Institut für Logik, Komplexität und Deduktionssysteme
der Universität Karlsruhe
Lehrstuhl Prof. Dr. A. Waibel

Automatische Generierung eines Ausspracheswörterbuches und Initialisierung eines Erkenners für die kroatische Sprache

Studienarbeit
von

Stefan Raschke

Betreuerin: Dipl.-Inform. Tanja Schultz

15. Januar 1998

Zusammenfassung

Bei der Erstellung eines neuen Spracherkenners wird man mit dem Problem konfrontiert, eine geeignete Initialisierung zu finden. Gegenstand dieser Arbeit ist, u. a. die Ergebnisse unterschiedlicher Initialisierungen miteinander zu vergleichen. Die Tests wurden mit Spracherkennern für die kroatischen Sprache durchgeführt. Für jeden der Tests wurde ein eigenes System erstellt. Das System, das die besten Ergebnisse lieferte wurde anschließend für das weitere Training verwendet. Diese Arbeit befaßt sich weiterhin mit Eigenschaften der kroatischen Sprache, die die automatische Spracherkennung beeinflußen. Ein aufwendiger Vorgang bei der Erstellung eines Spracherkenners ist der Aufbau eines Aussprachewörterbuches. Dieser Vorgang wurde im Laufe der Arbeit weitgehend automatisiert.

Inhaltsverzeichnis

1 Einleitung **7**

2 Spracherkennung mit Janus **9**

2.1 Spracherkennung 9

2.1.1 Wahrscheinlichkeitstheoretische Betrachtungen 9

2.1.2 Ablauf der Spracherkennung 10

2.2 Janus Recognition Toolkit (JRTk) 11

3 Die kroatische Sprache und ihre Eigenschaften **13**

3.1 Ursprung 13

3.2 Grammatik 14

3.3 Lauteigenschaften 16

4 Erstellung des kroatischen Spracherkenners **18**

4.1 Daten 18

4.1.1 Datensammlung und Transkribierung 18

4.1.2 Romanisierung der Transkriptionen 19

4.2 Abbildung auf Phoneme 20

4.3 Janus Datenbasis 21

4.4 Aussprachewörterbuch 21

4.4.1 Das Tool `erstelltAWB` 21

4.4.2 Ablauf von `erstelltAWB` 22

4.5 Language Model 22

4.6 Training von unterschiedlich initialisierten Systemen 23

4.6.1 Blick über die verwendeten Daten 23

4.6.2 Unterschiedliche Initialisierungen 25

4.6.3 Training 27

4.6.4 Bestimmung geeigneter Testparameter 27
4.6.5 Ergebnisse der Testläufe 29
4.7 Vergrößerung der Trainingsmenge 30
4.7.1 Unterschiede zur bisherigen Trainingsmenge 30
4.7.2 Ergebnisse 30

5 Kontextabhängiger Erkenner 33

5.1 Warum eine kontextabhängige Modellierung? 33
5.2 Polyphone 33
5.3 Voreinstellungen 33
5.4 Trainingsläufe 34
5.5 Testläufe 34
5.6 Besprechung der Ergebnisse 35

6 Das Skript `erstelltAWB` 37

6.1 Ein- und Ausgabe 37
6.2 Darstellung einfacher Wörter 38
6.3 Ziffernfolgen 38
6.4 Folgen von Großbuchstaben 40

7 Ausblick und Betrachtungen 41

1 Einleitung

Könnten wir Recheneinheiten für unsere Maschinen konstruieren, die unsere sprachlichen Eingaben genau verstehen, wäre unsere Welt um vieles einfacher (zu steuern). Es wäre möglich unsere Waschmaschine durch ein dahingesagtes Wort einzuschalten. Wir könnten Briefe an unsere ausländischen Freunde diktieren und diese in deren Landessprache übersetzt ausdrucken. Man könnte mit Hilfe eines elektronischen Dolmetschers mit den japanischen Geschäftspartnern verhandeln. Ein Pilot könnte in Situationen, in denen er beide Hände benötigt, weitere Systeme per Spracheingabe steuern.

Die Forschung ist auf dem Weg in diese Richtung. Die existierenden Probleme sind allerdings zahlreich. Es gibt für die einzelnen Probleme zwar schon zufriedenstellende Lösungsansätze, aber gemeinsam stellen sie ein gewaltige Hürde dar. Diese Hürde ist, wie so oft, zu einem großen Teil die mangelnde Rechenleistung. Die Vergangenheit hat aber gezeigt, daß durch gute Algorithmen diese Hürde überwunden werden kann.

Natürlich stellt die Spracherkennung genügend noch zu lösende Probleme zur Verfügung, die mehr in der Natur der Sache liegen. Wie würde obige Waschmaschine reagieren, wenn der neue Besitzer einen Sprachfehler hat und noch fünf andere Waschmaschinen in der Waschküche laufen. Wie schaffen wir es, der Diktiermaschine neue Sprachen beizubringen und sie verschiedene Sprecher akzeptieren zu lassen. Wie erreicht man es, daß ein empfindlicher Geschäftsmann nicht durch seltsame oder gar falsche Satzkonstruktionen abgeschreckt wird. Tragisch könnte es auch sein, wenn der gestreßte Pilot einen falschen Tonfall anschlägt und seine Maschine ihn nicht oder gar falsch versteht.

Die obigen Beispiele stellen nur einen Bruchteil der Situationen dar, in denen Spracherkennung einen sinnvollen Beitrag zur Vereinfachung des Alltags leistet. Doch sind, um die angeschnittenen Aufgaben wie Sprecherunabhängigkeit, Stabilität gegenüber Nebengeräuschen oder Multilingualität in den Griff zu bekommen, noch viele Jahre der Forschung nötig.

Um immer mehr Sprachen abzudecken, muß man neue Systeme entwickeln oder bestehende Erkenner ausbauen. Stehen bereits funktionierende Systeme zur Verfügung, kann man diese bei der Erstellung eines Erkenners für ein neue Sprache verwenden. Ziel dieser Arbeit ist, mehrere dieser Initialisierungsmöglichkeiten zu vergleichen. Als zu betrachtende Sprache wurde Kroatisch gewählt. Um eine gemeinsame Basis für den Vergleich der Ergebnisse zu haben, wurde die Architektur des Erkenners bei allen Systeme unverändert beibehalten.

Das bei dieser Arbeit verwendete Werkzeug ist das Janus *Recognition Toolkit* (JRTk). Alle Komponenten, die für einen Spracherkenner notwendig sind, mußten im Laufe der Arbeit für die Kroatische Sprache entwickelt werden. Besonderes Augenmerk wurde dabei auf die Erstellung des Aussprachewörterbuchs gelegt. Diese, oft manuelle Tätigkeit, wurde für das kroatische Wörterbuch weitgehend automatisiert.

Für die ersten Erkenner, die im Laufe dieser Arbeit erstellt wurden, kam nur eine Teilmenge der vorhandenen Daten zur Verwendung. Die Testläufe mit den verschiedenen Initialisierungen waren zahlreich. Jeder einzelne Durchlauf mit vollem Trainingsmaterial hätte zu lange gedauert. Das Ziel der Experimente war die Ergebnisse der Initialisierungen in Beziehung zu setzen, um dann das beste System mit vollem Trainingsmaterial neu zu trainieren. Als weitere Verbesserung war vorgesehen dieses System dann *kontextabhängig* weiterzuentwickeln.

2 Spracherkennung mit JANUS

Die nun folgende Beschreibung der Vorgänge bei der Spracherkennung hat keineswegs den Anspruch auf Vollständigkeit. Vielmehr soll nur ein kurzer Abriß der komplizierten Abläufe gegeben werden. Die Ausführungen in den Abschnitten über die allgemeine Spracherkennung sind an [Schu95a, Schu95b, Rogi96] angelehnt. Die Aussagen über JANUS basieren auf [LWLF97, FGHK97].

2.1 Spracherkennung

2.1.1 Wahrscheinlichkeitstheoretische Betrachtungen

Der Vorgang der Erkennung *kontinuierlicher* Sprache läßt sich in der Wahrscheinlichkeitstheorie auf folgende Aufgabe reduzieren. Sei X die Folge der beobachteten Merkmalvektoren, also die akustische Eingabe. Als W bezeichnen wir das vorhandene Vokabular und mit W^* die Menge aller möglichen Lösungen. Mit $\hat{w}$ wird die Wortkette $w = w_1 \dots w_m$ aus der Menge von *Hypothesen* bezeichnet, die am besten zur Eingabe paßt.

$$\hat{w} = \operatorname*{argmax}_{w \in W^*} P(w|X) \tag{1}$$

Eine Lösung basierend auf der Bayes-Entscheidungsregel liefert die Formel

$$P(w|X) = \frac{P(X|w) \cdot P(w)}{P(X)} \tag{2}$$

Die einzelnen Terme der Gleichung (2) haben folgende Bedeutung:

- $P(X)$
  stellt die mittlere Wahrscheinlichkeit dar, mit der X beobachtet wird. Sie spielt bei der Bestimmung der besten Wortkette keine Rolle.
- $P(X|w)$
  gibt die bedingten Verteilungsdichten dafür an, daß die Merkmalsequenz X beobachtet wird, wenn die ausgesprochene Wortsequenz w ist.
- $P(w)$
  steht für die *a-priori* Wahrscheinlichkeit. Dieser Ausdruck gibt an, mit welcher Wahrscheinlichkeit die Wortsequenz $w = w_1 \dots w_m$ beobachtet wird.

Der Ausdruck $P(X|w)$ steht für die akustische Modellierung der Spracheingabe. Spracherkenner arbeiten im Allgemeinen nicht mit Worten, sondern verwenden *Phoneme* oder ähnliche sprachliche Untereinheiten. Das Festlegen dieser stark sprachspezifischen Einheiten ist einer der ersten Schritte bei der Erstellung des Erkenners.

Ein *Aussprachewörterbuch* versorgt das System mit einem Nachschlagewerk, in dem die Worte in ihre Untereinheiten aufgeteilt sind. Ein Wörterbuch muß alle Wörter der Trainingsdaten enthalten. Idealerweise enthält es alle Wörter, die in den zu erkennenden Eingaben vorkommen.

Die eigentliche Spracheingabe wird in eine Reihe von Merkmalvektoren transformiert. Mit Hilfe eines geeigneten stochastischen Modells wird eine Teilfolge von Merkmalvektoren einem Phonem zugeordnet. Die so modellierte Phonemfolge wird dann mit Hilfe des Aussprachewörterbuchs und eines statistischen Wortfolgemodells (Language Model) durch eine Wortkette ersetzt. Für jede Eingabe werden mehrere Wortketten gebildet. Die Wortkette mit dem besten *score* wird als *Hypothese* ausgegeben.

