 09-184-755

27. Mai 2012

Zusammenfassung

Die meiste Arbeit für automatische Sprachverarbeitung orientiert sich an geschriebenen, standardisierten Sprachen. Im Rahmen eines Seminars, das sich mit Sprachtechnologie für wenig beachtete Sprachen beschäftigt, möchten wir uns mit der automatischen Verarbeitung des Schweizerdeutschen, also hauptsächlich gesprochener, nicht standardisierter Sprache befassen. Das Schweizerdeutsche ist gut dokumentiert. Es stellt eine der lebendigsten Dialektgebiete Europas dar, was soziale Akzeptanz und Abdeckung in den Medien betrifft. Trotzdem sind nur sehr wenige geschriebene Ressourcen vorhanden, und diese halten sich an keine orthographischen Regeln.
In der Dialektforschung ist Schweizerdeutsch seit Anfang des 20. Jahrhunderts ein Thema und es existieren einige Projekte, die sich auch mit der automatischen Verarbeitung des Schweizerdeutschen befassen. Das PoS-Tagging ist die Basisstufe für jegliche weitere automatische Verarbeitung von Geschriebenem und deshalb ein grundlegendes Verfahren. Automatische Verarbeitung von schweizerdeutschen Texten könnte, ausser in der Dialektforschung, auch dann interessant werden, wenn auf Internetressourcen zugegriffen wird, da vor allem in Chats und Foren immer mehr Schweizerdeutsch geschrieben wird.

Unser Ziel ist es, ein PoS-Tagset für das Schweizerdeutsche zu entwerfen und damit einen PoS-Tagger zu trainieren und evaluieren. Wir möchten herausfinden, inwiefern das Tagset für Schweizerdeutsch von demjenigen des Deutschen abweicht und ob es möglich ist, einen Tagger so zu trainieren, dass das Resultat für weitere sprachtechnologische Anwendungen weiterverwendet werden kann.

Inhaltsverzeichnis

1 Einführung

1.1 Struktur der Arbeit

Bevor wir unser Vorgehen und die Resultate erläutern, werden wir zunächst auf das Schweizerdeutsche eingehen und auch Unterschiede zum Standarddeutschen sowie zwischen den verschiedenen Dialekten aufzeigen. Des Weiteren werden wir auch auf die Forschung eingehen und einige Projekte erläutern, welche sich mit Schweizerdeutsch beschäftigt haben. In einem nächsten Kapitel werden wir unser Vorgehen erklären, die verwendeten Ressourcen vorstellen und unsere Python-Skripte erläutern. Danach werden wir auf die verwendeten Tagger sowie deren Resultate und Unterschiede eingehen. In einem letzten Punkt vor dem Fazit werden wir einige Projekte vorstellen, welche sich auch mit dem PoS-Tagging für wenig beachtete Sprachen beschäftigt haben.

1.2 Part-of-Speech Tagging

Part-of-speech- (PoS-) oder Wortarten-Tagging ist das Vorgehen, jedem Wort in einem Text die entsprechende Wortart zuzuweisen, indem jedem Wort ein Klassifikationskürzel, ein sogenanntes *Tag*, zugeordnet wird. PoS-Tagging ist eine eigenständige sprachtechnologische Anwendung und wird für verschiedenste Zwecke wie Lemmatisierung, Spracherkennung und -synthese, Dokumentensuche, Bedeutungsdesambiguierung usw. gebraucht. Zudem ist es ein grundlegendes Tool für die weitere Verarbeitung von natürlicher Sprache wie zum Beispiel beim Parsen oder der maschinellen Übersetzung. Es existieren zwei Arten von Tagging-Verfahren, statistische und regelbasierte.
Für unser Projekt haben wir zwei statistische Tagger ausgewählt, da diese einerseits einfach zugänglich und zu bedienen sind und andererseits viel schneller trainiert sind als regelbasierte.

2 Schwyzerdütsch

2.1 Klassifizierung

In *Ethnologue*[1] ist für Schweizerdeutsch ein eigener Eintrag vorhanden.Gemäss *Ethnologue* ist „German, Swiss“ eine „language of Switzerland“ mit dem ISO-639-3-Code[2] *gsw*. Gemäss Zählungen von Ethnologue im Jahr 2000 wird diese Sprache von 4’640’000 Menschen in der Schweiz gesprochen.
Schweizerdeutsch wird linguistisch als indoeuropäische Sprache klassifiziert. Dieser Ast führt weiter über Germanisch, Westgermanisch, Deutsch, Oberdeutsch, Alemannisch bis schliesslich zum Schweizerdeutsch. In der Sprachwissenschaft gibt es zwischen den schweizerdeutschen Dialekten und den übrigen alemannischen Dialekten keine strikten Sprachgrenzen, es wird eher von einem Dialektkontinuum gesprochen. Die alemannischen Dia-

[1] Ethnologue ist ein Sammelwerk, das seit 1951 besteht und zum Ziel hat, alle Sprachen zu erfassen und klassifizieren (siehe: http://www.ethnologue.com/)
[2] ISO 693 ist eine internationale Norm, die Kennungen von Namen von Sprachen definiert.

lekte sind unterteilt in Niederalemannisch, Hochalemannisch und Höchstalemannisch. Dabei wird zum Beispiel der Baslerdialekt zu den Niederalemannischen gezählt, der Aargauer-, Berner-, Ostschweizer-, und Zürichdialekt zu Hochalemannisch und der Walliserdialekt zu Höchstalemannisch. Ethnologue führt folgende Dialekte auf: „Bern (Bärndütsch), Zürich, Luzern, Basel, Obwalden, Appenzell, St. Gallen, Graubünden-Grisons (Valserisch), Wallis“. Gemäss Wikipedia besteht aber das deutsch-alemannische Dialektkontinuum in der Schweiz aus Hunderten von Deutschschweizer Mundarten.
Anders als zwischen den verschiedenen Dialekten herrscht eine klare Trennung zwischen Schweizerdeutsch und Hochdeutsch. Wenn es um die Dialekte der Deutschschweiz geht, trifft man auch auf den Begriff der Diglossie.[3] Hochdeutsch wird in der Schweiz beinahe ausschliesslich im schriftlichen Kontext gebraucht, während Schweizerdeutsch in gesprochener Sprache gebraucht wird. In Schulen wird in Standardsprache unterrichtet. Im Gegensatz zu anderen Sprachen ist es normal, auch in formellen Situationen, zum Beispiel auf Ämtern, Dialekt zu sprechen. Ebenso ist Schweizerdeutsch im Fernsehen und Radio gut vertreten. In jüngerer Zeit wird auch in der schriftlichen Kommunikation immer mehr Dialekt gebraucht, dies aber vor allem im privaten Kontext wie in E-Mails oder SMS. Andere schriftliche, schweizerdeutsche Ressourcen existieren zwar nicht in grossen Mengen, aber es gibt sie. Zum Beispiel in der Alemannischen Wikipedia, wo 11509 Artikel in Badisch, Elsassisch, Schwäbisch und auch in Schwyzerdütsch existieren.[4] Es existieren auch einige gedruckte Werke wie Schweizerdeutsche Wörterbücher, dialektspezifische Grammatiken oder auch sonstige Literatur. Seit 1984 gibt es sogar die Bibel auf Schweizerdeutsch. Üblicherweise werden die Dialekte gemäss administrativen oder topographischen Kriterien klassifiziert, welche aber nicht immer der Realität entsprechen. In diesem Dialektkontinuum haben einige Phänomene klarere Grenzen als andere.

2.2 Unterschiede zum Standarddeutsch und zwischen den Dialekten

Das Schweizerdeutsche unterscheidet sich in vielen Aspekten wie zum Beispiel in Phonetik, Wortschatz, Morphologie und Syntax vom Hochdeutschen. Es kann aber nicht nur von Unterschieden zwischen Standarddeutsch und Schweizerdeutsch die Rede sein, denn es existieren zum Teil auch signifikante Unterschiede zwischen den verschiedenen Dialekten. [Scherrer, 2011b] unterscheidet bei den syntaktischen Unterschieden zwischen Schweizer- und Standarddeutsch hauptsächlich zwei Arten, nämlich die, welche für den ganzen schweizerdeutschen Raum gelten und von welchen zum Teil sogar Varianten in gesprochenem Standarddeutsch gefunden werden und diejenigen, die nur in bestimmten Dialekten oder Dialektgruppen und nicht ausserhalb der alemannischen Dialekte auftreten.
Zu den ersteren zählen zum Beispiel Phänomene der Zeitformen. Im Schweizerdeutschen existieren weder das Präteritum noch das Plusquamperfekt. Diese werden durch

[3] „Form der Zweisprachigkeit, bei der die eine Sprachform die Standardsprache darstellt, während die andere im täglichen Gebrauch, in informellen Texten verwendet wird.“ (Definition aus: http://www.duden.de/rechtschreibung/Diglossie, Zugriff: 22. Mai 2012, 15:23)
[4] Stand 1. Mai 2012

das Perfekt, beziehungsweise eine Verdoppelung des Perfekts ausgedrückt. Auch die vier Kasus des Standarddeutschen werden in den Dialekten nicht übernommen. So wird im Schweizerdeutschen grundsätzlich (mit wenigen Ausnahmen im Walliserdeutsch) der Genitiv nicht gebraucht. Statt dessen wird ein Präpositionalgefüge oder ein possessiver Dativ verwendet. Die Formen von Nominativ und Akkusativ unterscheiden sich nur bei den Personalpronomen, womit einzig der Dativ noch durch eigenständige Artikel und Adjektiv- sowie Substantivendungen gekennzeichnet wird.
Phänomene, die in den verschiedenen Dialekten unterschiedlich behandelt werden, sind zahlreicher. So wird zum Beispiel die Reihenfolge der Verben im Schweizerdeutschen oft umgekehrt, wenn mehrere Verben in einem Satz vorkommen. Die daraus resultierende Reihenfolge variiert aber je nach Dialekt. Um den Finalsatz mit *um ... zu* auszudrücken gibt es die Konstruktion *für ... z* in Dialekten der Westschweiz, die Konstruktion *zum* oder sogar *zum ... z* eher in den Dialekten Richtung Ostschweiz.
Um einige Beispiele aufzuzeigen, haben wir einen standarddeutschen Satz konstruiert und von einigen SchweizerInnen in ihren Dialekt übersetzen lassen. Der Standarddeutsche Satz lautet:

*Sie **liess** ihn gehen, weil er nicht genug Geld **hatte**, **um** ein Billet **zu** lösen.*

Wir erhielten folgende Übersetzungen:

AG:	Si **hett** ihn **gah la**, will er ned gnueg Gäld **gha het**, **zum** es Billet **z**löse.
AI:	Si **hät** ihn **go loh**, will er nöd gnueg Geld **gha hät**, **zum** e Billet löse.
AR:	Si **het** en **springe loh**, well e nüd gnueg Göld **ka het**, **zum** e Billet löse.
BS1:	Sie **hed** ihn **lo go**, will er nid gnueg Geld **gha hed**, **zum** es Billet **z** löse.
BS2:	Sie **het** en **lo go**, wil är nit gnueg Gäld **ka het**, **zum** e Bilett **z**'löse.
BE:	Si **het** ne **la ga**, wüu er ne gnue Gäud **het gha**, **für** es Billet **z**'löse.
GR:	Schi **het**ne **lah gahn**, will är nid gnuag Gääld **ghan het**, **zum** äs Bilet **z**chaufä.
LU1:	Si **hed** ihn **lohgoh**, wil är ned gnueg Gäld **gha hed**, **zom** es Billet **z**'löse.
LU2:	Si **hed** ne **lo go**, wel är ned gnueg Gäld **gha hed**, **zom** nes Billet löse.
LU3:	Sie **het** ihn **lo goh**, wöu är ned gnueg gäud **gha het**, **zum** es billet löse.
OW1:	Si **hed** nä **la ga**, wil är nid gnuäg Gäud **gha hed**, **zum** näs Billeet leesä.
OW2:	Si **hed** nä **la gah**, will är nid gnuäg Gäld **gha hed**, **zum** äs Bilet **z**'chaifä.
SG1:	Si **hät**en **gooloo**, weller nöd gnuäg Gäld **kaa hät**, **zum** ä Billet **z**'lösä.
SG2:	Si **hät** en **go loh**, will er nöd gnueg Gelt **kah hät**, **zum** a billet kaufe.
TG:	Si **hät** en **go loh**, wil er nöd gnueg Gäld **gha hät**, **zum** sich ä Billet **z**löse.
VS:	Schi **het** nu **lah gah**, wiler nid gnüäg Gäld **het ka**, **fer** es Billet **z**lesu.
ZH1:	Si **hät** ihn **gah lah**, wil er nöd gnueg Gäld **gha hät**, **zum** es Billet löse.
ZH2:	Si **hät** ihn **gah lah**, wil er nöd gnueg Geld **ka het**, **zum** es Billet löse.
ZH3:	Si **hätt** ihn **gah lah**, well er nöd gnueg Geld **gha hätt**, **zum** es Billet **z**'chaufe.