Die Bestimmung des Terms $P(w)$ geschieht über das *Language Model (LM)*. Um Aussagen über die Beobachtungswahrscheinlichkeiten eines bestimmten Wortes machen zu können, muß die *Historie* des Wortes berücksichtigt werden. Deshalb arbeitet man mit *N-grammen*. Dies bedeutet, daß zusätzlich zu dem aktuellen Wort seine $n-1$ Vorgänger betrachtet werden. Zu jedem beobachteten N-gramm wird festgestellt, wie oft es im gegebenen Textkorpus vorkommt. Auf dieser Häufigkeit basierend wird dem N-gramm eine Wahrscheinlichkeit zugeordnet. Also wird $P(\text{dich}|\text{mag, ich})$ einen höheren Wert haben als $P(\text{du}|\text{mag, ich})$. Als guter Kompromiß zwischen Erkennungsleistung und Rechenaufwand hat sich eine Historie der Größe *drei* ergeben. Zusätzlich zu den *Trigrammen* wird noch eine Historie von *Zwei* und von *Eins* benötigt. Diese *Bi-* und *Unigramme* liefern Ergebnisse für Wörter, die mit der gegebenen höheren Historie nicht beobachtet wurden. Ein LM, wie es von JANUS verwendet wird, arbeitet also mit folgende Wahrscheinlichkeiten.

- $P(w_i|w_{i-2}w_{i-1})$
  Trigramme sind gut für die sprachlichen Modellierung geeignet. Der Nachteil ist, daß viele benötigte Trigramme in den gegebenen Trainingsmaterialen nicht vorkommen
- $P(w_i|w_{i-1})$
  Bigramme haben den Vorteil, daß sie oft beobachtet werden und ihre Grammatiken ausreichend Informationen enthalten.
- $P(w_i)$
  Unigramme stellen die Wahrscheinlichkeit dar, daß das Wort beobachtet wird.

2.1.2 Ablauf der Spracherkennung

Um die gesprochene Sprache des Menschen einem Rechner "verständlich" zu machen, muß das kontinuierliche Signal zuerst in eine digitale Form gebracht werden. Im Folgenden wird als Beispiel eine Form der Vorverarbeitung geschildert. Das Signal wird zuerst gefiltert und abgetastet [RaSc78]. Durch eine *diskrete Fouriertransformation* mit sich anschließender Filterung und Transformation in den *Cepstralbereich* erreicht man eine parametrische Darstellung der Übertragungsfunktion ohne die störenden Komponenten der Anregung.

Um die Information, die in den Merkmalvektoren steckt, nutzen zu können, werden sie verschiedenen Klassen zugeordnet. Diese Klassen, die *Kodebücher*, sind bei einem *kontextunabhängigen* Aufbau des Erkenners durch die Phoneme zu Beginn schon festgelegt. Bei einem *kontextabhängigen* Erkenner werden die Klassen im Laufe des Erkenneraufbaus erstellt. Mit Hilfe einer *linearen Diskriminanzanalyse (LDA)* oder einer ähnlichen Transformation lassen sich die Klassengebiete unter Wahrung der Kompaktheit auseinanderziehen. Unter Anwendung eines *nearest neighbor*-Algorithmus werden dann aus einer Stichprobe der Merkmalvektoren geeignete Vektoren für die *gaußschen Mischverteilungen* der Kodebücher festgelegt.

Die Daten müssen vor dem Training aufgeteilt werden. Dies geschieht, um nach dem Training eine zu den Trainingsdaten disjunkte Menge an Testdaten zur Verfügung zu haben. Die Aufteilung erfolgt in drei Datensätze. Die *Trainingsdaten* dienen ausschließlich zum Training. Die *Entwicklungsdaten* werden verwendet um einige Parameter optimal einzustellen. Wenn alle Einstellungen vorgenommen wurden, wird mit den *Evaluationsdaten* der eigentliche Test der Erkennungsleistung vorgenommen. Das Ziel des Trainings ist es, die Kodebücher zu verbessern. Daneben werden die Gewichte der gaußschen Mischverteilungen angepaßt.

Die Testläufe ordnen einer Spracheingabe eine Folge von Worten zu. Als stochastisches Modell können die *Hidden Markov Models (HMM)* verwendet werden. Die dabei zur Verfügung stehenden Möglichkeiten sind zu zahlreich um sie erschöpfend abzuarbeiten. Deshalb wird der Suchbereich eingeschränkt. Das sogenannte *pruning* bewirkt, daß nur eine festgelegte Anzahl von bisher gefundenen besten Lösungen weiterverfolgt wird. Am Ende der Suche werden die N-besten Hypothesen als Lösung ausgegeben.

2.2 JANUS Recognition Toolkit (JRTk)

Das bei der Erstellung des Spracherkenners verwendete System ist das JANUS Recognition Toolkit (JRTk). JANUS wurde in den Interactive Systems Laboratories an der Universität Karlsruhe und an der Carnegie Mellon University entwickelt. Die Aufgabe von JANUS ist, spontan gesprochene Sprache zu erkennen und in eine anderen Sprache übersetzt auszugeben. JANUS erreicht eine Erkennungsleistung von 75-90%. Die erkannte Eingabe wird in 70% der Fälle korrekt übersetzt. Der Erkennungsvorgang ist sprecherunabhängig. Dies wird durch Trainingsmaterial erreicht, daß aus unterschiedlichen Sprechern aufgebaut ist. Der modulare Aufbau des Toolkits erlaubt es dem Anwender JANUS für viele Erkennungsaufgaben zu konfigurieren. Ein Beispiel ist die Handschrifterkennung. Der, in *C* und *tcl/Tk* ausgeführte, objektorientierte Ansatz erlaubt einfache Benutzerführung. Zusätzlich wird dadurch eine hohe Ausbaufähigkeit des Systems erreicht. Tk erlaubt es, neue Visualisierungsmöglichkeiten für Spracheigenschaften zu realisieren.

3 Die kroatische Sprache und ihre Eigenschaften

Dieser Abschnitt versucht sich an einer Beschreibung der kroatischen Sprache. Bei dieser Sprache handelt es sich um die Sprache, die im heutigen Kroatien gesprochen wird. Sie unterscheidet sich von dem Kroatisch, das in der früheren jugoslawischen Republik gesprochen wurde dahingehend, daß es durch weitere Wörter, speziell türkische, ergänzt wurde. Eine von allen Bevölkerungsgruppen im ehemaligen Jugoslawien akzeptierte Sprache existiert nicht. Die serbokroatische Sprache, die früher eine sogenannte "offizielle" Funktion hatte, existiert nicht mehr. Selbst vor dem Krieg auf dem Balkan war es schwierig die Sprachen eindeutig zu identifizieren. Trotz Literatur und Rücksprachen mit Muttersprachlern kann hier nur ein rudimentärer Überblick gegeben werden. Speziell die grammatikalischen Eigenschaften der Sprache werden nur in soweit betrachtet, wie sie für die automatische Spracherkennung notwendig erscheinen.

3.1 Ursprung

Die kroatische Sprache ist eine *slawische* Sprache. Die heute noch gesprochenen slawischen Sprachen teilen sich in drei Gruppen auf. Die erste Gruppe ist die *Ostslawische* mit der größten Anzahl an Sprechern. Zur ihr gehört u.a. das Russische. Die zweite Gruppe ist die *Westslawische* mit Sprachen wie Polnisch und Tschechisch [Stör97]. Das Kroatische gehört zur dritten, der *Südslawischen* Gruppe.

Alle drei Gruppen stammen von einer gemeinsamen Muttersprache ab, dem *Urslawischen*. Diese Mutter aller slawischen Sprachen wurde noch in den ersten Jahrhunderten unserer Zeitrechnung gesprochen. Ein späterer Vertreter dieser gemeinsamen Ursprungssprache ist das *Kirchenslawisch*. Diese Form wurde im 11. Jahrhundert von allen slawischen Stämmen verstanden. Sie ist in veränderter Form als Sprache der orthodoxen Kirche erhalten geblieben.

Aufgrund der kriegerischen Auseinandersetzungen im ehemaligen Jugoslawien und der dementsprechenden Zersplitterung der verschiedenen Bevölkerungsgruppen, hat sich viel an der dortigen Sprachlandschaft geändert. Es ist schwierig, richtige Zuordnungen vorzunehmen. Die weiteren Ausführungen orientieren sich an [Brow93] und an Rücksprachen mit Muttersprachlern.

Jede der jugoslawischen Republiken hatte ihre eigene Sprache. Die Tabelle 1 ordnet den Republiken ihre Sprache und Schrift zu. Zusätzlich zu den eigenen Sprachen sprechen die Angehörigen aller Republiken auch Serbokroatisch, das eine Mischung aus Serbisch und Kroatisch darstellt. Daraus resultierte der Einsatz des Serbokroatischen als offizielle Sprache zur Verständigung zwischen den Angehörigen der verschiedenen Republiken.

Republik	# Sprecher	Sprache	Schrift
Kroatien	4,5 Mio	Kroatisch	Lateinisch
Serbien	8 Mio	Serbisch	Kyrillisch
Bosnien-Herzegowina	n/a	Serbokroatisch	Lateinisch
Mazedonien	n/a	Mazedonisch	Kyrillisch
Slovenien	n/a	Slovenisch	Lateinisch
Montenegro	600.000	Montenegro	Lateinisch

Tabelle 1: Sprachzugehörigkeiten

Nach dem Krieg kam es zur endgültige Trennung der Sprachen. Alle Nationen versuchen sich mit ihrer Sprache von der Serbokroatischen zu entfernen, die nach der Unabhängigkeit der einzelnen Republiken ihre Aufgabe als offizielle Sprache verloren hat. So wurde auch die Kroatische Sprache durch neue Worte ergänzt, die aus dem Serbischen und Türkischen stammen. Durch diese "Ergänzungen" der Sprachen ergeben sich für viele Begriffe mehrere verschiedene Wörter. Diese *Synonyme* ersetzen einander im Sprachgebrauch oft willkürlich.

3.2 Grammatik

Bei der kroatischen Sprache handelt es sich um eine *agglutinierende* Sprache. Das Wort *agglutinieren* bedeutet *anleimen*. Bei der Agglutination werden mehrere Wörter bzw. Silben aneinandergeleimt. So ist z.B. das französische Wort *aujourd'hui* eine Verkettung der vier Wörter *au jour de hui*.

Im Gegensatz zur französischen Sprache, in der die Agglutination nur bei vereinzelten Worte vorkommt, bildet das Kroatische viele Elemente der Grammatik auf diese Art. Im Kroatischen werden Pronomen, Nomen, Adjektive und Ordnungszahlen dekliniert. Es gibt sieben Fälle, drei Geschlechter sowie den Plural und den Singular. Die Deklination geschieht, indem an einen Wortstamm eine entsprechende Endung angehängt wird. Das Geschlecht wird ebenfalls über ein verändertes Suffix angezeigt. Die kroatische Grammatik kennt fünf Zeiten. Jede verwendet ihre eigenen Endungen.