In diesem Satz sind einerseits die Präteritumsformen *liess* und *hatte* vorhanden, die über alle Dialekte hinweg ins Perfekt übersetzt wurden: *hat ... (gehen) lassen* und *hat gehabt.*

Die Reihenfolge der Verben in dieser Perfektkonstruktion unterscheidet sich aber je nach Dialekt. Andererseits hat es in diesem Satz die Finalsatz-Konstruktion *um ... zu*. Aufgefallen ist uns, dass sich einige von den Strukturen des Standarddeutschen Satzes haben beeinflussen lassen. So haben einige am Anfang die *um ... zu*-Konstruktion genau so ins Schweizerdeutsche übernommen. Die Frage ob sie sicher seien, dass sie das Wort *um* wirklich so übernehmen würden, haben die meisten verneint und in *zum* oder Ähnliches korrigiert.
Des Weiteren war interessant, dass wir niemals zweimal den gleichen Wortlaut bekommen haben, auch nicht, wenn wir zwei Sprecher des gleichen Dialektes gefragt haben und nicht einmal wenn wir zwei Schwestern um ihre Übersetzung gebeten haben. Zum Beispiel das *weil*, das in den drei Luzernerdialekt-Varianten mit *wil, wel* und *wöu* übersetzt wurde oder das *hatte*, das sich in den drei Varianten des Zürcherdialekts in die drei Übersetzungen *gha hät, ka het* und *gha hätt* transformiert hat. Auch die verschiedenen Schreibvarianten des, in manchen Dialekten vorkommenden, Überbleibsels von *zu*, welches manchmal an das *lösen* geklebt (AG, GR, TG, VS), manchmal mit Apostroph abgegrenzt wird (BS2, BE, LU1, OW2, SG1, ZH3), und einmal als eigenständiges Wort geschrieben wird (BS1), ist ein sehr inkonsequent geschriebenes Phänomen. Doch es betrifft nicht nur das *z*, sondern auch ander Wörter, welche manchmal fusioniert wurden wie zum Beispiel *hetne* (GR) oder *gooloo* (SG1).
Was in diesen Beispielen auch schön zum Vorschein kommt, ist, dass sich einige eher am Standarddeutschen orientieren, während andere wirklich genau so schreiben, wie sie es sagen. Vergleiche: *Geld* vs. *Gäld*, *gnueg* vs. *gnuäg*, *löse* vs. *lösä*, etc. Die ersten Varianten existieren nicht in gesprochenem Schweizerdeutsch und doch werden sie geschrieben, während die zweiten genau den ausgesprochenen Laut wiedergeben.
Dies sind alles Konsequenzen des Dialektkontinuums und der nicht vorhandenen Schreibregeln im Schweizerdeutschen.

2.3 Forschung

In der dialektologischen Forschung ist Schweizerdeutsch seit Anfang des 20. Jahrhunderts ein Thema. Ein grosser Beitrag leistet der *Sprachatlas der deutschen Schweiz* (SDS)[5], ein linguistischer Atlas, der auf Karten die Vielfalt des Schweizerischen Wortschatzes sowie phonetische, morphologische und lexikalische Unterschiede der verschiedenen Dialekte aufzeigt. In diesem Projekt wurden keine syntaktischen Daten gesammelt, was allerdings im Projekt *Syntaktischer Atlas der deutschen Schweiz* (SADS)[6] das Ziel ist. Dieses untersucht die Verteilung von verschiedenen syntaktischen Konstruktionen im Gebiet der deutschsprachigen Schweiz. Die Resultate sollen in einem Atlasband veröffentlicht werden.
Des Weiteren haben wir vor allem drei wissenschaftliche Publikationen gelesen, welche konkret von der maschinellen Verarbeitung der Schweizer Dialekte handeln. In diesen

[5] http://www.ds.uzh.ch/ksds/
[6] http://www.ds.uzh.ch/dialektsyntax/

Projekten wird auch auf die Ressourcen oben genannter Atlanten zurückgegriffen. In den folgenden Abschnitten gehen wir kurz auf diese Projekte ein.

2.3.1 Morphologiegenerierung für schweizerdeutsche Dialekte

Im Paper *Morphology generation for Swiss German dialexts* [Scherrer, 2011a] wird ein Morphologiegenerierugssystem vorgestellt, das mit kontinuierlichen Sprachvariationen, wie sie im Schweizerdeutschen auftreten, umgehen kann. Das System soll flektierte dialektale Varianten eines standarddeutschen Wortes generieren.
Als Eingabe dient dem System der Wortstamm eines standarddeutsches Wortes inklusive einigen Features. Als Resultat gibt es alle flektierten, in der Schweiz geläufigen dialektalen Formen dieses Wortes aus. Die Features sind PoS-Tag, morphologische Merkmale wie Numerus, Genus sowie lexikalische Informationen wie die Flexionsklasse. Die Generierung wird in zwei Schritten durchgeführt. Zuerst wird anhand von phonetischen und lexikalischen Transformationen auf dem standarddeutschen Wortstamm der dialektale Wortstamm bestimmt, welcher im zweiten Schritt gemäss den Eingabe-Features mit Affixen ergänzt wird. Die meisten Regeln, die für diesen Prozess benötigt werden, wurden von den Karten aus dem SDS extrahiert. Wegen vielen Unregelmässigkeiten in den Schweizer Dialekten mussten zum Beispiel für die phonetischen Transformationen „whitelists“ und „blacklists“ verwendet werden, für diejenigen Wörter, bei welchen die jeweilige Regel angewandt oder eben nicht angewandt werden soll.
Für unser Projekt hat uns Yves Scherrer eine mit diesem Morphologiegenerierungssystem erstellte Liste Schweizerdeutscher Wörter mit dem ihm entsprechenden standarddeutschen Wort und PoS-Tag zur Verfügung gestellt. Wie wir sie verwendet haben wird in einem nachfolgenden Abschnitt beschrieben.

2.3.2 Maschinelle Sprachverarbeitung für schweizerdeutsche Dialekte

Im Paper *Natural Language Processing for the Swiss German dialect area*
[Scherrer and Rambow, 2010] werden Modelle für ein maschinelles Übersetzungssystem in einen der Schweizer Dialekte, für Dialektidentifikation und multi-Dialekt-Parsing vorgestellt. Das System enthält phonetische, lexikalische und morpholgoische Regeln, von welchen einige auf alle Dialekte angewendet werden können, andere nur auf einzelne. Die verschiedenen Varianten der einzelnen Regeln sind mit einer Wahrscheinlichkeits-Karte verlinkt, die anzeigt, wo diese Varianten wirklich gebraucht werden. Diese Karten wurden wiederum aus dem SDS extrahiert. Solche Regeln erlauben es, standarddeutsche Wörter in diejenigen eines spezifischen Schweizer Dialekts umzuwandeln.

Maschinelle Übersetzung: Für die maschinelle Übersetzung wird zunächst der standarddeutsche Satz syntaktisch und morphologisch analysiert. Dabei wird jedes Wort lemmatisiert, PoS-getaggt und mit morphologischen Features annotiert. Im nächsten Schritt wird jedes Wort übersetzt, wobei lexikalische oder phonetische Regeln auf die Grundform des standarddeutschen Wortes angewendet werden und dabei die schweizerdeutsche Grundform entsteht. An dieser Stelle kommt das oben beschriebene Mor-

phologiegenerierungssystem ins Spiel und kreiert aus der schweizerdeutschen Grundform und den morphologischen Features die flektierte dialektale Form. Der Zieldialekt wird vom User bestimmt, indem er Koordinaten eines Punktes auf der Karte aussucht. Dieses Übersetzungsmodell berücksichtigt keine morphosyntaktischen Anpassungen oder Wortumstellungen. Ein solcher wort-zu-wort-Übersetzungsansatz ist in vielen Fällen hinreichend, aber es gibt auch Situationen, in welchen andere Konstruktionen gebraucht würden. Da es zum Beispiel kein Präteritum gibt, müssen alle diese Sätze in Perfekt umgewandelt werden und Genitive, die zu verschiedenen syntaktischen Strukturen werden um Possessiv-Konstruktionen anzuzeigen, müssen umgestellt werden. Für Fälle wie diese müssen zusätzlich syntaktische Regeln eingebaut werden.
Ein wichtiger Punkt für die maschinelle Übersetzung ist das Vorhanden sein von parallelen Korpora. Diese sind aber für Schweizer- und Standarddeutsch eigentlich nicht zu finden, denn der Bedarf nach Übersetzungen zwischen diesen Sprachen ist, wie üblich bei Diglossie, nicht vorhanden.

Dialekterkennung: Spracherkennungssysteme funktionieren grundsätzlich gut, indem sie mit Buchstaben- oder N-Gramm-Verteilungen arbeiten. Dies funktioniert aber für Dialekte nicht, da diese mit sehr ähnlichen Phonemen und Graphemen aufgebaut sind. Dazu kommt, dass Trainings-Daten für alle Dialekte vorhanden sein müssten. Als Alternative wird in diesem Paper ein Ansatz beschrieben, bei welchem ganze Wörter in Texten identifiziert werden und von diesen herausgefunden wird, in welchen Regionen sie in dieser Form vorkommen. Auch für diesen Ansatz kommt das Morphologiegenerierungssystem zum Einsatz. Im Gegensatz zum maschinellen Übersetzungssystem wird hier kein Zieldialekt festgelegt sondern es werden alle möglichen Dialektformen generiert und zusammen mit den dazugehörigen Karten gespeichert. Für jedes schweizerdeutsche Wort wird dann eine Wahrscheinlichkeits-Karte produziert und die Karten von allen Wörtern eines Segments werden verbunden.

Dialektparsing: Das Ziel des multi-dialektalen Parsers ist, einen unannotierten Text syntaktisch zu analysieren. Ein Parser besteht normalerweise aus einer Grammatik und einem Lexikon. Im multi-dialektalen Ansatz müssen die Grammatikregeln und Lexika wiederum mit den ihnen entsprechenden Wahrscheinlichkeits-Karten verlinkt sein. Als Lexika können die gleichen dienen, welche für die Dialekterkennung gebraucht werden. Die Grammatik wird aus einer standarddeutschen Baumbank extrahiert und manuell modifiziert, um auf die Besonderheiten des Schweizerdeutschen zu matchen. Dazu können sie syntaktischen Regeln für die maschinelle Übersetzung als Guideline dienen.

2.3.3 Syntaktische Transformation für schweizerdeutsche Dialekte

Im Paper *Syntactic transformations for Swiss German dialects*[Scherrer, 2011b] werden manuell Regeln entwickelt, die standarddeutsche Sätze syntaktisch korrekt in schweizerdeutsche Sätze eines bestimmten Dialekts überführen. Als Eingabe dienen morphosyntaktisch annotierte standarddeutsche Daten.

3 Vorgehen

3.1 Tagset

Als Grundlage für die Erstellung eines Tagsets für das Schweizerdeutsche haben wir das Stuttgart-Tübingen-TagSet (STTS) [Schiller et al., 1999] für die deutsche Sprache gewählt. Dieses Tagset umfasst insgesamt 54 Tags, aufgeteilt in elf Hauptwortarten (Nomina, Verben, Artikel, Adjektive, Pronomina, Kardinalzahlen, Adverbien, Konjunktionen, Adpositionen, Interjektionen und Partikeln), welche wiederum unterschiedlich stark unterklassifiziert sind. Es gibt zum Beispiel 12 verschiedene Tags für Verben, welche zwischen Hilfs-, Modal- und Vollverben sowie Infinitiv, Partizip Perfekt, Imperativ und finitem Verb unterscheiden und 14 für Pronomen, welche als Possessiv, Demonstrativ, Indefinit, Interrogativ und Relativ, sowie als attribuierend oder substituierend beschrieben werden.

Dieses Tagset kann nicht ohne Änderungen für das Schweizerdeutsche übernommen werden. So haben wir es während der Erstellung des Trainingskorpus bei Bedarf angepasst. Dies resultierte nach der manuellen Annotation aller Trainingsdaten in 25 neuen Wortartentags. Das daraus entstandene Tagset kann aber auf keinen Fall als ein vollständiges PoS-Tagset für Schweizerdeutsch angesehen werden. Denn die Schweizer haben ein Flair, wenn es darum geht, Wörter zusammenzuschreiben. Zwar werden im Deutschen auch gerne Wörter zusammen geschrieben, was in der fleissigen Produktion neuer Komposita sichtbar wird, die für jede Art von automatischer Verarbeitung des Deutschen einen Stolperstein darstellen. Im Deutschen ist es aber, zumindest manuell, kein Problem, das Wort einer Wortart zuzuordnen, denn die Wortart, wie auch Flexionsklasse und wenn nötig das Geschlecht, werden vom letzten Glied auf das zusammengesetzte Wort übertragen. Das Phänomen im Schweizerdeutschen gleicht aber weniger dem der Komposita, sondern eher demjenigen von Klitika; „Ein Klitikon (...) ist ein schwach- oder unbetontes Wort, das sich an ein benachbartes (betontes) Wort ‚anlehnt' (...). Klitika können nicht isoliert und unabhängig auftreten. Sie sind keine freien, unabhängigen Wörter und nehmen eine Sonderstellung zwischen freien Wörtern und Affixen ein."[7] mit dem Unterschied, dass die „Anhängsel" im Schweizerdeutschen sehr wohl auch unabhängig auftreten können. Meistens werden sie jedoch zusammengeklebt, was die Folge hat, dass diese Zusammenschreibungen nicht einer einzigen Wortart zugewiesen werden können und unser Tagset um zusammengesetzte Wortarten ergänzt werden musste. Es handelt sich meistens um Zusammensetzungen von zwei Wörtern, also zwei Wortarten. In unserem Trainingsset haben wir zwei Fälle entdeckt, wo sogar drei Wörter zu einem fusioniert haben. Im STTS selbst existiert schon eine zusammengesetzte Wortart, nämlich die Präposition (APPR) mit Artikel (ART), APPRART. Dieses Tag haben wir als Vorbild für unsere Zusammensetzungen genommen und unsere Benamsung für neue Tags dieser angepasst. Die daraus resultierten Wortarten sind in Tabelle 1 mit der Beschreibung und einem Beispiel aufgeführt. Zusätzlich haben wir in dieser Tabelle die Häufigkeiten der neuen

[7] Definition aus: http://de.wikipedia.org/w/index.php?title=Klitikon&oldid=102607455, Zugriff: 22. Mai 2012, 15:15

Wortart in unserem Trainings- und Testset gezählt und in einer separaten Spalte aufgeführt. Das erste in der Tabelle ist speziell, wir werden gleich darauf eingehen. Die nächsten drei setzen sich je aus drei Tags zusammen und sind deshalb oben aufgeführt. Alle anderen setzen sich aus zwei Tags zusammen und sind alpahbetisch geordnet.
Auf ein neues Tag möchten wir speziell eingehen. Es handelt sich um das erste Tag in der Tabelle, das wir INFPART, was für Infintivpartikel steht, getauft haben und als „cho“, „go“, wie auch „afa“ in schweizerdeutschen Texten auftaucht. Es ist nämlich das einzige Tag, das im Deutschen nicht existiert, da es diese Konstruktion nicht gibt, und das einzige, das keine Zusammensetzung von mehreren Tags ist. Dieses Phänomen wurde schon etliche Male untersucht und es gibt einiges darüber zu lesen. Der Entschluss, unser Tagset um ein neues Tag INFPART für*Infinitivpartikel* zu erweitern, stützt sich auf [Glaser, 2003], wo dieses Phänomen kurz beschrieben und so genannt wird.

Tabelle 1: Neue Tags als Erweiterung des STTS

Tag	**Beispiel**	**Übersetzung**	**Häufigkeit**
INFPART	cho, go, afa	-	5
KOUSPPER2	bissne	bis sie ihn	1
VAFINPPER2	heisisech	haben sie sich	1
VVFINPISPPER	bruchtmese	braucht man sie	2
ADJANN	erschtemol	erste Mal	1
ADV2	ono, widereinisch	auch noch, wieder einmal	5
ADVPAV	dadruus, dodrbi	da daraus, da dabei	2
ADVPIS	sovill	so viel	1
ADVVAFIN	gärnha	gern haben	1
APPRPIS	vorallem, underanderem	vor allem, unter anderem	7
KOUSPIS	dasme, wemme	dass man, wenn mann	3
KOUSPPER	dases, indems, siter	dass es, indem sie, seit er	15
PIATNN	vilmol	viel mal	1
PRELSPIS	wome	wo man	1
PRELSPPER	wos, wone, woner	wo es, wo ihn, wo er	12
PTKNEGADV	nüme, nonig	nicht mehr, noch nicht	10
PTKZUVAINF	zhaa	zu haben	1
PWAVPPER	wienes, wiener	wie es, wie er	4
VAFINPPER	hets, ischs, sisi	hat es, ist es, sind sie	31
VAFINPRF	heisech	haben sich	1
VMFINPIS	chamer	kann man	3
VMFINPPER	söler, chaner, wotsech	soll er, kann er, will sich	4
VVFINPIS	bruchtme, machtme	braucht man, macht man	5
VVFINPPER	gits, nimmts, heissts	gibt es, nimmt es, heisst es	27

Wie wir in der Einleitung beschrieben haben, war unser Ziel am Anfang, ein PoS-Tagset zu entwerfen und damit einen Tagger zu trainieren. Während des manuellen Taggings haben wir jedoch feststellen müssen, dass wir am Schluss dieses Projekts kein fertiges Tagset werden präsentieren können. Je länger wir manuell getaggt haben, desto mehr neue Wortarten haben wir entdeckt und hinzugefügt. Nach den ersten 11'000 Wörtern kamen 18 neue Wortarten hinzu, nach den nächsten 11'000 Wörtern weitere neun Tags. Natürlich würden immer weniger neu hinzugefügt werden müssen und trotzdem ist es sehr unwahrscheinlich, dass man es mit diesem Ansatz jemals als vollständig betrachten könnte, denn die Zusammensetzungen sind so inkonsistent und beliebig, dass niemals alle möglichen Varianten abgedeckt werden könnten. So drängt sich die Frage auf, ob es überhaupt sinnvoll ist, ein Tagset mit so vielen Tags zu benutzen. Ein Tagger muss feinere Unterscheidungen treffen können, je detailreicher ein Tagset ist. An dieser Stelle sei auch auf Punkt 5.2 „Bhasa Indonesia"verwiesen, wo in einem Projekt die Resultate eines grösseren und eines kleineren Tagsets verglichen wurden und das kleine Tagset besser abgeschnitten hat. Auch für die weitere Verarbeitung der PoS-getaggten Texte ist es eher Hindernis als wertvolle Information, wenn das Tagset aus Hunderten von Tags besteht. Man müsste also einen Weg finden, die Grösse des Tagsets zu beschränken, um mit einem fixierten, vollständigen Tagset arbeiten zu können.
Für das Schweizerdeutsche würde dies bedeuten, dass die zusammengesetzten Wörter aufgesplittet werden müssten, um sinnvolle Wortartenzuweisungen zu machen, was wiederum ein Problem der Normalisierung darstellt.

3.2 Korpus

Um die Trainings- und Testsets für unser Experiment zusammenzustellen, haben wir aus der Alemannischen Wikipedia 453 schweizerdeutsche Artikel extrahiert. Die Alemannische Wikipedia ist unterteilt in Schwyzerdütsch, Badisch, Elsassisch und Schwäbisch wobei für Schweizerdeutsch Artikel in verschiedenen Dialekten von „Aargauerdüütsch" bis „Züritüütsch" existieren. Es sind aber bei weitem nicht für alle Dialekte Wikipediaartikel vorhanden. Von den existierenden Dialekten haben wir die fünf ausgewählt, für welche am meisten Artikel geschrieben wurden. Diese umfassen Aargauer-, Basler-, Berner-, Ostschweizer-, Zürichdeutsch. Zu den Ostschweizer-Dialekten zählen in Wikipedia die Dialekte von Schaffhausen, Thurgau, St. Gallen und Appenzell.
Dieses Korpus hat einige unangenehme Eigenschaften. Zum einen ist der Dialekt eines Artikels bestimmt durch den Dialekt des Autors. Dies ist eine vage Klassifizierung, denn wie schon erwähnt, können keine klare Grenzen zwischen den verschiedenen Dialekten gezogen werden. Das Ganze wird erschwert, wenn es Autoren und Autorinnen gibt, die eigentlich einen anderen Dialekt haben als jenen, in welchem sie schreiben, wie zum Beispiel Benutzer „Skywalker", der Artikel für den Bernerdialekt schreibt: „Zwar hani e chli e angere Dialäkt, aber ig gloube dr chöit mi scho vrstah..." [8].
Dazu kommt, dass jeder Autor seine eigenen Schreibregeln hat, beziehungsweise eben