Diese Verkettungen führen in den agglutinierenden Sprachen zu einer großen Anzahl von verschiedenen Wörtern, die einen gleichen Stamm besitzen, sich aber nur in wenigen Buchstaben unterscheiden. Im Kroatischen, aber auch in anderen Sprachen wie dem Türkischen, führt hauptsächlich diese Eigenschaft zu einem starken Vokabularwachstum. Hinzu kommt noch die im vorhergehenden Abschnitt erwähnte Tatsache, daß mehrere unterschiedliche Wörter für die gleichen Begriffe vorkommen. Ein Beispiel für die zahlreichen Synonyme sind die kroatischen Zahlwörter. Auf diese wird

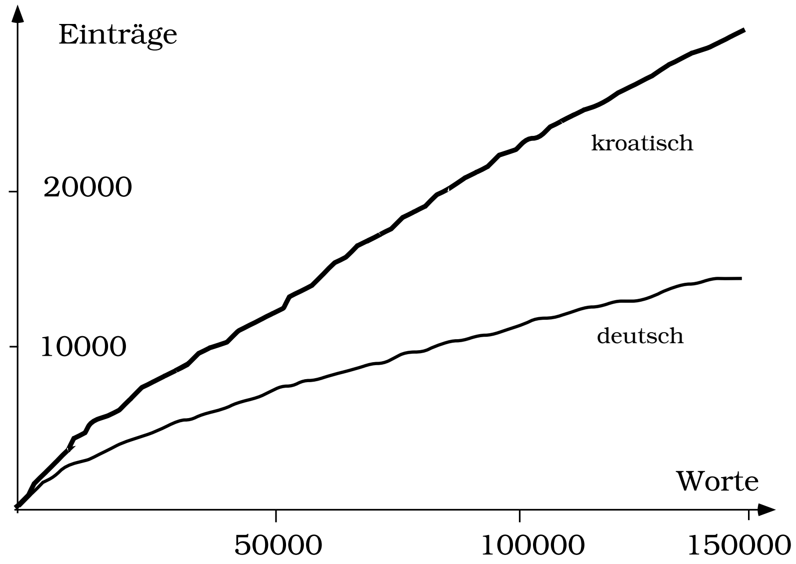

Abbildung 1: Vokabularwachstum

im Kapitel 6.3,das das Erstellen des Aussprachewörterbuchs behandelt, näher eingegangen. Durch diese Synonyme wird die Anzahl der Wörter weiter erhöht. Selbst bei einem Textkorpus mit mehr als 150000 Worten ist keine Abschwächung des Vokabularwachstums zu erreichen. Das Vokabular wächst dabei pro 50000 Worte im Textkorpus um 10000 Worte an. Siehe hierzu auch Abbildung 1. Zum Vergleich wurde ein deutscher Textkorpus herangezogen, der ebenfalls von einem Diktiersystem stammt. Bei gleicher Größe des Textkorpus erreicht man im Deutschen nur 15000 verschiedene Vokabulareinträge. Um dieselbe Anzahl von Vokabulareinträgen wie im Kroatischen zu erzielen, benötigt man einen deutschen Textkorpus mit 540000 Wörtern.

Das oben beschriebene Vokabularwachstum hat zur Folge, daß sowohl das Aussprachewörterbuch als auch das Language Model bei der verwendeten Vokabulargröße einen großen Umfang annehmen. Durch die große Anzahl von Wörtern mit gleichen Stämmen und Endungen, die sich oft nur in wenigen Buchstaben unterscheiden, entstehen hohe Verwechslungsraten. Diese Ambiguitäten verschlechtern die Erkennungsleistung.

Trotz der beschränkten Größe des Korpus ergibt sich eine große Wortvielfalt. Dies führt beim Erstellen des Language Models zu schlechten ngramm-Schätzungen. Das starke Vokabularwachstum bedeutet ebenfalls, daß die Testdaten viele Worte enthalten, die der Erkenner nicht kennt, da sie weder zur Erzeugung des LM noch zur

	Front	Central	Back
Close	i	r	u
Close-mid	e		o
Open	a		

Tabelle 2: Vokale

	Bilabial	Labio-Dental	Dental	Alveo-palatal	Palatal	Velar
Stops	p,b		t,d			k,g
Fricatives		f,v	s,z	š, ž		h
Affricates			c	č, dž	ć, đ	
Nasals	m		n		nj	
Laterals			l		lj	
Vibrant			r			
Glide					j	

Tabelle 3: Konsonanten

Bildung des Aussprachewörterbuches verwendet wurden. Dies führt zu hohen, sogenannten *Out-Of-Vocabulary-Raten (OOV-Raten)*, die höhere Fehlerraten mit sich führen. Dies alles erschwert die Aufgabe einen Erkenner mit guten Ergebnissen bei kurzen Antwortzeiten zu erstellen, beträchtlich.

3.3 Lauteigenschaften

Im Zuge der erwünschten Vereinigung der Südslawen in den Zwanziger Jahren des letzten Jahrhunderts, wurde die lateinische Schrift eingeführt. Um auch mit dieser Schrift alle Laute darstellen zu können, wurden digraphische: *lj*, *nj* und *dž* und diakritische Zeichen: *ć*, *č*, *š*, *ž* und *đ* eingeführt. Die durch die Sonderzeichen dargestellten Laute entstanden durch die sogenannten *Palatalisierungen* von Konsonanten. Als Palatalisierung bezeichnet man die Verschmelzung des j-Lautes der *jotiert* gesprochenen *hellen* Vokale der slawischen Sprache mit den vorangehenden Konsonanten. Dies führte zu den Lauten [nj] und [lj]. Aus manchen jotierten Konsonanten entstanden später die Zischlaute. Aus [kj] wurde so der Laut [$t\int$].

Um ein Beschreibung der Laute der verschiedenen Sprachen bemüht sich die *International Phonetic Association (IPA)*. Sie ordnet die Laute, die ein Mensch äußern kann nach ihren Eigenschaften. Dabei achtet die IPA auf die Art wie die Laute gebildet werden und welche Teile des spracherzeugenden System des Menschen maßgeblich an ihrer Bildung beteiligt sind. Die Tabellen 2 und 3 zeigen eine erschöpfende Auflistung aller Konsonanten und Vokale der kroatischen Sprache. Beim Kroatischen ist aber zu beachten, daß der Konsonant r silbisch genutzt werden kann. Dies geschieht im Allgemeinen dann, wenn r von zwei Konsonanten umgeben ist. Für eine genaue Erläuterung siehe [Brow93].

Eine, für die Spracherkennung angenehme, Eigenschaft der kroatischen Sprache ist die Möglichkeit einer einfachen Identifizierung der Laute. Die Worte werden so gesprochen wie sie geschrieben werden. Es gibt keine Veränderung der Aussprache eines Buchstaben durch Veränderung seines Kontextes. Es ist für den Sprecher und damit für den Spracherkenner egal an welcher Stelle des Wortes der auszusprechende bzw. zu erkennende Buchstabe steht. Diese Eigenschaft macht einen sogenannten *Graphem-to-Phonem*-Ansatz möglich. Jeder Buchstabe in jedem Wort kann einfach von links nach rechts durch ein entsprechendes Phonem ersetzt werden. Dies ist zum Beispiel im Deutschen nicht möglich. Hier müssen mehrere ähnliche Laute modelliert werden um eine korrekte Abdeckung zu erreichen. Einzige Ausnahme im Kroatischen stellt die Aussprache des Buchstaben $\check{z}$ dar. Beginnt der Stamm eines Wortes mit $\check{z}$ und endet die vorangehende Vorsilbe mit einem d, dann wird der Buchstabe anders ausgesprochen. Bis auf diese Ausnahme ist die Erzeugung von Aussprachewörterbucheinträgen konform und kann für viele Wörter automatisiert werden.

4 Erstellung des kroatischen Spracherkenners

Um einen Spracherkenner aufzubauen, ist es wichtig ein gutes Werkzeug an der Hand zu haben. Dies war mit JANUS gegeben. Außer einem geeigneten Arbeitsmittel benötigt man zur Erstellung des Erkenners eine sehr große Menge Daten.

4.1 Daten

In den folgenden Abschnitten wird auf den Aufwand und die Probleme beim Sammeln und Aufbereiten der Daten eingegangen.

4.1.1 Datensammlung und Transkribierung

Die kroatischen Sprachdaten wurden im Rahmen des Aufbaus der **GlobalPhone**-Datenbasis gesammelt. Das **GlobalPhone**-Projekt befaßt sich mit *MLVCSR* (multilingual large vocabulary continuous speech recognition). Außer Kroatisch beinhaltet diese Datenbasis acht weitere Sprachen. Die Sprachen Portugiesisch, Spanisch, Russisch, Türkisch, Koreanisch, Chinesisch, Arabisch und Japanisch runden die Datenbasis ab. Die Daten für alle Sprache wurden in den jeweiligen Ländern gesammelt. Die ungefähr 100 Sprecher pro Sprache waren im Alter zwischen 18 und 80, wobei das Geschlecht gleichmäßig verteilt war [ScWW97].

Die Sprachdaten müssen nicht nur gesammelt, sondern danach auch noch entsprechend aufbereitet werden. Dieser Schritt stellt einen erheblichen Teil der Gesamtarbeit bei der Erstellung eines Spracherkenners dar. Der Dank für diese langwierige Arbeit gebührt Sanela Habibija. Im Folgenden wird ihre Vorgehensweise bei der Aufbereitung der Daten beschrieben.

Als Datenquelle für die Texte diente das Internet. Die Seite des kroatischen Senders HRT http://hrt.com.hr/vijesti/hrt lieferte ausreichend Nachrichtentexte über die angestrebten Themen Politik und Wirtschaft. Ein großer Nachteil der Textbeschaffung über das Internet war zu diesem Zeitpunkt noch, daß die meisten Texte keine Sonderzeichen enthielten und somit weitgehend wertlos sind oder Wort für Wort überarbeitet werden müssen. Bei den verwendeten Texten handelt es sich um die Nachrichten der Monate Januar, März und Juli des Jahres 1996. Alle Textpassagen die mit dem blutigen Geschehen auf dem Balkan zu tun hatten, wurden entfernt. Diese Maßnahme schien angebracht, um keine unnötigen Gefühlsregungen oder gar eine ablehnende Haltung bei den Sprechern zu bewirken. Fast jeder auf dem Balkan hat im Krieg einen Familienangehörigen verloren.

Die Daten wurden durch vor Ort gekaufte Zeitungen ergänzt. Diese Art der Textbeschaffung hat den Nachteil, daß die Daten nicht in elektronischer Form vorliegen und eingetippt werden müssen. Die eigentlichen Aufnahmen wurden dann innerhalb von zwei Wochen in Kroatien und Bosnien gemacht. Ein großes Problem war, überhaupt Sprecher zu finden. Die angesprochenen Personen zeigten meist großes Mißtrauen oder waren einfach lustlos.

Ein Spracherkenner benötigt zu jeder Audiodatei einen entsprechenden Referenztext. Manche Sprecher lesen falsch von den Originaltexten ab oder fügen Wörter ein, die nicht geschrieben stehen. In fast allen Aufnahmen kommen Häsitationen und Geräusche wie Räuspern und Schmatzen vor. Deshalb müssen alle Aufnahmen angehört und nacheditiert werden. In Tabelle 4 findet sich eine Zusammenfassung der wichtigsten Eigenschaften der Daten.

Sprecher- und Texteigenschaften	
gesprochene Dialekte (#)	Zagrebisch (23), Bosnisch (42) Nordbosnisch (5), Kajkawisch (5), Schtokawisch (1), Purgerisch (1)
Sprecheranzahl	83
männlich Sprecher	33
weibliche Sprecher	50
Sätze	4019
gesprochene Wörter	106152
Vokabulargröße	20381
Gesamtdauer	ca. 14 Stunden
mittlere Satzlänge	12,6 Sekunden

Tabelle 4: Zahlen und Daten

4.1.2 Romanisierung der Transkriptionen

Da die kroatischen Sonderzeichen mit normalem ASCii-7-bit-Kode nicht dargestellt werden können, müssen diese auf geeignete Zeichen bzw. Zeichenketten abgebildet werden. Die Motivation bei der Auswahl einer geeigneten *Romanisierung* war, so nahe wie möglich an der Aussprache der Laute zu bleiben. Dadurch werden die romanisierten Texte lesbarer. Die Sonderzeichen wurden durch die in Tabelle 5 dargestellten Zeichenketten ersetzt. In der Tabelle ist auch jeweils ein Beispiel angegeben, um die teilweise doch sehr ähnlichen Aussprachen der jeweiligen Laute zu verdeutlichen.

Laut	Transkribtion	Beispiel
ć	dscH	Dschungel
č	tscH	Deutsch
š	sscH	Schule
ž	jscH	Jean
dž	djscH	Budget
đ	djcH	Jeep
nj	nj	españa
lj	lj	Ilja

Tabelle 5: Romanisierung

4.2 Abbildung auf Phoneme

Die kroatischen Laute sind, wie in Abschnitt 3.3 beschrieben, einfach zu identifizieren. Selbst die einzige Ausnahme beschränkt sich auf nur einige Fälle. Als Phonemliste wurde deshalb das kroatische Alphabet gewählt. So wird *a* auf *KR_a* abgebildet, *b* auf *KR_b*, usw. Die Sonderzeichen wurden durch geeignete Phoneme ersetzt. In Tabelle 6 sind die Abbildungen der Sonderzeichen auf die Phoneme dargestellt.

h	KR_x
ć	KR_tj
č	KR_tS
š	KR_S
ž	KR_Z
dž	KR_dS
đ	KR_dj
nj	KR_nj
lj	KR_L
predž	KR_p KR_r KR_e KR_d KR_Z
nadž	KR_n KR_a KR_d KR_Z
podž	KR_p KR_o KR_d KR_Z

Tabelle 6: Abbildung auf Phoneme

4.3 JANUS Datenbasis

Eine JANUS Datenbasis besteht aus je drei Teilen für die Sprecher und für die gesprochenen Sätze. Diese werden in separaten Dateien gespeichert. Die Datendatei und die Textdatei für die Sprecher enthalten die Informationen über jeden Sprecher in verschiedenen Formaten. Dort sind die Kennung, das Geschlecht, die Sprache mit Dialekt und welche Sätze gesprochen wurden, aufgelistet. Die Dateien für die Sätze enthalten die Satzkennung, die Länge der Audiodatei, sowie den Textinhalt. Für Sprecher und Sätze existiert jeweils eine Indexdatei. Die Erstellung geschieht über das Werkzeug *mapper* aus dem entsprechenden Textkorpus.

Die Datenbasis wurde vor ihrer Verwendung einer Überprüfung unterzogen. Das Ziel war Fehler wie Verweise auf falsche Audiodateien, beschädigte Audiodateien oder fehlerhafte Transkriptionen aufzuspüren. Diese Fehler können im weiteren Verlauf zu einer Verschlechterung der Erkennungsleistung bzw. zu einer Reduzierung der Trainingsmenge führen. Um diesen Fehlern zu begegnen, wurde das Skript `checker` erstellt. Dieses Skript versucht, mit sich erhöhenden Beamwerten, Viterbipfade durch die Trainingsdaten zu legen. Kann für einen Satz beim höchsten Beam kein Pfad erstellt werden, dann muß dieser Datenbasiseintrag kontrolliert und verbessert bzw. ungünstigstenfalles gelöscht werden.

4.4 Aussprachewörterbuch

4.4.1 Das Tool `erstelltAWB`

Die Basis für ein Aussprachewörterbuch ist ein vollständiges Vokabular. Dieses muß alle unterschiedlichen, in den Trainingsdaten vorkommenden Wörter, enthalten. Das Wörterbuch kann aber auch durch andere Quellen ergänzt werden, solange diese Quellen nicht die Testdaten sind.

Die herkömmliche Methode zur Erstellung eines Aussprachewörterbuchs war, die Einträge von Hand vorzunehmen. Dies war möglich, da die Größe der Wörterbücher sich um die 2000 Einträge bewegte. Da mehrere verschiedene Personen die Einträge vornahmen, kam es zu Inkonsistenzen. Um diese Problem zu beheben, schrieb **Tilo Sloboda** das `pearl`-Skript `make-dict`. Diese Skript liest ein bestehendes Wörterbuch und eine zu überprüfende Wortliste ein. Sind Wörter noch nicht enthalten, versucht es die neuen Wörter durch Kombination und Veränderung von bereits eingetragenen Wörtern zu bilden. Außerdem kann es Korrekturen durchführen und auf Fehler hinweisen. Ein Ansatz, der im Deutschen Sinn macht und gute Ergebnisse erzielt, aber für andere Sprachen nicht funktionieren muß.

Um mit `make-dict` ein kroatisches Wörterbuch auf Basis des Graphem-to-Phonem Ansatzes erstellen zu können, wurde es erweitert. Das Skript erstellt jetzt zusätzlich bei Eingabe einer *Graph-to-Phon*-Datei aus einem Wort des Vokabulars die entsprechende Phonemkette. In der Graph-to-Phon-Datei sind die Abbildungen der Buchstaben auf die Phoneme gespeichert.

4.4.2 Ablauf von `erstelltAWB`

Das Skript `make-dict` kann nicht mit allen Einträgen des Vokabulars umgehen. So werden z.B. Folgen von Ziffern oder Großbuchstaben nicht durch die entsprechende Aussprache ersetzt. Die Phonemfolgen für diese Wörter mußten bisher von Hand in das Wörterbuch eingetragen werden. Um jetzt diese automatisch abzudecken, wurden die Ausgabedateien von `make-dict` weiterverarbeitet. Zur Bewältigung dieser Aufgabe wurden mehrere neue tcl-Routinen erstellt und in dem Skript `erstelltAWB` zusammengefaßt. Der Ablauf und Aufbau des Skripts wird in Abschnitt 6 detailliert erläutert. Hier folgt nur ein kurzer Abriß der Funktionsweise.

Die grundlegende Arbeit wird von `make-dict` erledigt. Der Aufruf des *pearl*-Skriptes erfolgt aus `erstelltAWB` heraus. `Make-dict` liefert als Ausgabe zwei Dateien. Die erste beinhaltet alle bearbeiteten Wörter mit ihren Phonemfolgen. Die zweite Datei listet alle Wörter auf, für die das *pearl*-Skript keine Phonemfolge erstellen konnte. Diese Restedatei wird weiterverarbeitet.

Als erstes werden *Ziffernfolgen*, die in der Restedatei stehen, betrachtet. Die Prozedur `zahlen` liefert für jede Zahl bis 9999 die entsprechende Phonemfolge. Zusätzlich werden noch bis zu drei Aussprachevariationen der Zahl dem Aussprachewörterbuch beigefügt. Die Aussprachevarianten sind notwendig, da für die Zahlwörter im Kroatischen mehrere gleichbedeutende Worte existieren. Die Ausgabe erfolgt wieder in zwei Dateien, wobei die Datei mit den nicht abgebildeten Worten weiterverarbeitet wird.

Bei den Folgen von Großbuchstaben handelt es sich zumeist um sogenannte Akronyme, also Wörter die buchstabiert werden müssen. Beispiele für solche Folgen sind *ADAC*, *IBM* und *HP*. Die Prozedur `buchstaben` bildet die Zeichenfolgen auf eine Folge von Phonemen ab, die eine Buchstabierung des Wortes darstellen. Folgen von Großbuchstaben die zusammenhängend gesprochen werden, müssen von Hand nachgetragen werden. *IFOR* und *NASA* sind Beispiele hierfür.

4.5 Language Model

Das Language Model wurde mit dem Tool `ngrammodel` erstellt. Dieses Programm wurde von **Klaus Ries** implementiert. Für eine genaue Beschreibung siehe [RiSG96].

Es liefert bei der Eingabe eines Textkorpus und des zugehörigen Vokabulars ein Trigramm-LM. Außerdem beinhaltet das LM verschiedene Möglichkeiten zur Berechnung der *Backoff*-Faktoren zu Bi- und Unigrammen.

Man kann dem Tool einen zweiten Textkorpus angeben. Bezüglich zu diesem berechnet es dann die *Out-of-Vocabulary-Rate (OOV-Rate)* und die prozentuale Abdeckung der Tri-, Bi- und Unigramme. Zusätzlich wird die Trigrammperplexität berechnet. Die Perplexität ist ein Wert für den mittleren Verzweigungsgrad eines Language Models. Je größer die Perplexität, desto mehr mögliche Nachfolgeworte kommen im Mittel in Frage und desto schwieriger die Erkennungsaufgabe.