[8]Zitat aus: http://als.wikipedia.org/wiki/Benutzer:Ilya_Skywalker, Zugriff: 22. Mai 2012, 15:22

nicht hat, denn sogar innerhalb eines gleichen Artikels werden verschiedene Schreibvarianten derselben Wörter verwendet. Zum Beispiel werden in einem Artikel, der in Ostschweizerdeutsch verfasst wurde, drei verschiedene Varianten von *von den* benutzt: *vo de*, *vode* und sogar *fode*. In dieser Hinsicht fehlt eindeutig eine Standardisierung der schweizerdeutschen Schreibung, was die ganze automatische Verarbeitung des Schweizerdeutschen deutlich vereinfachen würde. Es gab auch schon Versuche, die Orthographie zu vereinheitlichen und Schreibregeln zu definieren, wie zum Beispiel in einem Buch von Eugen Dieth von 1938: „Schwyzertütschi Dialäktschrift". Von diesem Buch gab es 1986 eine Neuauflage, die Christian Schmid-Cadalbert herausgegeben hatte. Viele heute existierende Wörterbücher und Grammatiken fürs Schweizerdeutsche brauchen diese Schreibung. Die Grundregeln sind sehr einfach: „Schryyb wie de schwäzesch!"[9]. Des Weiteren wird unterschieden zwischen langen und kurzen und geschlossenen und offenen Selbstlauten sowie zwischen Lenis und Fortis. Um kurz einen Einblick zu erhalten, wie diese Schreibung aussieht, haben wir den Artikel darüber gelesen: „Dieth-Schryybig"[Die, 2012] aus der Alemannischen Wikipedia, der selbst auch in der Dieth-Schreibung verfasst wurde. Bei der Lektüre wurde uns schnell klar, warum sich diese Schreibung nicht bis in den Alltag durchgesetzt hat. Es gibt sehr viele Sonderzeichen, welche aus unserer Sicht das Lesen und natürlich vor allem das Schreiben erschweren. Dazu kommt, dass es unter all diesen Sonderzeichen sogar eines gibt, für welches nicht einmal ein Unicode-Standard existiert. Eine solche Schreibung hat natürlich heute, im Zeitalter von Emails und SMS, sowieso keine Chance mehr.
Ein weiteres Merkmal ist, dass das Schweizerdeutsch in diesen Artikeln von standarddeutschen Ausdrucksweisen beeinflusst ist. Denn die Quellen, welche benutzt wurden um diese Artikel zu schreiben, sind meist in Hochdeutsch verfasst. Dies beeinflusst die Schreibweise sehr und macht die Ausdrucksweise weniger natürlich, wie wir auch in unserem kleinen Experiment in Punkt 2.2 beschrieben haben. Zum Beispiel haben wir in einem Berndeutschen Text das Wort *merkwürdig* gefunden, was grundsätzlich niemand brauchen würde in spontaner schweizerdeutscher Sprache. Aber es beeinflusst nicht nur die Wortwahl, sondern auch die Satzstruktur und die Grammatikkonstruktionen, wie dieses *um...zu* in unserem Beispielsatz in Punkt 2.2. Und auch [Scherrer and Rambow, 2010] schreibt, dass viele schweizerdeutschen Artikel Übersetzungen von den standarddeutschen Texten sind und sie diese deshalb als kleines paralleles Korpus für ihr maschinelles Übersetzungssystem brauchen konnten.
Trotz diesen vielen Unerwünschtheiten überwogen die positiven Argumente, die Alemannische Wikipedia als Korpus zu benutzen. Diese sind sehr einfach: es gibt einigermassen viel geschriebenen Text, welcher in Dialekte aufgeteilt ist.

3.3 Vorverarbeitung

Zunächst haben wir die Wikipedia-Artikel als html-Dateien heruntergeladen. Von diesen haben wir den untersten Teil mit Informationen über Änderungen inklusive Benutzer-

[9]Zitat aus: http://als.wikipedia.org/w/index.php?title=Dieth-Schreibung&oldid=403830, Zugriff: 22. Mai 2012, 15:22

angaben und Sprachen, in welchen der jeweilige Artikel vorhanden ist, mit Hilfe eines kleinen Python-Skripts abgeschnitten. Mit dem Unix-Programm textutil haben wir das Übriggebliebene in normale Textdateien (*.txt*) konvertiert. Somit lag uns eine ansehbare Menge an Rohdaten, also Text im Rohformat vor, die nun ins richtige Format gebracht und verarbeitet werden musste.
In einem nächsten Schritt mussten diese txt-Dateien tokenisiert, das heisst, in einzelne Token unterteilt werden. Als Token werden üblicherweise Wortformen oder Satzzeichen gezählt. Für diesen Schritt haben wir den Perl-Tokenizer von Frau Dr. Stefanie Dipper von der Ruhr Universität Bochum[10] benutzt. Zunächst mussten wir aber noch einige kleine Änderungen vornehmen, um den Tokenizer mit unseren Eingabedateien zum funktionieren zu bringen. Des Weiteren trennt dieser Tokenizer die Token nur durch Leerzeichen ab. Um die Texte zu vertikalisieren haben wir deshalb in einem zweiten Tokenisierungsschritt alle Leerzeichen durch Zeilenendzeichen ersetzt.

An dieser Stelle möchten wir auf einen weiteren Stolperstein für die automatische Verarbeitung des Schweizerdeutschen ansprechen. Während der ganzen Vorverarbeitung hatten wir, wie für Schweizerdeutsch zu erwarten war, einige Probleme mit den Umlauten. So mussten wir bei jedem Schritt herausfinden, welches Terminal wir benutzen dürfen und in welchem Text-Editor wir das Resultat anschauen und bearbeiten können, damit schliesslich die Encodierung stimmt.

3.4 Vortagging

Unser gesamtes Trainingsset besteht aus 20'000 Token (2000 pro Dialekt), das Testset aus 1000 Token (200 pro Dialekt). Damit wir die Trainingstexte und den Goldstandard für das Testset nicht zu 100% manuell taggen mussten, haben wir die Tokenliste automatisch vorgetaggt. Zu diesem Zweck haben wir ein kleines Python-Skript geschrieben. In diesem Skript haben wir einige kleine Regeln für Satzzeichen und Zahlen geschrieben, da bei diesen eindeutig ist, welches Tag sie bekommen. Einen grossen Teil konnten wir aber mit Hilfe von schon PoS-getaggten Wortlisten zuordnen. Am Anfang hatten wir zwei Wortlisten zur Verfügung. Zum Einen hat uns Yves Scherrer freundlicherweise eine Wortliste zur Verfügung gestellt, in welcher schweizerdeutsche Wörter den entsprechenden STTS-Tags zugeordnet sind. Diese Wortliste wurde, wie im Punkt 2.3.2, beziehungsweise [Scherrer and Rambow, 2010] beschrieben, anhand von Regeln aus deutschen Wörtern generiert. Da es im Schweizerdeutschen aber auch Wörter gibt, die gleich geschrieben werden wie im Standarddeutschen, haben wir eine zweite Wortliste mit deutschen Wörtern zu Hilfe genommen. Diese haben wir aus dem TIGER-Korpus[11], einer Baumbank für das Deutsche extrahiert. In unserem Skript werden diese Wortlisten zeilenweise eingelesen, daraus die Wörter mit den dazugehörigen PoS-Tags extrahiert und gespeichert. In der zu taggenden Datei wird dann jedes Wort in diesen Wortlisten gesucht. Wird es gefunden, wird diesem Wort das gleiche Tag zugewiesen, das in

[10] http://www.linguistics.ruhr-uni-bochum.de/ dipper/tokenizer.html
[11] http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/

der Wortliste gespeichert wurde. Wenn es nicht gefunden wird, bekommt das Wort das „Dummy“-Tag „unknown“.

In Tabelle 2 ist, nach Dialekten aufgeteilt, aufgelistet, wieviel Mal das „Dummy“-Tag vergeben wurde und wieviele Wörter ein Token aus der Schweizerdeutsen Wortliste, beziehungsweise der TIGER Wortliste bekommen haben. Insgesamt umfasst jedes Dialekt-Testset 400 Token, das gesamte 2000 Token.

Tabelle 2: Häufigkeiten „unknown“ vs. Gefundene im gesamten Testset

Dialekt	**unknown**	**Schweizerdeutsch**	**TIGER**
Aarau	101	249	70
Basel	133	206	65
Bern	131	215	74
Ostschweiz	100	238	54
Zuerich	91	211	73
Alle	556	1119	336

Von den Wörtern des gesamten Testsets wurden in der schweizerdeutschen Wortliste 1119 Wörter gefunden, in der deutschen (TIGER) Wortliste 336. 556 Token, knapp 28%, wurden nicht gefunden und erhilten das Tag „unknown“.

Damit wir nicht dauernd die gleichen Fehler korrigieren und immer die gleichen unbekannten Wörter taggen mussten, haben wir in weiteren Schritten auch unsere schon fertig getaggten Listen für das Vortagging gebraucht. Dazu haben wir unsere getaggten Texte, wie im ersten Schritt die zwei Wortlisten, zeilenweise eingelesen und gespeichert. Beim Vortagging des neuen Textes wurde jedes Wort als erstes in dieser manuell korrigierten Wortliste gesucht, bevor die anderen Wortlisten konsultiert wurden. Dadurch wurde das manuelle Vortagging effizienter und weniger zeitaufwändig.

Die Tabelle 3 entspricht der Tabelle 2, ergänzt mit einer Spalte für die Anzahl, wie viele Token in unserem Trainingsset gefunden wurden.

Tabelle 3: Häufigkeiten „unknown“ vs. Gefundene im gesamten Testset mit Trainingsset

Dialekt	**unknown**	**Schweizerdeutsch**	**TIGER**	**Trainingsset**
Aarau	81	59	17	263
Basel	97	44	21	242
Bern	93	44	17	266
Ostschweiz	79	48	13	252
Zuerich	58	55	12	250
Alle	408	250	80	1273

Im Vergleich zum Vortagging ohne Trainingsset wurden 148 Token mehr gefunden und insgesamt noch 408, gut 20%, als „unknown“ getaggt, was 7.6% weniger ist als ohne Trainingsset. Insgesamt wurden im Trainingsset 1273 Token gefunden, was 63.3% aller Token entspricht. In den Schweizerdeutsch- und TIGER-Wortlisten wurden entsprechend weniger Tags nachgeschaut.
Bei den Zahlen von Tabelle 3 ist jedoch zu beachten, dass dies hypothetische Zahlen sind, denn bei diesem Versuch haben wir das ganze manuell korrigierte Trainingsset als dritte Wortliste mitgegeben. Dies war natürlich während des Prozesses noch gar nicht möglich, weil das Skript ja genau dazu diente, diese Texte vorzutaggen.
Mit Hilfe dieses Skripts konnten wir je nach Dialekt und mitgegebenen Wortlisten zwischen 67% und knapp 85% der Trainingstexte automatisch vortaggen. Über dem ganzen Trainingstext sind es 72.4% ohne mitgegebenes Trainingskorpus und 79.7% mit diesem. Diese Tags waren noch sehr fehlerhaft und mussten von uns noch manuell korrigiert und ergänzt werden, um schliesslich als Trainingstexte oder Goldstandard dienen zu können.

3.5 Evaluation

Für die Evaluation haben wir ein Skript geschrieben, das den vom Tagger getaggten Text mit unserem Goldstandard vergleicht. Dazu werden beide Dateien zeilenweise eingelesen und die PoS-Tags miteinander verglichen. Das Skript gibt uns die Genauigkeit (engl. *accuracy*) in Prozent, sowie die Anzahl korrekt getaggter Token aus. Die Genauigkeit entspricht der Anzahl korrekt getaggter Token durch die Anzahl aller Token[Carstensen et al., 2010]. Die nachfolgenden Genauigkeitsangaben in unserer Arbeit entsprechen der Genauigkeit in Prozent.

$$Genauigkeit = \frac{Anzahl\ korrekt\ gettagte\ Token}{Anzahl\ Token}$$

Zudem generiert das Skript eine Ausgabedatei, in welcher die Fehler ihrer Häufigkeit nach geordnet sind, wobei in der ersten Spalte das Goldstandard-Tag steht, in der zweiten

dasjenige, das vom Tagger zugewiesen wurde und in der dritten Spalte die Häufigkeit dieses Fehlers. Als Beispiel hier der Anfang der Ausgabe des Evaluationsskripts für einen getaggten Text:

1712 von 2011 korrekt getaggt
accuracy: 85.13%
('NE', 'NN') 30
('NN', 'NE') 24
('ART', 'APPRART') 16
('ADJA', 'NN') 14
('VVFIN', 'VVPP') 14
('ADJD', 'ADV') 9
('ADV', 'ADJD') 8
('VVPP', 'VVFIN') 5
('ADJD', 'ADJA') 5
('NN', 'ADJA') 4

Mit diesem Skript haben wir nicht nur unsere Experimente mit den beiden Taggern, welche in Punkt 4 aufgeführt sind, evaluiert, sondern auch unser Vortagging-Skript zum Vergleich.

In Tabelle 4 sind die Resultate einerseits aufgeteilt nach Dialekten und andererseits nach den verwendeten Wortlisten. Die schweizerdeutsche und die deutsche Wortliste haben wir beides Mal mitgegeben, in der unteren Reihe zusätzlich noch unser ganzes Trainingsset (20'000 Token). Getestet haben wir einmal über das ganze Testset (2000 Token) und einmal aufgeteilt über jeden Dialekt (400 Token).
Beim Betrachten dieser Resultate ist erstens zu beachten, dass unser Skript höchstens ein Tag pro Wort gespeichert hat und sich nicht zwischen mehreren möglichen entscheiden muss und zweitens bei unbekannten Wörtern das Tag „unknown" vergibt, anstatt das wahrscheinlichste zu raten.

Tabelle 4: Genauigkeit Vortagging über dem gesamten Testset

Dialekt	**ohne Trainingsset**	**mit Trainingsset als 3. Wortliste**
Aarau	56.67%	69.76%
Basel	52.97%	65.35%
Bern	47.38%	67.14%
Ostschweiz	55.87%	69.13%
Zuerich	59.47%	73.87%
Alle	54.35%	68.97%

Wenn wir nur die Wortlisten mitgeben, bewegen sich die Resultate zwischen 47.38% für

Berndeutsch und 59.47% für Zürichdeutsch. Wenn wir zusätzlich das Trainingsset mitgeben, bewegen sie sich zwischen 65.35% für Baseldeutsch und 73.87% für Zürichdeutsch. Das Resultat über das ganze Testset, mit mitgegebenem Trainingsset, beträgt 68.97% Genauigkeit, das heisst, es wurden 1387 von 2011 Token richtig getaggt. Die restlichen 624 Token wurden falsch oder als „unknown“ getaggt.

4 Tagger

4.1 TreeTagger

4.1.1 Architektur

Zuerst verwendeten wir den statistischen, frei verfügbaren *TreeTagger*, der ein Hidden Markov Model (HMM) implementiert, um das PoS-Tagging durchzuführen. Dieser Tagger wurde ursprünglich für Englisch entwickelt und danach für Deutsch angepasst und verbessert. Mittlerweile gibt es die nötigen Parameter auch für andere Sprachen wie zum Beispiel Spanisch, Französisch, usw. ([Schmid, 1995]).
Der TreeTagger operiert über Wahrscheinlichkeiten, dass ein Wort im betrachteten Kontext einer bestimmten Wortart angehört. Das Markov Modell lässt sich durch zwei Zufallsprozesse beschreiben. Der erste wird von einer Markov-Kette definiert, welche aus verborgenen Zuständen und Übergangswahrscheinlichkeiten besteht und der zweite Prozess erzeugt zu jedem Zeitpunkt Aussgangssymbole gemäss einer zustandsabhängigen Wahrscheinlichkeitsverteilung ([Eddy, 2004]). Wenn ein Markov Modell erster Ordnung verwendet wird, werden Bigramme analysiert, ein Markov Modell zweiter Ordnung analysiert Trigramme. Die wahrscheinlichste Tag-Sequenz wird anhand des Viterbi-Algorythmus' berechnet. Dieser Algorithmus der dynamischen Programmierung wird benutzt, um die wahrscheinlichste Sequenz von versteckten Zuständen bei einem gegebenen HMM und einer beobachteten Sequenz von Aussgangssymbolen zu ermitteln.
Der TreeTagger verwendet bei unbekannten Wörtern Entscheidungsbäume (engl. *decision trees*) mit kontextuellen Parameter. Diese Entscheidungsbäume werden verwendet, um die Wortart von Wörtern, die nicht im Trainingsset vorkommen, zu bestimmen und um Ambiguitäten aufzulösen. Der TreeTagger schaut also, welches PoS-Tag das Wort vor dem unbekannten Token hat und berechnet dann die Wahrscheinlichkeit für das unbekannte Tag, zum Beispiel wenn Tag_0 = 'Haus', Tag_{-1} = ADJA und Tag_{-2} = ART, somit ist das gesuchte Tag mit sehr grosser Wahrscheinlichkeit ein normales Nomen (NN) ([Schmid, 1994]). In der Abbilbung 1 kann man sehen, wie solche Entscheidungsbäume angewendet werden.

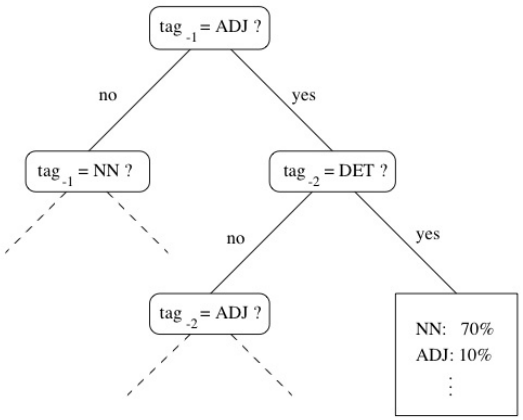

Abbildung 1: Beispiel für einen Entscheidungsbaum des TreeTaggers aus [Schmid, 1994]

4.1.2 Training

Um den TreeTagger zu trainieren, braucht man mehrere vorbearbeitete Dateien: ein Trainingsset, in unserem Fall 20'000 Token mit manuell korrigierten Tags aus schweizerdeutschen Wikipedia-Texten, eine Textdatei mit den möglichen Wortarten für unbekannte Wörter und ein Lexikon. Unser Lexikon besteht aus Yves Scherrers generierten Wortlisten, der deutschen Wörter des TIGER Korpus und je einem Beispiel für die neuen Wortarten aus unserem Tagset. Wähend des Trainings wird eine Parameterdatei generiert, die danach für das PoS-Tagging massgebend ist.

4.1.3 Tagging

Für das eingentliche Tagging haben wir ein Testset mit 2'000 Token und die obengenannte Parameterdatei verwendet. Man könnte dem TreeTagger bei diesem Schritt noch ein zusätzliches Lexikon mitgeben. Dies war uns aufgrund fehlender Resourcen nicht möglich.

Mit der Option *-cl* haben wir zusätzlich die Genauigkeit zwischen einer Konextlänge von 1, 2, 3 oder 4 unterschieden, d.h. zwischen HMM erster, zweiter und dritter Ordung. Kontextlänge 1 bedeutet, dass nur Bigramme, also zwei Token, berücksichtigt werden, Kontextlänge 2 bedeutet, dass Trigramme den statistischen Kontext bilden, usw. Standardmässig funktioniert der TreeTagger mit der Kontextlänge 2. Eine grössere Kontextlänge kann für kleinere Trainingskorpora und/oder grosse Testsets nützlich sein, zeigte bei uns jedoch nur minimale Unterschiede.

4.1.4 Evaluation

Wir haben über das ganze Trainings- und Testset, über die einzelnen Testsets sowie auf Dialekte aufgeteilt evaluiert. Bei der Evaluation über das gesamte Trainings- und Testset haben wir zusätzlich die verfügbaren Optionen getestet.

Resultat über unsere gesamte Korpora mit 20'000 Token im Trainingsset (4'000 pro Dialekt) und 2'000 Token im Testset (400 pro Dialekt):

68.97%

Tabelle 5 zeigt einige allgemeine Resultate des PoS-Tagging. Es wird angegeben welches Trainings- und Testset wir verwendet haben, sowie die Genauigkeit und die Anzahl korrekt getaggter Token unserer Versuche.

Tabelle 5: Evaluation auf Testsets aufgeteilt

	Testset 1	**Testset 2**
Trainingsset 1	66.4%	70.96%
Trainingsset 2	66.8%	70.56%

Da unser Korpus die gleiche Anzahl Sätze für jeden Dialekt hat, haben wir auch eine dialektspezifische Evaluation durchgeführt. Manuell korrigiert haben wir für jeden Dialekt ein Trainingsset von 4000 Token und ein entsprechendes Testset von 400 Token. Der Dialekt, der am besten abgeschnitten hat, ist Zürichdeutsch mit 74.67% gefolgt von Aaraudeutsch (siehe Tabelle 6). Das PoS-Tagging für Berndeutsch ergab die schlechtesten Resultate. Auffallend sind die grossen Unterschiede zwischen den einzelnen Dialekten, was auf die abweichende Orthographie zurückgeführt werden kann.

Tabelle 6: Dialektspezifische Evaluation

Dialekt	**Genauigkeit**
Aarau	74.29%
Basel	65.84%
Bern	63.81%
Ostschweiz	68.37%
Zürich	74.67%

Schliesslich kann man in der Evaluationstabelle für verschiedene Kontextlängen (Tabelle 7) sehen, dass diese Einstellungen nur minimale Unterschiede in der Genauigkeit

bringen. Mit der Kontextlänge 3 erzielte der TreeTagger 69.02%, jedoch auch mit der Kontextlänge 4. Somit wird klar, dass eine Kontextlänge, die grösser als 3 ist, keine Verbesserungen, wenn nicht sogar schlechtere Resultate bringt.

Tabelle 7: Evaluation Kontextlängen auf dem gesamten Trainings- & Testset

Kontextlänge	**Genauigkeit**
Kontextlänge 1	68.77%
Kontextlänge 2	68.97%
Konextlänge 3	69.02%
Konextlänge 4	69.02%

4.2 Trigrams'n'Tags-Tagger

4.2.1 Architektur

Als Vergleich haben wir die gleichen Texte vom TnT-Tagger taggen lassen. TnT steht für „Trigrams'n Tags". Es handelt sich um einen statistischen PoS-Tagger, basierend auf dem Markov Modell zweiter Ordnung. Für das Smoothing wird lineare Interpolation von Unigrammen, Bigrammen und Trigrammen verwendet. Unbekannte Wörter werden mit Suffix-Analyse behandelt, wobei der Term Suffix hier nicht linguistisch motiviert ist, sondern eine begrenzte Sequenz von Buchstaben am Ende des Wortes bezeichnet. Die Wahrscheinlichkeit für ein bestimmtes Suffix wird von den Wörtern des Trainingssets, die das gleiche Suffix tragen, bestimmt.