4.6 Training von unterschiedlich initialisierten Systemen

Beim Bootstrapping eines Erkenners werden die Kodebuchvektoren sowie die Verteilungen mit geeigneten Vektoren initialisiert. Das naheliegende Vorgehen bei der Erstellung eines Spracherkenners ist, die Verteilungen eines bereits trainierten Erkenners zu kopieren. Der Zugriff auf einen geeigneten Erkenner einer anderen Sprache ist allerdings nicht unbedingt gegeben. In diesem Fall muß der Erkenner mit entworfenen oder zufällig erstellten Vektoren initialisiert werden. Die nachfolgenden Abschnitte zeigen vier der Methoden, die angewandt werden können und welche Ergebnisse sie erzielen. Ein Augenmerk soll auch darauf gelegt werden, ob sich der Mehraufwand einer speziellen Initialisierung im Endeffekt lohnt.

Die eigentliche Initialisierung geschieht über eine *mapping*-Datei. In dieser werden alle kroatischen Phoneme auf entsprechende Phoneme anderer Sprachen abgebildet. Ein Initialisierungsskript kopiert die Gewichtsdateien des bestehenden Erkenners in die des neuen. Eine Ausnahme bildet hier der zufällige initialisierte Erkenner. Das unterschiedliche mapping wird in den einzelnen Abschnitten besprochen und ist in Tabelle 7 zusammengefaßt. Die Ergebnisse der unterschiedlich initialisierten Systeme sind in Tabelle 11 dargestellt.

4.6.1 Blick über die verwendeten Daten

Bei den Erkennern dieses Abschnitts wurden nicht alle verfügbaren Daten verwendet. Die Systeme mit den verschiedenen Initialisierungen sollten möglichst schnell trainier- und testbar sein. Die reduzierte Datenmenge liefert in kurzer Zeit befriedigende Ergebnisse, die für eine relative Bewertung verwendet werden können. Das beste System wurde dann mit der vollen Trainingsmenge erneut trainiert.

Die Datenmenge für den ersten Trainingslauf bestand aus 31 Sprechern. Die Entwicklungs- und die Evaluationsmenge beinhalteten jeweils vier Sprecher. Aus den Daten

kroatisch	multi	deutsch	einheitlich
KR_a	DE_A		DE_+QK
KR_b	DE_B		DE_+QK
KR_c	DE_TS		DE_+QK
KR_d	DE_D		DE_+QK
KR_dj	EN_JH	DE_SCH	DE_+QK
KR_dS	EN_JH	DE_SCH	DE_+QK
KR_e	DE_EH		DE_+QK
KR_f	DE_F		DE_+QK
KR_g	DE_G		DE_+QK
KR_x	DE_H		DE_+QK
KR_i	DE_I		DE_+QK
KR_j	DE_J		DE_+QK
KR_k	DE_K		DE_+QK
KR_l	DE_L		DE_+QK
KR_L	DE_J		DE_+QK
KR_m	DE_M		DE_+QK
KR_n	DE_N		DE_+QK
KR_nj	SP_N5	DE_J	DE_+QK
KR_o	DE_O		DE_+QK
KR_p	DE_P		DE_+QK
KR_r	DE_R		DE_+QK
KR_s	DE_S		DE_+QK
KR_S	DE_SCH		DE_+QK
KR_t	DE_T		DE_+QK
KR_tj	EN_CH	DE_TSCH	DE_+QK
KR_tS	EN_CH	DE_TSCH	DE_+QK
KR_u	DE_U		DE_+QK
KR_v	DE_V		DE_+QK
KR_z	DE_Z		DE_+QK
KR_Z	DE_SCH		DE_+QK

Tabelle 7: mapping

der 31 Sprecher wurde das erste LM und das erst Aussprachewörterbuch erstellt. Die Eigenschaften der Daten, des LM und des Aussprachewörterbuchs sind in den Tabellen 8 und 9 aufgelistet.

Sprecheranzahl	31
Utterances	1533
gesprochene Wörter	41955
Vokabulargröße	10374
Gesamtdauer	ca. 5,6 Stunden
mittlere Satzlänge	13,16 Sekunden
Wörterbucheinträge	10744

Tabelle 8: Trainingsmenge, erster Lauf

The perplexity is 70.196
excluding <UNK> 219.32
#words 7968 (28.93% OOV-rate)
23.66% 1-grams used
35.55% 2-grams used
40.79% 3-grams used

Tabelle 9: Language Model, erster Lauf

4.6.2 Unterschiedliche Initialisierungen

Multilinguale Initialisierung

Diese Art der Initialisierung ist nur möglich, wenn ein entsprechender multilingualer Spracherkenner oder mehrere einsprachige Systeme zur Verfügung stehen. In diesem Falle war ein multilingualer Erkenner vorhanden. Das System enthält Phonemsätze für die Sprachen Spanisch, Japanisch, Englisch und Deutsch.

Bei der multilingualen Initialisierungen werden aus den vier Sprachen die geeignetsten Phoneme zur Modellierung der kroatischen Phoneme ausgewählt. Dies ist nur erfolgversprechend, wenn die Abbildung vom jemandem vorgenommen wird, der die Sprachen gut beherrscht. Der jotiert Konsonant *ñ* aus dem Spanischen dient zur Initialisierung des Phonemes KR_nj. Die Verwendung der Zischlaute aus dem Englischen verbessert die Darstellung der zahlreichen Zischlaute im Kroatischen. Die restlichen Phoneme werden durch die dem kroatische ähnlichen Laute der deutschen Sprache initialisiert.

Der Erkenner mit allen Lauten aus den vier Sprachen bietet eine gute Ausgangsbasis zur Initialisierung eines neuen Erkenners. Leider ist ein verwendbares, bereits trainiertes System nicht immer gegeben. In diesen Fällen muß eine alternative Methode gewählt werden.

Einsprachliche Initialisierung

Die Verwendung nur einer Sprache ist eine Vereinfachung der multilingualen Initialisierung. Hier werden die kroatischen Gewichte aus den trainierten Kodebüchern einer Sprache kopiert. Die deutsche Sprache wurde gewählt, weil für viele Laute im Kroatischen ein gleicher deutscher Laut existiert. Zusätzlich stellt das deutsche System zwei verschiedene Zischlaute zur Verfügung.

Diese Initialisierungsmethode bietet eine geringere Zahl an Phonemen zur Auswahl. Sie setzt aber auch nicht voraus, daß zahlreiche bereits trainierte Erkenner zur Verfügung stehen.

Einheitlich Initialisierung

Bei der Einheitlichen Initialisierung werden alle neuen Phoneme mit den Werten eines Kodebuchs des bestehenden Erkenners initialisiert. Diese Methode erspart die Auswahl von geeigneten Phonemen und damit das Wissen über die genaue Aussprache. Der entscheidende Nachteil ist, daß man ein initial schlechteres System zu erwarten hat, als bei den beiden bereits beschriebenen Methoden. In diesem konkreten Fall wurde zur Initialisierung alles kroatischen Kodebücher das deutsche "Müll"-Phonem *DE_+QK* gewählt. Diese Phonem wird als Modellierung aller Wortfragmente verwendet. Es enthält also Eigenschaften und Eigenheiten aller Phoneme.

Zufällige Initialisierung

Steht kein System zur Verfügung, das kopierbare Initialisierungen liefert, müssen geeignete Vektoren erzeugt werden. Dies geschah hier über ein tcl-Skript, daß Zufallszahlen erstellt und im entsprechenden Format abspeichert, so daß die Gewichte einfach von JANUS eingelesen werden können. Der Bereich der Zufallszahlen wurde durch die kleinste und die größte Zahl, die in den Gewichten für die deutschen Phoneme vorkommen, abgesteckt. Bei diesem System konnte die Hälfte der initialen Labels nicht geschrieben werden. Trotz Erhöhung des Beams war bei diesen Sätzen eine Rückverfolgung der Viterbipfade nicht möglich.

4.6.3 Training

Das Training für alle vier verschiedenen Initialisierungen verlief identisch. Der einzige Unterschied bestand in den verschiedenen *mapping*-Dateien. Das Training besteht aus folgenden Schritten. Für jeden der Punkte ist bereits ein gleichnamiges Skript in der *Janus-Script-Collection* vorgesehen. Eine genaue Beschreibung der folgenden Abläufe findet sich in [Rogi96, Rogi97].

- init
  Mit diesem Skript werden die Gewichte des neuen Erkenners initialisiert. Es benötigt die mapping-Datei als Eingabe.
- label
  In den Label-Dateien werden Viterbipfade für alle Trainingsdaten gespeichert. Dies beschleunigt das Training erheblich. Die Pfade verändern sich während des Trainings. Die Label müssen deshalb nach einigen Trainingsiterationen erneuert werden.
- LDA
  Die LDA erleichtert das Klassifizieren der Merkmalsvektoren, indem sie den Merkmalsraum reduziert.
- samples
  Da es zu aufwendig ist, alle vorhandenen Vektoren zu betrachten, werden mit diesem Skript "samples" aus der Datenbasis extrahiert.
- kmeans
  Der k-means Algorithmus ist ein Clusterverfahren, mit dem die Stichprobenmenge in geeignete Klassen eingeteilt wird.
- train
  Diese Skript paßt durch ein Viterbi-Training die gaußschen Mischverteilungen der Kodebücher an.

Das Training verläuft über fünf Iterationen. Die label-Dateien und die LDA-Matrix werden dann mit dem trainierten System neu erstellt und der Zyklus beginnt von vorne. Beim zweiten Durchlauf wird ebenfalls fünf Iterationen lang trainiert.

4.6.4 Bestimmung geeigneter Testparameter

Vor den eigentlichen Testläufen müssen die Parameter des Language Models sowie die Einstellungen für das Pruning ermittelt werden. Die Werte für *Beam* und *topN* schränken die Zahl der gleichzeitig verfolgten Hypothesen ein.

- topN
  Mit diesem Parameter wird die Anzahl der gleichzeitig verfolgten möglichen Lösungen eingeschränkt. *topN* gibt an, wieviele Verzweigungen zu Folgeworten von jedem Wort maximal ausgehen dürfen. Alle diese Möglichkeiten bilden dann die betrachteten Hypothesen.

- Beam
  Dieser Wert dient zur Berechnung des Scores, bei dessen Überschreiten eine Hypothese nicht weiterverfolgt wird. Liegt der Score einer Hypothese über der Summe des *Beam*-Wertes und des bisher erreichten Scoreminimums, wird die Hypothese verworfen.

- lp
  der Parameter lp steht für die Wortübergangsstrafe. Je höher sein Wert gesetzt wird, desto stärker werden Wortübergänge bestraft. Mit diesem Wert kann das Verhältnis zwischen *Einfügungs-* und *Auslassungsfehlern* in den Hypothesen beeinflußt werden.

- lz
  Der Wert von lz bestimmt, mit welcher Gewichtung das LM in die Berechnung des Score eingeht.

Diese Parameter werden anhand von wiederholten Testläufen auf der Entwicklungsmenge experimentell bestimmt. Bei der Bestimmung der geeignetsten Parameter darf nicht nur der absolute Wert der Fehlerrate beachtete werden, sondern auch das Verhältnis zwischen der Anzahl der Einfügungen und Auslassungen einzelner Worte. Diese beiden Fehlerarten sollten sich im Gleichgewicht befinden. Die mit Hilfe des multilingualen Systems ermittelten Parameter wurden dann für alle Systeme beibehalten, um die Vergleiche nicht zu verfälschen. Für *lz* und *lp* wird je ein Parametersatz angegeben, mit denen dann beim *Lattice-Rescoring* eine Matrix von Ergebnissen berechnet wird. Tabelle 10 zeigt die endgültig verwendeten Parameter.

beam	100					
topN	100					
lp	-6	-4	-2	0	2	4
lz	20	22	26	28	30	32

Tabelle 10: verwendete Parameter

4.6.5 Ergebnisse der Testläufe

Nachdem die optimalen Werte für die Parameter auf den Daten der Entwicklungsmenge ermittelt waren, erfolgten die eigentlichen Tests auf der Testmenge. Tabelle 11 zeigt jeweils den besten Wert, den die entsprechenden Testläufe ergaben. Die Tests der Erkennungsleistung wurden zu drei verschiedenen Zeitpunkten des Trainings durchgeführt. Jedes System wurde direkt nach der Initialisierung (S1/0i), nach der fünften Iteration des ersten Durchlaufs (S1/5i) und nach der fünften Iteration des zweiten Durchlaufs (S2/5i) getestet.

	multi	deutsch	gleich	zufällig
S1/0i	34,2	33,3	3,8	3,2
S1/5i	36,7	34,2	4,8	3,5
S2/5i	37,2	36,8	24,9	14,3
S3/5i	n/a	n/a	34,4	24,0
S4/51	n/a	n/a	36,5	29,0

Tabelle 11: Ergebnisse - erster Lauf - Worterkennungsrate in %

Das multilinguale System erreichte die besten Erkennungsraten. Das deutsche System liegt aber jeweils nur knapp unter diesen Ergebnissen. Die Werte liegen so nahe beieinander, da 25 der 30 Kodebücher beim multilingualen System ebenfalls mit Gewichten des deutschen Erkenners initialisiert wurden. Die besseren Ergebnisse des multilingualen führen trotz der geringen Unterschiede zum deutschen System, zu der Folgerung, daß die zusätzlich verwendeten spanischen und englischen Gewichte besser für eine Initialisierung geeignet sind. Die sich unterscheidenden Zischlaute und die Verwendung des jotierten Lautes haben Auswirkungen auf die Erkennungsleistung um 0,4% bis 2,5%.

Die anfänglich so schlechten Werte des zufällig- und des gleich initialisierten Systems sind darauf zurückzuführen, daß die Hälfte der Daten nicht für das Training zur Verfügung standen, da für diese keine Labeldateien geschrieben werden konnten. Zusätzlich müssen die zu Beginn uniformen Kodebücher erst durch Training an die Merkmalvektoren angepaßt werden. Bei dem multilingualen und dem deutschen System ist dies schon weitgehend durch die Initialisierung geschehen.

Ein Test hat gezeigt, daß bereits nach der zweiten Iteration die Gewichte ausreichend angepaßt sind, um die Viterbipfade für alle Trainingsdaten zu erstellen. Tabelle 11 zeigt, daß das gleich initialisierte System die Erkennungsleistung des multilingualen und des deutsch initialisierten Sytems erreicht, wenn man einen höheren Trainingsaufwand betreibt.

4.7 Vergrößerung der Trainingsmenge

4.7.1 Unterschiede zur bisherigen Trainingsmenge

Bei diesem System wurde die gesamte vorhandene Datenmenge verwendet. Das Trainingsmaterial wurde auf 63 Sprecher ausgedehnt. Die jetzt noch fehlenden restlichen 12 Sprecher werden für zukünftige Experimente reserviert. Sie sollen dann die beiden Testmengen erweitern. Da das hier trainierte System auch mit diesen erweiterten Testmengen verwendet werden soll, können diese 12 Sprecher nicht in die Trainingsmenge aufgenommen werden.

Tabelle 12 stellt die Eigenschaften der größeren Datenmenge dar. Die Daten haben sich fast verdoppelt. Besonders zu beachten ist, daß dies auch für die Vokabulargröße gilt.

Die erweiterten Trainingsdaten wurden auch dazu verwendet, ein neues Aussprachewörterbuch und ein neues Language Model zu erstellen. Die Eigenschaften des zweiten LM sind in der Tabelle 13 veranschaulicht. Das LM konnte nicht deutlich verbessert werden. Die Trigrammabdeckung ist ungenügend und der Wert der Perplexität ist viel zu hoch. Die einzige Verbesserung ist die Senkung der OOV-Rate.

Der im folgenden beschriebene Spracherkenner basiert auf dem multilingual initialisierten System des letzten Abschnitts. Das Vorgehen beim Training im zweiten Lauf ist identisch mit dem ersten Lauf. Keine Einstellungen in den Skripten wurde verändert. Die Parameter der Testläufe wurden ebenfalls gleich belassen.

Sprecheranzahl	63
Utterances	2906
gesprochene Wörter	76079
Vokabulargröße	17120
Gesamtdauer	ca. 10,25 Stunden
mittlere Satzlänge	12,13 Sekunden
Wörterbucheinträge	17647

Tabelle 12: Trainingsmenge - zweiter Lauf

4.7.2 Ergebnisse

Die im zweiten Lauf erzielten Ergebnisse sind nur geringfügig besser, als die Werte, die der erste Lauf geliefert hat. Der zweite Trainingsdurchgang des zweiten Laufs

The perplexity is 822.68
excluding <UNK> 387.93
#words 31194 (12.93% OOV-rate)
59.54% 1-grams used
24.47% 2-grams used
15.99% 3-grams used

Tabelle 13: Language Model, zweiter Lauf

S1/0i	31,6
S1/5i	36,5
S2/5i	37,9

Tabelle 14: Ergebnisse - zweiter Lauf - Worterkennungsrate in %

liefert nach der fünften Iteration (S2/5i) ein, nur um 0,6% besseres, Ergebnis, als der vergleichbare Durchgang im ersten Lauf.

Ein Grund für die geringfügige Verbesserung der Erkennungsleistung bei Verdoppelung der Trainingsmenge ist das proportional zum Gesamttext wachsende Vokabular. Die Trigrammabdeckung des neuen LM ist um einiges schlechter als als beim ersten Lauf (Tabelle 9). Die Testdatenmengen enthalten immer noch zahlreiche Wörter, die weder im LM noch im Aussprachewörterbuch auftauchen. Die Modellierung von *<UNK>* ist nicht ausreichend gut, um die hohe OOV-Rate zu kompensieren.

Bisher wurden für das LM alle Wörter des Vokabulars verwendet. Um eine bessere Modellierung für die unbekannten Wörter zu bekommen, kann man das LM mit einem reduzierten Vokabular erstellen. Dies erhöht die OOV-Rate. Um dies zu erreichen, werden nur Wörter des Vokabulars verwendet, die mindestens x-mal vorkommen. In Abbildung 2 sind Vokabulargröße und OOV-Rate in Abhängigkeit von verschiedenen Werten für x aufgetragen.

Leider ist diese Methode für die kroatische Sprache nicht anwendbar da die OOV-Rate zu sprunghaft anwächst. Die Eigenheiten der Sprache, die das starke Vokabularwachstum bewirken, führen dazu, daß bei einer Textkorpusgröße von ungefähr 30000 Wörtern ungefähr 18000 Worte nur einmal vorkommen. Die OOV-Rate erhöht sich schon beim Schritt von 1 auf 2 so stark, daß, selbst mit der verbesserten Modellierung der unbekannten Worte, kein leistungsstärkeres Language Model zu erwarten ist.

Um das LM zu verbessern, ist eine deutliche Erhöhung der Datenmenge nötig. Die

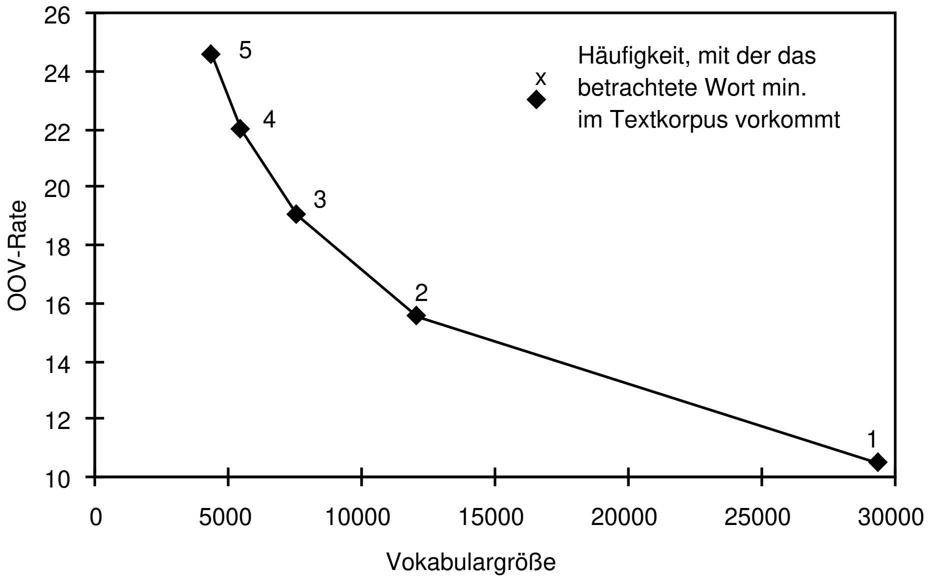

Abbildung 2: Verhältnis Vokabulargröße - OOV-Rate

größere Datenmenge ermöglicht dann eine bessere ngram-Schätzung. Bei der Besprechung des kontextabhängigen Spracherkenners in Abschnitt 5 wird dieser Ansatz beschrieben.

5 Kontextabhängiger Erkenner

Das in Abschnitt 4.7 beschriebene System dient als Grundlage für den kontextabhängigen Erkenner. Das Language Model und Ausspracheworterbuch wurden, für einen Teil der Experimente, übernommen.

5.1 Warum eine kontextabhängige Modellierung?

Wird bei der Modellierung der Akustik ein kontextunabhängiger Ansatz gewählt, werden viele Laute auf wenige Phoneme abgebildet. Dies führt dazu, daß unterschiedliche Laute mit einer ähnlichen Aussprache in die gleichen Modelle trainiert werden.

Um eine schärfere Trennung zu erreichen, wählt man eine kontextabhängige Modellierung der Akustik. Dabei ist darauf zu achten, daß die gewählten Spracheinheiten sowohl akustisch präzise als auch statistisch robust sind. Die erfordert eine geeignete Wahl von Anzahl und Art der Spracheinheiten.