4.2.2 Training

Für das Training braucht der TnT-Tagger ein getaggtes Korpus mit zwei Spalten, wobei in der ersten Spalte die Token sind, in der zweiten das dazugehörige Tag. Während des Trainings erstellt der TnT-Tagger zwei Dateien, eine mit N-Grammen und ein Lexikon. Das erstellte Lexikon besteht aus mindestens vier Spalten pro Eintrag. In der ersten Spalte das Token, in der zweiten die Häufigkeit dieses Token im Trainingskorpus. In den weiteren Spalten stehen die Tags, welche mit dem Token aufgetreten sind zusammen mit ihren Häufigkeiten. Die Summe aller Häufigkeiten mit den verschiedenen Tags entspricht der Zahl in der zweiten Spalte.
Es gibt einige Optionen, um die Resultate des Trainings zu verändern. Unter anderen zum Beispiel eine Option *–c*, welche die Gross- und Kleinschreibung explizit markiert oder eine Option *–i*, welche die Gross- und Kleinschreibung ignoriert. Bei unseren Daten machen diese Optionen keinen grossen Unterschied ($< 0.5\%$).

4.2.3 Tagging

Anders als der TreeTagger muss dem TnT-Tagger kein Lexikon mitgegeben werden. Für den Prozess des Taggings verwendet er nur zwei Dateien mit den Modellparametern für

die Wort- und Kontexthäufigketien. Diese zwei Dateien werden vom Tagger selbst im Trainingsschritt erstellt. Zusätzlich muss natürlich ein Testset mitgegeben werden, dieses besteht nur aus einer Spalte, nämlich den Token.
Auch für diesen Prozess gibt es einige Optionen. Es kann zum Beispiel die Suffixlänge mit Option *–alength* verändert werden, mit Option *–umode* kann das Umgehen mit unbekannten Wörtern bestimmt werden, mit Option *-nn* kann die N-Gramm Ordnung fürs Tagging festgelegt werden und mit Option *–bfile* wird ein Backuplexikon mitgeben, um nur einige davon zu nennen. Auch mit diesen Optionen konnten wir keine markanten Verbesserungen erzielen. Die grösste Verbesserung erzielten wir mit einem Backuplexikon mit 0.8% im Vergleich zum Tagging ohne Optionen. Das Backuplexikon muss das gleiche Format haben wie das Lexikon, das vom Tagger selbst im Trainings-Schritt erstellt wird. Es muss also Informationen über die Häufigkeit eines Token mit den jeweiligen PoS-Tags enthalten. Wir haben das gleiche Lexikon benutzt wie beim Tree-Tagger, mit allen Wörtern der Tiger-Wortliste und der schweizerdeutschen Wortliste. Dieses haben wir aber nicht über einem Trainingsset trainiert, sonder nur mit „Dummy"-Häufigkeitswerten erweitert. Ein trainiertes Backuplexikon würde vielleicht zu einer grösseren Verbesserung führen.

4.2.4 Evaluation

Wie schon beim TreeTagger haben wir über das ganze Trainings- und Testset, über die einzelnen Testsets und auf Dialekte aufgeteilt evaluiert.

Resultat über unsere gesamte Korpora mit 20'000 Token im Trainingsset (4'000 pro Dialekt) und 2'000 Token im Testset (400 pro Dialekt):

85.13%

Beim Training und Testen über das ganze Korpus hat der TnT-Tagger 299 Token falsch getaggt. Die häufigsten zwei Fehler betreffen die Verwechslung von Eigennamen (NE) und Nomen (NN). 30 Mal hat der Tagger einen Eigennamen als Nomen getaggt und 24 Mal umgekehrt. Dies ist nicht weiter erstaunlich, da es ein allgemein bekanntes Problem von PoS-Tagging ist. Ein weiterer häufiger Fehler ist Präposition plus Artikel (APPRART) anstatt Artikel (ART), dies lässt sich vermutlich damit erklären, dass im Schweizerdeutschen APPRART viel häufiger als ART auftritt, weshalb die Wahrscheinlichkeit höher ist und bei unbekannten Wörtern eher das Tag APPRART vergeben wird. Das gleiche passiert wahrscheinlich mit der Verwechslung von Partizip perfekt (VVPP) und finitem Verb (VVFIN). Erstaunlich ist aber, dass der TnT-Tagger 14 Mal ein Adjektiv (ADJA) als Nomen (NN) taggt.

Wir haben zwei Trainingssets mit je 10'000 Token und zwei Testsets mit je 1'000 Token erstellt. Trainiert haben wir den TnT-Tagger, so wie zuvor den TreeTagger, einmal mit beiden Trainingssets einzeln und getestet haben wir je mit beiden Testsets.

Tabelle 8: Evaluation auf Testsets aufgeteilt

	Testset 1	**Testset 2**
Trainingsset 1	83.75%	82.83%
Trainingsset 2	82.06%	81.94%

Die Resultate reichen von 81.94% mit Trainingsset 2 und Testset 1, bis 83.75% mit Trainingsset 1 und Testset 1. Mit einem Training auf dem ganzen Trainingsset, also 20'000 Token und dem Testen auf 2'000 Token erzielten wir 85.13% Genauigkeit.
Tabelle 8: Training 10'000, Test 1'000 Token (2'000 bzw. 200 pro Dialekt)

Für die dialektspezifische Evaluation haben wir das Trainings- und Testset in ihre Dialekte aufgeteilt. Das heisst, der Tagger wurde pro Dialekt nur mit 4'000 Token trainiert und mit 400 Token des gleichen Dialekts getestet.

Tabelle 9: Dialektspezifische Evaluation

Dialekt	**Genauigkeit**
Aarau	80.24%
Basel	78.47%
Bern	83.10%
Ostschweiz	84.44%
Zuerich	87.20%

Die verschiedenen Dialekte weisen signifikante Unterschiede auf, die Ergebnisse reichen von 78.47% für Baseldeutsch bis 87.20% für Zürichdeutsch. Das beste Resultat mit dem TnT-Tagger erzielten wir also mit dialektspezifischem Training auf dem Zürichdeutsch-Korpus. Dies ist erstaunlich, wenn man bedenkt, dass das Trainingskorpus ein Fünftel des ganzen Trainingskorpus beträgt, mit welchem wir 85.13% erreicht haben.

4.3 Vergleich TreeTagger vs. TnT-Tagger

Um die Resultate des TnT-Taggers und des TreeTaggers zu vergleichen, haben wir zuerst die Genauigkeiten verglichen (siehe Tabelle 10) und danach versucht, die grossen Unterschiede zu begründen.

Tabelle 10: Vergleich TnT-Tagger vs. TreeTagger

	Genauigkeit
TreeTagger	68.97%
TnT	85.13%

Der grösste Unterschied zwischen den beiden Taggern, die beide auf Markov Modellen basieren, liegt beim Vorgehen: der TreeTagger benötigt im Unterschied zum TnT-Tagger ein Lexikon. Zudem verwendet der TreeTagger Entscheidungsbäume und der TnT-Tagger nicht. Der letzte integriert jedoch eine Suffixanalyse.

4.3.1 Häufige Fehler

Ebenfalls ist uns aufgefallen, dass der TreeTagger nicht diesselben Fehler macht wie der TnT-Tagger. Der TreeTagger erkennt keine Eigennamen (NE), sondern taggt alle Eigennamen als normale Nomen (NN) (58 mal im gesamten Korpus). Viele Artikel (ART) und Präpositionen (APPR) werden ebenfalls nicht erkannt und da schweizerdeutsche Artikel sehr oft nur aus einem Buchstaben bestehen, werden diese vom TreeTagger automatisch als Nichtwörter (XY) markiert (50 mal). Hinzu kommt, dass der TreeTagger die abgetrennten Verbzusätze (PTKVZ) nicht erkennt, obwohl er dies im Deutschen beherrscht, und diese als Präpositionen (APPR) markiert (20 mal). Ein seltener, aber trotzdem auffallender Fehler ist, dass der TreeTagger attributive Adjektive (ADJA) nicht von adverbialen oder prädikativen Adjektiven (ADJD) unterscheiden kann.
Wie auch der TreeTagger markiert der TnT-Tagger Artikel oft als Präpositionen (16 Mal). Dieser Tagger hat noch andere Schwierigkeiten als der TreeTagger: Oft verwechselt dieser Tagger Eigennamen und normale Nomen und umgekehrt (30 bzw. 24 Mal) und finite Verben (VVFIN) werden als Partizip Perfekt (VVPP) markiert (14 mal).

Natürlich sind unsere Resultate nicht optimal und nicht mit denjenigen für grössere Sprachen vergleichbar. Trotzdem sind sie viel besser als das Vortagging alleine (54.35%) und so kann ein statistisches Tagging für Schweizerdeutsch nicht ausgeschlossen, aber auch nicht als alleinstehende Methode angewendet werden.

5 Vergleiche mit anderen Sprachen

Zur Vor- und Nachbereitung haben wir unser Vorgehen mit anderen Ansätzen und Verfahren zu PoS-Tagging in wenig beachteten Sprachen oder Dialekten verglichen. Wir haben uns mit einigen Versuchen und Projekten befasst und werden im folgenden auf diejenigen von Bengali, Bhasa Indonesia und den arabischen Dialekten näher eingehen. 'Wenig beachtet' bezieht sich in diesem Fall auf die Sprachtechnologie und dessen Res-

sourcen und nicht auf die Sprecheranzahl, da sie in den drei genannten Sprachen in jedem Fall über 200 Millionen liegt.

5.1 Bengali

Bengali ist eine indoeuropäische Sprache und ist mit 215 Millionen Sprechern die siebtmeistgesprochene Sprache der Welt. Bengali verfügt über ein eigenes Schriftsystem. Diese Sprache ist wie Schweizerdeutsch sehr reich an Morphologie. [Dandapat et al., 2007] haben einen PoS-Tagger für Bengali konstruiert. Es stehen ebenfalls sehr wenig annotierte Korpora zur Verfügung. So wurde ein verhältnissmässig kleines Korpus mit 45'000 Wörtern verwendet. Diese wurden in 40'000 Wörter (3'625 Sätze) Trainingsmaterial und 5'000 Wörter (400 Sätze) Testmaretial geteilt. Es wurde eine Tagger gebraucht, der auf einem Hidden Markov Modell (HMM) und einem Maximum Entropy Modell basiert. Somit haben Dandapat et al herausgefunden, dass ein Maximum Entropy Modell für wenig Trainingsdaten bessere Resultate erzeugt. Die Maximum-Entropie-Methode ist eine Methode der Bayesschen Statistik, welche erlaubt, trotz mangelhafter problemspezifischer Information eine Anfangswahrscheinlichkeit zuzuweisen.
Da von Anfang an klar war, dass ein einfaches stochastische Vorgehen nicht ausreichen würde für eine Sprache wie Bengali, wurde eine Morphologie Analyse durchgeführt. Man kam zum Schluss, dass die Einbeziehung von morphologischen Merkmalen für das PoS-Tagging sehr hilfreich sein kann, beziehungsweise notwendig ist, speziell für Sprachen, für welche wenig Ressourcen wie annotierte Korpora vorhanden sind. Schliesslich präsentierten sie eine Genauigkeit von 88.75%.

Eine Morphologieanalyse wäre sicherlich auch ein möglicher, sinnvoller Verbesserungsansatz für schweizerdeutsches PoS-Tagging. Es wäre denkbar, eine Morphologieanalyse vor dem Tagging einzugliedern.