5.2 Polyphone

Die Bezeichnung kontextabhängig für den Erkenner, rührt von der kontextabhängigen Modellierung der Phoneme her. Die sogenannten *Polyphone* werden abhängig von ihren Nachbarphonemen modelliert, um die Koartikulation zu berücksichtigen. Der Kontext kann theoretisch auf beliebig viele Buchstaben vor oder nach dem Phonem ausgedehnt werden. Einzige Einschränkung bei JANUS ist, daß der Kontext nur einen Buchstaben des nächsten Wortes betreffen darf. Ein zu hoher Kontext liefert allerdings zu viele Polyphone, die den Erkenner zu groß werden lassen. Außerdem stehen oft nicht genügend Trainingsdaten zur Verfügung, um die resultierenden, zahlreichen Modelle zu trainieren.

Um die im Trainingskorpus enthaltenen Polyphone zu ermitteln, werden die entsprechenden Transkriptionen durchlaufen und alle gefunden Polyphone des gewählten Kontextes gespeichert. Als Datenstruktur zur Speicherung der Polyphone dient ein Entscheidungsbaum. Die so erhaltene Datenmenge wird durch phonetischen Fragen nach dem Kontext geclustert.

5.3 Voreinstellungen

Eine der ersten Überlegungen ist die Wahl der Kontextgröße. Für dieses System wurde ein Kontext von jeweils zwei vor und nach dem Phonem gewählt. Dies führt auf den Trainingsdaten zu 184995 *Quintphonen*. Bei der kontextabhängigen Modellierung entstehen offensichtlich viel mehr Kodebücher und Verteilungen als bei dem kontextunabhängigen System. Deshalb erfolgt die Initialisierung der Polyphone, unbeachtet ihres Kontextes, durch die entsprechenden Phoneme aus dem kontextunabhängigen System. Die Verteilungen werden zuerst durch einige Trainingsiterationen angepaßt, bevor das eigentliche Training beginnt. Die Wahl der Parameter ist in Tabelle 15 dargestellt.

linker Kontext	2
rechter Kontext	2
#Quintphone	184995
Modelle	2000

Tabelle 15: Parameter des kontextabhängigen Erkenners

5.4 Trainingsläufe

Das Training verläuft nach dem gleichen Prinzip wie in Abschnitt 4.6.3. Die label-Dateien des kontextunabhängigen Systems wurden, während den drei Trainingsläufen zur Anpassung der neuen Verteilungen, beibehalten. Vor dem eigentlichen Training wurden sie neu erstellt. Das Training lief über 10 Iterationen, wobei nach der fünften und nach der letzten Iteration Testläufe gestartet wurden.

5.5 Testläufe

Der Ablauf der Testläufe folgt dem gleichen Schema wie in Abschnitt 4.6.4. Die Parameter für das kontextunabhängige System wurden neu ermittelt. Sie sind in Tabelle 16 dargestellt.

beam	100				
topN	100				
lp	-14	-12	-10	-8	-6
lz	26	28	30	32	34

Tabelle 16: Testparameter

Zur Ergänzung der Testläufe wurden Tests mit einer OOV-Rate von 0% durchgeführt. Zu diesem Zweck mußten das Language Model und das Aussprachewörterbuch angepaßt werden. Dem Wörterbuch wurden alle Wörter aus den Testsets hinzugefügt. Das Language Model erhielt die Information über die hinzugefügten Wörter in Form von Unigrammen. Das Hinzufügen von Bi- und Trigrammen hätte zu einem zu weitreichendem Wissen des Erkenners über die Testdaten geführt.

5.6 Besprechung der Ergebnisse

Die Erkennungsleistung des kontextabhängigen Erkenners ist in Tabelle 17 zusammengefaßt. Bei dem System, dessen Ergebnisse in der Spalte "Artefakte" aufgeführt sind, wurden Worte, die bis auf die Groß-/Kleinschreibung identisch sind, als Synonyme eingesetzt. Zusätzlich wurden noch einige Rechtschreibfehler und ähnliche Vorkommnisse, die zu Synonymen führen hier beachtet.

	normal	OOV-Rate 0%	Artefakte
S1/5i	45,6	54,3	n/a
S2/5i	47,9	56,4	62,6

Tabelle 17: Worterkennungsrate des kontextabhängigen Erkenners (in %)

```
REF1: izvanrednom SKUPSSCHTINSKOM SJEDNICOM I  konsenzusom
      dosschlo BI   DO zajed nitschkog PISMA VIJEDSCHU ...
HYP1: izvanrednom SKUPSSCHTINSKOG SJEDNICA  NI konsenzusom
      dosschlo BITI ** zajed nitschkog PISMO VIJEDSCHE ...

REF2: ... vernoatlanske SKUPSSCHTINE koje je vodio predsjednik
      SKUPSSCHTINE I tschlan njem ...
HYP2: ... vernoatlanske SKUPSSCHTINA koje je vodio predsjednik
      SKUPSSCHTINI * tschlan njem ...

REF3: ... za POTPUNU provedbu washingtonskih I DAYTONSKIH
      sporazuma I  RIMSKOG DOGOVORA
HYP3: ... za POTPUNO provedbu washingtonskih * DAYTONSKOG
      sporazuma JE MINSKIH DOGOVOROM
```

Tabelle 18: Hypothesen - Referenz - Paare

Die Ergebnisse erscheinen in einem etwas besseren Licht, wenn man die, vom Erkenner erstellten Hypothesen direkt mit den tatsächlichen Äußerungen vergleicht. Im folgenden wird das System S2 mit der OOV-Rate von 0% betrachtet. In Tabelle 18 sind repräsentative Ausschnitte einiger Paare dargestellt. Die zusammengehörenden Wörter in den Hypothesen-Referenz-Paaren sind aufeinander ausgerichtet. Die Wörter in Blockschrift wurden falsch erkannt. Befinden sich in der Referenz "*", so steht an der entsprechenden Stelle in der Hypothese ein Wort zuviel, das der Erkenner eingefügt hat. Dieselben Zeichen in der Hypothese zeigen an, daß Wörter vom Erkenner ausgelassen wurden.

Vergleicht man die vom Erkenner ausgegebenen Hypothesen genauer mit ihren Referenzen, stellt man fest, daß zahlreiche der falsch erkannten Worte den gleichen Stamm wie ihre Referenzworte aufweisen. Die Unterschiede liegen in diesen Fällen dann nur in den verschiedenen Endungen. Oft unterscheiden sich dann die "falschen" Worte von den "richtigen" nur in ein oder zwei Buchstaben. Der Inhalt der Äußerung wird trotz der Fehler richtig vermittelt. Man könnte sagen, daß der Erkenner die Sätze genauso "bildet", wie es ein Mensch tun würde, der die Zeiten und Fälle der Grammatik einer gerade erlernten Sprache verwechselt.

6 Das Skript erstelltAWB

Dieser Abschnitt stellt eine Dokumentation des Skripts **erstelltAWB** dar. Gleichzeitig soll eine Benutzerführung gegeben werden. In Tabelle 19 sind der Aufruf des Skripts und einige der Prozeduraufrufe im Skript aufgelistet. Die einzelnen Prozeduraufrufe werden im Laufe der nächsten Abschnitte erläutert.

erstelltAWB.tcl	<Vokabular> <graph2phon> <Ausgabedatei> [-special Sonderzeichen] [-vari Variationen]
make_dict.gp-alpha	[-d $graph] [-l $vocab] [-o neudict] [-c]
zahlen	<buch.rest> <zahlen.ready> <zahlen.rest>
buchstaben	<rest.lst> <buch.ready> <buch.rest>
cat	dict.ready ready.lst buch.ready zahlen.ready > dict
cat	vari.ready >> dict
make_dict.gp-alpha	[-d neudict] [-o neudict2] [-n]

Tabelle 19: Prozeduraufrufe

6.1 Ein- und Ausgabe

Der eigentlich Programmaufruf erfolgt mit den Parametern Vokabular, graph2phon und Ausgabedatei. Diese Parameter müssen dem Skript die Namen der zu verwendenden Dateien liefern. In der Datei Vokabular stehen die abzubildenden Wörter. Die Datei graph2phon liefert die Abbildungen der kroatischen Zeichenketten auf die von Janus verwendeten Phoneme. Aus Tabelle 21 ist das, für die Einträge verwendete, Format ersichtlich. Die Ausgabedatei gibt den Namen der Datei an, in die das erstellte Wörterbuch geschrieben wird.

Die letzten beiden Parameter sind optional. Sie können über die Flags -special und -vari angegeben werden. Über -special wird eine Datei mit Abbildungen angegeben, die direkt in das Wörterbuch kopiert werden. Dies geschieht hier zum Beispiel mit den von Janus verwendeten Zeichen und Bezeichnern wie **#fragment#** und SIL. Mit dem Flag -vari können dem Wörterbuch Aussprachevarianten beigefügt werden. Dies ist zum Beispiel bei Fremdwörtern, aus kroatischer Sicht, empfehlenswert. Die Einträge in der Datei Sonderzeichen müssen im, von Janus verwendeten, Format gemacht werden. Die Einträge in Variationen werden vom Programm formatiert und müssen die gleiche Formatierung wie die Vokabular-Datei haben.

Jede der von, `erstelltAWB` verwendeten Prozeduren, schreibt ihre Ausgabe in zwei Dateien. Als Eingabe dient jeweils eine Datei. Die erste Ausgabedatei beinhaltet die erfolgreich abgebildeten Wörter und die zweite den nicht bearbeiteten Rest der Eingabe. Die Eingabe jeder Prozedur ist die Restedatei der vorhergehenden Prozedur. In der Datei `Ausgabedatei` stehen am Ende alle abgebildeten Wörter und die Einträge der `Sonderzeichen`- und `Variationen`-Dateien. Zusätzlich werden in einer Restedatei die Wörter zusammengefaßt, für die in keiner der beschriebenen Vorgänge eine Ersetzung gefunden werden konnte.

6.2 Darstellung einfacher Wörter

Wie bereits erwähnt, wird zur Darstellung *normaler* Wörter ein schon vorhandenes pearl-Skript verwendet. Dem Skript `make-dict` wird das `Vokabular` und die `graph2phon`-Datei übergeben. Das Flag `-d` bewirkt, daß die mit übergebene Datei als vorgegebenes Wörterbuch betrachtet wird. In ihr wird die verwendete Graph-to-Phon-Datei und Wörter, die am Stück ersetzt werden sollen, übergeben. Mit dem Flag `-l` wird die Datei, die die abzubildenden Wörter enthält gekennzeichnet. Die Ausgabe des Skripts besteht aus zwei Dateien. In `neudict.clean` stehen die Wörter, die ganz in `graph2phon` stehen und entsprechend abgebildet wurden. In `neudict.problems` finden sich alle Wörter, die entweder gar nicht abgebildet werden konnten oder die das Skript aus den angegebenen Zeichenketten zusammengesetzt hat. Diese Vorschläge des Skripts können hier direkt weiterverwendet werden. Es ist nur eine Umformatierung nötig. Der verwendete Ersetzungsalgorithmus ist in Abbildung 3 dargestellt.