5.2 Bahasa Indonesia

Die indonesische Sprache gehört ebenfalls zu den meistgesprochenen Sprachen der Welt. Doch auch für diese Sprache besteht die Ressourcenarmut in der Sprachtechnologie. [Pisceldo et al., 2008] haben probabilistische Ansätze zu PoS-Tagging für Bahasa Indonesia entwickelt. Da kein standardisiertes indonesiches Korpus verfügbar ist, das annotiert wäre, haben diese Forscher, gleich wie wir, aufgrund der offiziellen indonesischen Grammatikschreibung ein eigenes Tagset mit 37 Wortartentags und Tags für Satzzeichen erstellt. Bahasa Indonesia kann in fünf Grundwortarten unterteilt werden: Verben, Adjektive, Nomen, Adverbien und Funktionswörter (Päpositionen, Artikel, usw.). Diese Grundwortarten werden weiter in Subkategorien unterteilt. Die Versuche wurden mit diesem Tagset und einem reduzierten Tagset mit nur 25 Tags durchgeführt.
Nach mehreren Experimenten mit einem Korpus aus Zeitungsartikeln (14'165 Wörter) und einem Teil der Penn Treebank (der laufend erweitert wird), wurde festgestellt, dass der Tagger mit der Maximum Entropy Methode, im Vergleich zu dem mit Conditional Random Fields (CRF), die besten Resultate erzielte. CRF ist ein stochastisches, unge-

richtetes Modell, das im Unterschied zu einem Hidden Markov Modell an jeder Stelle auf die komplette Information der Eingabesequenz zugreifen kann. Ein HMM hingegen sieht nur die aktuelle Eingabe. Für die Experimente mit der Maximum Entropy Methode wurde der Stanford Postagger und für die CRF das open source Werkzeug CRF++ angewendet. Das beste Resultat von Pisceldo et al. erbrachte eine Genauigkeit von 95.19%.

Je detailreicher ein Tagset ist, desto feinere Unterscheidungen müssen vom Tagger getroffen werden. Deshalb schnitten die Experimente für Bahasa Indonesia mit dem grösseren Tagset auch schlechter ab als jene mit dem kleineren Tagset. In Sprachen mit reichhaltiger Morphologie ergeben sich durchschnittlich weniger Ambiguitäten als in Sprachen, deren Token kaum morhpologische Merkmale aufweisen. In der indonesischen Sprache gibt es weder Deklination noch Konjugation und nur sehr wenig Verbflexion und Derivation. In der Folge ist es durchaus verständlich, dass das verwendete Tagset viel kleiner ist als das Tagset, das wir für die schweizerdeutschen Dialekte verwendet haben, zumal unser Tagset nicht als abgeschlossen bezeichnet werden kann.

5.3 Arabische Dialekte

Das Egyptian Colloquial Arabic, einer der vielen arabischen Dialekte, hat einige Gemeinsamkeiten mit dem Schweizerdeutschen: Beide werden hauptsächlich mündlich verwendet und eine Standardorthographie ist bei beiden nicht vorhanden. Zudem unterscheidet sich die moerne Standardsprache stark von den gepsrochenen Varianten. Obwohl es sehr schwierig war, für gesprochene arabische Dialekte grosse Mengen an Daten zu bekommen, haben [Duh and Kirchhoff, 2005] mit einem minimal überwachten Ansatz, welcher nur rohe Textdaten aus verschiedenen arabischen Dialekten und einem Morphologieanalysesystem für modernes Standardarabisch braucht, PoS-Tagging-Versuche für arabische Dialekte durchgeführt. Es kamen keine dialekt-spezifischen Werkzeuge zum Einsatz. Es wurde ebenfalls eine statistischer Tagger in der Form eines Hidden Markov Models benutzt, der jedoch ein viel grösseres Korpus zur Verfügung hatte, wofür aber extensive manuelle Annotation für Konsistenz nötig war. Das verwendete ECA Korpus (CallHome Egyptian Colloqial Arabic Corpus) besteht aus Telephongesprächen und ist bis heute das einzige annotierte Korpus für diesen Dialekt.
Duh und Kirchhoff präsentieren zwei Ansätze für ein cross-dialect sharing: Einerseits die *Conextual Model Interpolation*, eine weit verbreitete data-sharing Methode, die annimmt, dass Daten von verschiedenen Quellen "gemischt" werden können, mit dem Ziel, die beste Wahrscheinlichkeitsverteilung für die Zieldaten zu erreichen. Als zweite Methode stellen sie das *Joint Training of Contextual Models* vor, das ein einziges Modell mit zwei verschiedenen Datasets gemeinsam trainiert.
Hinzu kommen die Verbesserungen, die sie mit einem eingeschränkten Lexikon für das Erkennen von unbekannten Wörtern und die Berücksichtigung von Affix-Merkmalen erzielt haben. Diese Verbesserungen eignen sich gut, um die Genauigkeit zu erhöhen, auch wenn wenige Ressourcen vorhanden sind. Schlussendlich erreichen sie eine Genauigkeit von 76.29%.

Wir haben, im Vergleich zu diesen Ansätzen, eine allgemeinere Methode gewählt, nämlich die Dialekte von Anfang an zu mischen und sowohl in Trainings- als auch im Testset Texte aus allen Dialekten zu verwenden. Natürlich wäre dies bei einem grösseren Korpus sinnvoller. Eine andere Variante wäre, dialekt-spezifisches PoS-Tagging zu versuchen, wofür es jedoch zu wenig vorhandenes Trainingsmaterial gibt und auch in diesem Fall keine Konsistenzgarantie vorhanden wäre.

6 Erkenntnisse

6.1 Optimierungsmöglichkeiten

Wir haben erfahren, dass es ein schwieriges Unterfangen ist, ein konsistentes Korpus für Schweizerdeutsch zu erstellen. Die fehlende Standardorthographie und das daraus folgende unbeschränkte Tagset eignen sich nicht dafür. Als Notlösung gäbe es die Möglichkeit, unsere neuen, zusammengesetzten Tags durch Bindestriche zu trennen (z.B. VAFIN-PPER-PRF) um die Verständlichkeit zu erhöhen. Dies hätte auch zur Folge, dass nur das STTS verwendet werden müsste, da es nur diese Tags geben würde, die man beliebig zusammensetzen könnte. Dies würde bedeuten, dass kein neues, schweizerdeutsch-spezifisches Tagset nötig wäre, abgesehen von Ausnahmen wie zum Beispiel das INF-PART.
Eine andere Variante, das Problem des immer grösser werdenden Tagsets zu lösen, wäre die zusammengeklebten Wörter im Tokenisierungsschritt aufzusplitten. Somit hätte man beim Taggen keine Token mehr, die eigentlich aus mehreren Wortarten bestehen und könnte somit sinnvolle PoS-Tags vergeben.
Eine weitere Variante wäre, sich auf einen Dialekt zu konzentrieren, also dialektspezifisches PoS-Tagging. Wir könnten uns vorstellen, dass ein Versuch mit Zürichdeutsch, der Dialekt, der bei unserem Tagging mit 87.20% Genauigkeit am besten abgeschnitten hat, die Resultate deutlich verbessen könnte. Als Korpus könnte man zum Beispiel aus der Alemannischen Wikipidia nur die zürichdeutschen Artikel verwenden. Bei diesen hätte man aber nach wie vor das Problem der inkonsistenten Schreibung, vor allem auch wegen den verschiedenen Autoren. Deshalb haben wir uns überlegt, dass man als Korpus auch zum Beispiel die von Emil Weber übersetzte Bibel „s Nöi Teschtamänt Züritüütsch“ [12] verwenden könnte.

Ein weiterer Optimierungsansatz für das PoS-Tagging für Schweizerdeutsche wäre ein regelbasierter Ansatz, um auf morphologisches Ebene witerzukommen (siehe [Brill, 1994]). Oder vielleicht noch besser, ein hybrides Verfahren, wie jenes für das Koreanische von [Lee et al., 1997]. Wie auf der Abbildung 2 sichtbar, wird der Eingabetext vor dem Tagging einer Morphologieanalyse unterzogen. Das eigentliche Tagging beruht wiederum auf einem Hidden Markov Modell, das die lexikalischen Übergangswahrscheinlichkeiten

[12] Emil Weber, „s Nöi Teschtamänt Züritüütsch“, Jordan-Verlag, 2011

berechnet. Schliesslich wird dank Regeln zur Fehlerkorrektur die Qualität des Tagging noch verbessert.

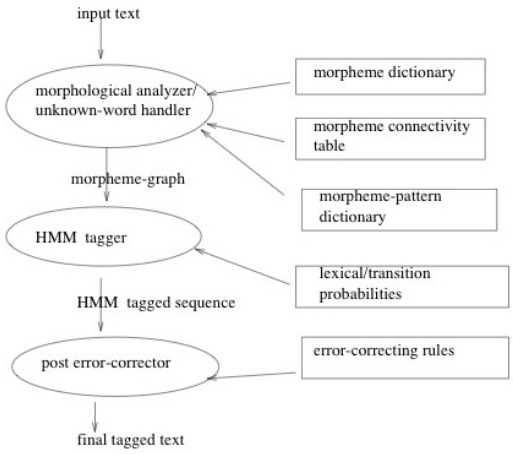

Abbildung 2: Diagramm des hybriden Verfahrens aus [Lee et al., 1997]

6.2 Zusammenfassung der Ergebnisse

Wir haben auf der Basis des Stuttgart-Tübingen-TagSets ein PoS-Tagset für Schweizerdeutsch erstellt und damit ein schweizerdeutsches Korpus mit 20'000 Token manuell getaggt. Dieses haben wir als Trainingsset für die Wortarten-Tagger verwendet. Für das PoS-Tagging haben wir zwei statistische Tagger gebraucht: den Trigrams'n Tags-Tagger und den TreeTagger. Mit diesen haben wir Genauigkeiten von 85.13% für den TnT-Tagger, beziehungsweise 68.97% für den TreeTagger erreicht. Zudem haben wir auch dialektspezifische Evaluationen durchgeführt, wobei der TnT-Tagger mit 87.20% Genaugkeit für Zürichdeutsch am besten abgeschnitten hat.
Unsere Versuche haben wir mit anderen Projekten verglichen, welche sich mit PoS-Tagging für sprachtechnologisch wenig beachtete Sprachen wie Bengali, Bhasa Indonesia und die arabischen Dialekte beschäftigt haben.
Zum Schluss haben wir einige Optimierungsmöglichkeiten für das Verfahren des PoS-Tagging für Schweizerdeutsch aufgezeigt. Wir vermuten, dass man mit einem hybriden System die Resultate verbessern könnte.

7 Literaturverzeichnis

Literatur

[Die, 2012] (2012). Dieth-schryybig.

[Brill, 1994] Brill, E. (1994). A simple rule-based part of speech tagger.

[Carstensen et al., 2010] Carstensen, K.-U., Ebert, C., and Jekat, S. (2010). *Einführung in die Computerlinguistik und Sprachtechnologie*. Spektrum Akademischer Verlag, Heidelberg, 3 edition.

[Dandapat et al., 2007] Dandapat, S., Sarkar, S., and Basu, A. (2007). Automatic part-of-speech tagginf for bengali: An approach for morphologically rich languages in a poor resource scenario. ACL.

[Duh and Kirchhoff, 2005] Duh, K. and Kirchhoff, K. (2005). Pos tagging of dialectal arabic: a minimally supervised approach. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*.