6.3 Ziffernfolgen

Die Datei `neudict.problems` wird, nach geeigneter Formatierung, der Prozedur `zahlen` als Eingabe übergeben. Jeder Eintrag wird mit einem regulären Ausdruck verglichen, der überprüft ob es sich um eine Ziffernfolge mit eventuell angehängter Endung handelt. Nachdem feststeht, daß das Wort eine Ziffernfolge ist, wird die Stellenanzahl festgestellt. Bisher kann das Skript Zahlen bis 9999 verarbeiten. Alle Zahlen die größer sind, müssen von Hand in das Wörterbuch eingetragen werden.

Die kroatische Sprache bildet ihre Zahlen einfach. Im Grunde werden die Zahlwörter für die einzelnen Stellen aneinandergehängt. Bei jeder Variation in den Zahlwörtern verdoppelt sich die Gesamtanzahl der Synonyme, die die Zahl beschreiben. Eine Ausnahme in der Regelmäßigkeit bilden hier die Zahlen zwischen 11 und 19, für die, ähnlich wie im Deutschen, eigene Begriffe existieren. Wenn die Anzahl der Stellen

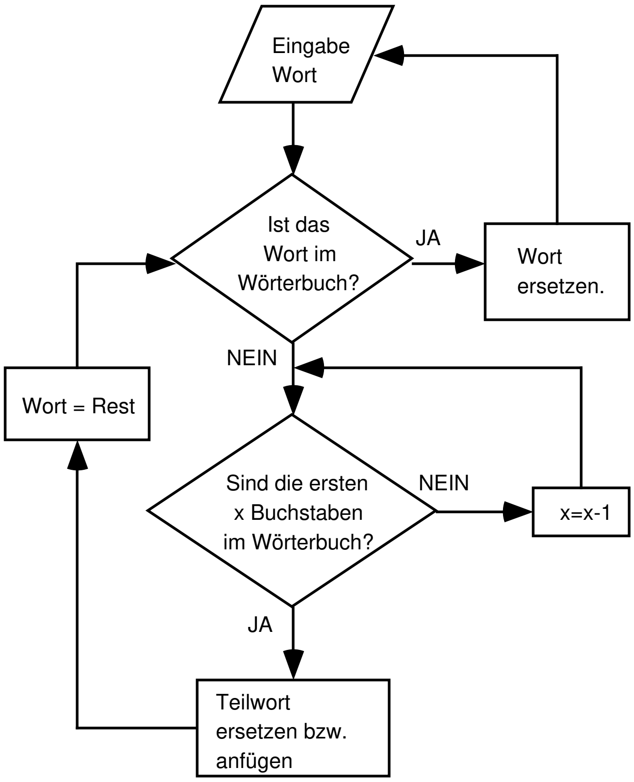

Abbildung 3: make-dict

feststeht, wird die Zahl von links nach rechts ersetzt. Für jede Variation wird eine eigene Variable mitgeführt.

Weitere Schritte müssen unternommen werden, wenn es sich um Ordnungszahlen handelt. Diese Zahlen werden einer Deklination unterzogen und sind deshalb in den Transkriptionen mit den entsprechende Endungen versehen. Außerdem ändert sich der Wortstamm der Zahlwörter für die Einerstellen. So wird aus dem Wort für "Zwei" *dva* das Wort *druge* für "die Zweite". Weitere Beispiele finden sich in Tabelle 20.

Die Zahlen sind ein gutes Beispiel für die im Kroatischen häufig vorkommenden Synonyme. Es existieren je zwei Wörter für die Zahlen, die für die Tausender stehen. Weitere Varianten entstehen dadurch, daß zwischen der Einer- und Zehnerstelle ein "i" eingefügt werden kann. Die Tabelle 20 führt die Zahl 4357 als Beispiel an. Für diese Zahl gibt es vier Varianten, die zu beachten sind. Es ist nicht wichtig, welche der Varianten tatsächlich zum Einsatz gekommen sind, da alle vier dem Aussprachewörterbuch beigefügt werden. Somit ist sichergestellt, daß die richtige Variante nachschlagbar ist.

Zahl	Varianten
4000	četiritisuće
	četirixiljade
300	tristo
50	pedeset
7	sedam
4357	četiritisućetristopedesetsedam
	četirixiljadetristopedesetsedam
	četiritisućetristopedesetisedam
	četirixiljadetristopedesetisedam
2.-og	drugog
13.-e	trinaeste

Tabelle 20: Aussprachevarianten bei Zahlen

ADAC	KR_a KR_d KR_e KR_a KR_c KR_e
HP	KR_x KR_a KR_p KR_e
IBM	KR_i KR_b KR_e KR_e KR_m
NHCR-a	KR_e KR_n KR_x KR_a KR_c KR_e KR_e KR_r KR_a
IDS-om	KR_i KR_d KR_e KR_e KR_s KR_o KR_m

Tabelle 21: Großbuchstaben

6.4 Folgen von Großbuchstaben

Weitere Eingaben, die das Skript `make-dict` nicht bearbeiten kann, sind Folgen von Großbuchstaben. Bei diesen handelt es sich zumeist um Worte die buchstabiert werden müssen. Beispiele für solche Folgen sind *ADAC*, *IBM* und *HP*. Die Prozedur `buchstaben` ersetzt die Großbuchstaben durch die entsprechende Folge von Phonemen. Dabei wird den Konsonanten der entsprechende Vokal angehängt. Beim Buchstabieren der Wörter müssen gegebenenfalls auch die Endungen beachtet werden. Allerdings müssen Folgen von Großbuchstaben die zusammenhängend ausgesprochen werden, wie z.B. *IFOR* und *NASA* von Hand nachgetragen werden. Für Beispiele zur Buchstabierung siehe Tabelle 21.

7 Ausblick und Betrachtungen

Um die Ergebnisse in das rechte Licht zu rücken, kommt man nicht umhin, sich die verwendeten Daten und die angewandten, oder vielmehr die nicht angewandten Vorgehensweisen im Rahmen dieser Studienarbeit näher zu betrachten.

Für diese Arbeit wurden alle Teile, die für einen Spracherkenner nötig sind für die kroatische Sprache neu erstellt.Zu Begin mußte ein Phonemsatz gewählt werden. Nach der Romanisierung und Erstellung der Datenbasis für JANUS folgte das Aussprachewörterbuch und das Language Model. Die Einstellungen für alle Parameter der Trainings- und Testskripten mußten angepaßt werden. Diese Abläufe bieten genügend Platz für Fehler und Auslassungen die leider nicht mit Sicherheit ausgeschlossen werden konnten, obwohl alle Skripten und Einstellungen wiederholt überprüft wurden.

Es wurde weitgehend davon ausgegangen, daß die Transkriptionen mit den Aufnahmen übereinstimmen. Eine genau Überprüfung der gesamten Datenbasis könnte unter Umständen Fehler aufdecken. Fehler in der Datenbasis ziehen sich durch das ganze System, da ein nicht korrektes Aufnahme-Transkription-Paar in jedem Falle zu Erkennungsfehlern führt.

Das Aussprachewörterbuch enthielt ausreichend viele Einträge. Allerdings kann nicht davon ausgegangen werden, daß alle Einträge frei von Schreibfehlern sind. Um diese Fehlerquelle auszuschließen, muß es von einem Muttersprachler noch überprüft werden. Außerdem sind noch einige Aussprachevarianten nachzutragen.

Speziell für die Erstellung des Language Models sind sehr große Datenmengen nötig. Diese standen für diese Arbeit nicht zur Verfügung. Ein neues Language Model auf einer umfangreicheren Datenbasis sollte weitere Verbesserungen erbringen. Auch ein morphologischer Ansatz bei der Erstellung des Language Model ist denkbar.

Um weitere Verbesserungen zu erreichen kann man ein *Vokaltraktnormierung* vornehmen. Mit dieser Methode wird versucht, die Auswirkungen unterschiedlich langer Vokaltrakte zu kompensieren. Dieser Unterschied besteht vor allem zwischen den Geschlechtern.

Literatur

[Brow93] W. Browne: *Serbo-Croat* in B. Comrie and G. G. Corbett (editors) The Slavonic Languages, 1993

[FGHK97] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, M. Westphal: *The Karlsruhe-Verbmobil Speech Recognition Engine* IEEE International Conference on Acoustics, Speech and Signal Processing, Munich, 1997

[LWLF97] A. Lavie, A. Waibel, L. Levin,. M. Finke, D. Gates, M. Gavaldà, T. Zeppenfeld, P. Zhan: *Janus-III: Speech-To-Speech Translation in Multiple Languages* IEEE International Conference on Acoustics, Speech and Signal Processing, Munich, 1997

[RaSc78] L. Rabiner, R. Schafer: *Digital Processing of Speech Signals*, Prentice Hall Inc., 1978

[RiSG96] K.Ries, B. Suhm, P. Geutner: *Language Modeling in* JANUS , Interner Bericht, Institut für Logik, Komplexität und Deduktionssysteme der Universität Karlsruhe, 1996

[Rogi96] I. Rogina: JANUS *3 Tutorial*, Interner Bericht, Institut für Logik, Komplexität und Deduktionssysteme der Universität Karlsruhe, 1996

[Rogi97] I. Rogina: *Paramaterraumoptimierung für Diktiersysteme mit unbeschränktem Vokabular*, Dissertation, Fakultät für Informatik der Universität Karlsruhe, 1997

[Schu95a] T. Schultz: *Identifizierung von Sprachen* Diplomarbeit, Institut für Logik, Komplexität und Deduktionssysteme der Universität Karlsruhe, 1995

[Schu95b] E. G. Schukat-Talamazzini: *Automatische Spracherkennung*, Vieweg Verlag, 1995

[ScWW97] T. Schultz, M. Westphal, A. Waibel: *The GlobalPhone Project: Multilingual LVCSR with* JANUS *-3* in : Multilingual Information Retrieval Dialogs: 2nd SQEL Workshop, Seite 20-27, Plzen, April 1997

[Stör97] Hans Joachim Störig: *Abenteuer Sprache, Ein Streifzug durch die Sprachen der Erde*, Humboldt Verlag (in Zusammenarbeit mit Langenscheidt), 1997