[Eddy, 2004] Eddy, S. R. (2004). What is a hidden markov model? *Nature Biotechnology*, 22(10):1315–1316.

[Glaser, 2003] Glaser, E. (2003). Schweizerdeutsche syntax: Phänomene und entwicklungen. In Beat Dittli, Annelies Häcki Buhofe, W. H., editor, *Gömmer MiGro?*, pages 39–66, Freiburg, Schweiz.

[Lee et al., 1997] Lee, G., Cha, J., and hyeok Lee, J. (1997). Hybrid pos tagging with generalize unknown-word handling.

[Pisceldo et al., 2008] Pisceldo, Adirani, and Manurung (2008). Probabilistic part of speech tagging for bahasa indonesia.

[Scherrer, 2011a] Scherrer, Y. (2011a). Morphology generation for swiss german dialects. In *Second Workshop on Systems and Frameworks for Computational Morphology (SFCM 2011)*, Zürich.

[Scherrer, 2011b] Scherrer, Y. (2011b). Syntactic transformations for swiss german dialects. In *First Workshop on Algorithms and Resources for Modelling of DIalects and Language Vareities*, Edinburgh. EMNLP.

[Scherrer and Rambow, 2010] Scherrer, Y. and Rambow, O. (2010). Natural language processing for the swiss german dialect area. Saarbrücken. KONVENS 2010.

[Schiller et al., 1999] Schiller, A., Teufel, S., Stöckert, C., Christine, and Thielen (1999). Guidelines für das taging deutscher textkorpora mit stts.

[Schmid, 1994] Schmid, H. (1994). Probabilistic part of speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

[Schmid, 1995] Schmid, H. (1995). Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT-Workshop*, Dublin.

Danke

Herzlichen Dank an Yves Scherrer (LATL, Faculté des lettres, Universität Genf) für die Bereitstellung seiner schweizerdeutschen Wortliste.

8 Anhang

8.1 Python-Scripte

8.1.1 Vortagging

```
# -*- coding: utf-8 -*-
import csv
import re
import sys

# Vortagging anhand von vorhandenen Wortlisten mit POS-Tags
# hier:
# - Schweizerdeutsche Wortliste mit POS-Tags, generiert von
#                Yves Scherrer
# - Deutsche Wortliste mit POS-Tags aus dem TIGER-Korpus
# - von uns manuell korrigierte Trainingstexte

# Aufruf: Datei, die getagged werden soll

# Ausgabe: TXT-Datei: Wort, Tab, POS-Tag, Newline

#-------------------------------------------------------------------

# Schweizerdeutsche Wortliste mit POS-Tags, generiert von
#       Yves Scherrer

reader = csv.reader(open('tiger_swg_lexicon_2012_12_08b.csv'),
delimiter=';')

chdeDict = {}
for row in reader:
        chdeDict[row[0]] = row[2]

# Deutsche Wortliste mit POS-Tags aus dem TIGER-Korpus

de_words = open("tigerwordpos.txt", "r").readlines()
deDict = {}
for line in de_words:
        line = line.lower().strip()
        list = re.split("\t", line)
        deDict[list[0]] = list[1].upper()

# von uns manuell korrigierte Trainingstexte
```

```
korr_train = open("trainingsset_all.txt", "r").readlines()
korrDict = {}
for line1 in korr_train:
        line1 = line1.strip()
        list1 = re.split("\t", line1)
        if len(list1) > 1:
                korrDict[list1[0]] = list1[1]

#-----------------------------------------------------------------

# taggt Woerter, die in oben kreierten Dictionaries vorhanden sind
#       sowie Zahlen und Satzzeichen
# Datenstruktur: Liste mit (Wort, POS)-Tupeln

infile = open(sys.argv[1], "r").readlines()
name = "tag_"+sys.argv[1]
outfile = open(name, "w")
testList = []
for line in infile:
        line = line.strip().lower()
        if line in korrDict:
                tuple = (line, korrDict[line])
                testList.append(tuple)
        elif line in chdeDict:
                tuple = (line, chdeDict[line])
                testList.append(tuple)
        elif line in deDict:
                tuple = (line, deDict[line])
                testList.append(tuple)

# folgende Zeilen falls Satzzeichen oder Zahlen in den Wortlisten
#       nicht vorhanden sind

        elif line == ",":
                tuple = (line, "$,")
                testList.append(tuple)
        elif line in ".!?:;":
                tuple = (line, "$.")
                testList.append(tuple)
        elif line in "()[]-":
                tuple = (line, "$(")
                testList.append(tuple)
        elif re.match("\d+", line):
```

```
                tuple = (line, "CARD")
                testList.append(tuple)
        else:
                tuple = (line, "unknown")
                testList.append(tuple)

# Ausgabe: Wort, Tab, POS-Tag, Newline

for tup in testList:
        outfile.write("{0}\t{1}\n".format(tup[0], tup[1]))
outfile.close()
```

8.1.2 Evaluation

```
# -*- coding: utf-8 -*-
from __future__ import division
import re
import sys
from collections import defaultdict



# Evaluation: vergleicht Goldstandard mit Tagger-Resultat
#       und berechnet Genauigkeit (accuracy)

# Aufruf mit 2 Dateinamen: Goldstandard und Tagger-getaggte Datei

# Ausgabe: TXT Datei: Genauigkeit & Liste mit Fehlern
#       --> 1. Spalte Goldstandard, 2. Spalte Tagger-Resultat,
#                3. Spalte Haeufigkeit des Fehlers

gs = open(sys.argv[1], "r").readlines()
tagger = open(sys.argv[2], "r").readlines()
name = "ev_"+sys.argv[2]
differences = open(name, "w")

gsList = []
tList = []
evalList = []

for gsLine in gs:
        gsLine = gsLine.strip()
        gsLine = re.split("\t", gsLine)
        gsList.append(gsLine[1])
```

```
for tLine in tagger:
        tLine = tLine.strip()
        tLine = re.split("\t", tLine)
        tList.append(tLine[1])

currLine = 0
equal = 0

while currLine < len(gsList):
        if gsList[currLine] == tList[currLine]:
                equal = equal + 1
        else:
                tuple = (gsList[currLine], tList[currLine])
                evalList.append(tuple)
        currLine = currLine + 1

accuracy = equal/len(gsList) * 100
result = round(accuracy,2)

# Ausgabe auf Kommandozeile

print "{0} von {1} korrekt getagged".format(equal, len(gsList))
print "accuracy: {0}%".format(result)

# Ausgabe in Ausgabedatei

differences.write("# {0} von {1} korrekt getaggt".format(equal,
        len(gsList)) + "\n")
differences.write("# accuracy: {0}%".format(result) + "\n")

evalDict = defaultdict(int)
for tup in evalList:
        evalDict[tup] += 1

for k in sorted(evalDict, key = evalDict.get, reverse = True):
        differences.write("{0}\t{1}\n".format(k, evalDict[k]))
differences.close()
```

8.2 Stuttgart-Tübingen Tagset

Auf den nächsten zwei Seiten ist das vollständige STTS-Tagset, wie es auf den Seiten sechs und sieben der STTS-Guidelines aufgeführt wird ([Schiller et al., 1999]).

2.3 Tag-Tabelle

POS =	Beschreibung	Beispiele
ADJA	attributives Adjektiv	*[das] große [Haus]*
ADJD	adverbiales oder prädikatives Adjektiv	*[er fährt] schnell* *[er ist] schnell*
ADV	Adverb	*schon, bald, doch*
APPR	Präposition; Zirkumposition links	*in [der Stadt], ohne [mich]*
APPRART	Präposition mit Artikel	*im [Haus], zur [Sache]*
APPO	Postposition	*[ihm] zufolge, [der Sache] wegen*
APZR	Zirkumposition rechts	*[von jetzt] an*
ART	bestimmter oder unbestimmter Artikel	*der, die, das,* *ein, eine*
CARD	Kardinalzahl	*zwei [Männer], [im Jahre] 1994*
FM	Fremdsprachliches Material	*[Er hat das mit "]* *A big fish [" übersetzt]*
ITJ	Interjektion	*mhm, ach, tja*
KOUI	unterordnende Konjunktion mit "zu" und Infinitiv	*um [zu leben],* *anstatt [zu fragen]*
KOUS	unterordnende Konjunktion mit Satz	*weil, daß, damit,* *wenn, ob*
KON	nebenordnende Konjunktion	*und, oder, aber*
KOKOM	Vergleichspartikel, ohne Satz	*als, wie*
NN	Appellativa	*Tisch, Herr, [das] Reisen*
NE	Eigennamen	*Hans, Hamburg, HSV*
PDS	substituierendes Demonstrativ–pronomen	*dieser, jener*
PDAT	attribuierendes Demonstrativ–pronomen	*jener [Mensch]*
PIS	substituierendes Indefinit–pronomen	*keiner, viele, man, niemand*
PIAT	attribuierendes Indefinit–pronomen ohne Determiner	*kein [Mensch],* *irgendein [Glas]*
PIDAT	attribuierendes Indefinit–pronomen mit Determiner	*[ein] wenig [Wasser],* *[die] beiden [Brüder]*
PPER	irreflexives Personalpronomen	*ich, er, ihm, mich, dir*
PPOSS	substituierendes Possessiv–pronomen	*meins, deiner*
PPOSAT	attribuierendes Possessivpronomen	*mein [Buch], deine [Mutter]*
PRELS	substituierendes Relativpronomen	*[der Hund,] der*

POS =	Beschreibung	Beispiele
PRELAT	attribuierendes Relativpronomen Relativpronomen	*[der Mann ,] dessen [Hund]*
PRF	reflexives Personalpronomen	*sich, einander, dich, mir*
PWS	substituierendes Interrogativpronomen	*wer, was*
PWAT	attribuierendes Interrogativpronomen	*welche [Farbe], wessen [Hut]*
PWAV	adverbiales Interrogativ– oder Relativpronomen	*warum, wo, wann, worüber, wobei*
PAV	Pronominaladverb	*dafür, dabei, deswegen, trotzdem*
PTKZU	"zu" vor Infinitiv	*zu [gehen]*
PTKNEG	Negationspartikel	*nicht*
PTKVZ	abgetrennter Verbzusatz	*[er kommt] an, [er fährt] rad*
PTKANT	Antwortpartikel	*ja, nein, danke, bitte*
PTKA	Partikel bei Adjektiv oder Adverb	*am [schönsten], zu [schnell]*
TRUNC	Kompositions–Erstglied	*An– [und Abreise]*
VVFIN	finites Verb, voll	*[du] gehst, [wir] kommen [an]*
VVIMP	Imperativ, voll	*komm [!]*
VVINF	Infinitiv, voll	*gehen, ankommen*
VVIZU	Infinitiv mit "zu", voll	*anzukommen, loszulassen*
VVPP	Partizip Perfekt, voll	*gegangen, angekommen*
VAFIN	finites Verb, aux	*[du] bist, [wir] werden*
VAIMP	Imperativ, aux	*sei [ruhig !]*
VAINF	Infinitiv, aux	*werden, sein*
VAPP	Partizip Perfekt, aux	*gewesen*
VMFIN	finites Verb, modal	*dürfen*
VMINF	Infinitiv, modal	*wollen*
VMPP	Partizip Perfekt, modal	*[er hat] gekonnt*
XY	Nichtwort, Sonderzeichen enthaltend	*D2XW3*
$,	Komma	**,**
$.	Satzbeendende Interpunktion	**. ? ! ; :**
$(	sonstige Satzzeichen; satzintern	– [](}