Karlsruhe Institute of Technology

Analyse von Methoden zum Graphemclustering bei der automatischen Spracherkennung

Bachelorarbeit
von

Christoph Wallisch

An der Fakultät für Informatik
Institut für Anthropomatik und Robotik (IAR)

Erstgutachter:	Prof. Dr. Alexander Waibel
Zweitgutachter:	Dr. Sebastian Stüker
Betreuender Mitarbeiter:	M.Sc. Markus Müller

Bearbeitungszeit: 20. Februar 2014 – 13. Mai 2014

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, den 12.05.2014

.....

(**Christoph Wallisch**)

Abstract

In dieser Arbeit werden drei verschiedene Methoden für die graphembasierte Automatische Spracherkennung entwickelt, implementiert und getestet. Dabei wird ein großer Fokus darauf gerichtet, dass die Methoden vor allem für Sprachen mit geringer Verbreitung gute Ergebnisse erzielen.

Die erste Methode ist das Merging von Graphemen. Bei dieser Methode werden Graphempaare, die oft in der Zielsprache vorkommen, zu einem neuen, kombinierten Graphem zusammengefügt. Basierend auf den neuen Graphemen wird anschließend ein neues System trainiert. Dabei werden zwei verschiedene Modifikationen der Methode behandelt. Zum Einen ein rein vorkommenbasierter Ansatz und zum Anderen ein konsonantenbasierter Ansatz, bei dem nur Konsonanten zu neuen kombinierten Graphemen zusammengefügt werden.

Die zweite Methode ist das Codebook-Clustering. Bei dieser Methode werden Codebooks, welche die Gauß-Verteilungen enthalten, von verschiedenen Distributionen, welche die Gaußschen Mischverteilungsgewichte enthalten, miteinander geclustert um die Menge an zu trainierenden Gauß-Verteilungen zu reduzieren. Anschließend wird basierend auf den geclusterten Codebooks ein neues System trainiert. Bei dieser Methode werden auch zwei Modifikationen behandelt. Zum Einen das rein distanzbasierte Codebook-Clustering, welches die Codebooks, die sich am Ähnlichsten sind, clustert und zum Anderen das vorkommenbasierte Codebook-Clustering, welches Codebooks, die wenige Trainingsdaten abbekommen, mit besser trainierten Codebooks clustert. Die Ähnlichkeit der Codebooks wird dabei mit Hilfe der Kullback-Leibler-Divergenz festgestellt.

Die dritte Methode ist das Graphem-Clustering. Bei dieser Methode werden Grapheme, die sich am Ähnlichsten sind geclustert. Um die Grapheme zu clustern müssen dabei auch die entsprechenden Distributionen und Codebooks geclustert werden. Die Ähnlichkeit von Graphemen wird dabei mit Hilfe einer modifizierten Kullback-Leibler-Divergenz festgestellt. Anschließend wird basierend auf den geclusterten Graphemen ein neues System trainiert.

Alle drei Methoden werden basierend auf den Sprachen Haitianisches Kreolisch und Zulu implementiert und getestet. Die Systeme werden dabei mit Hilfe der Word Error Rate (WER) bewertet.

Inhaltsverzeichnis

1. Einleitung

Die Rechenleistung hat in den letzten Jahren immer weiter zugenommen und es ist auch jetzt kein Ende dieser Entwicklung in Sicht. Davon haben in den letzten Jahren praktisch alle Felder der Wissenschaft profitiert und werden es wohl auch künftig weiterhin tun. Manches Problem der Wissenschaft wäre ohne die gesteigerte Rechenkapazität nur sehr viel langsamer oder sogar überhaupt nicht lösbar.

Ein Gebiet in der Wissenschaft, das sehr stark von der gesteigerten Rechenleistung profitiert, ist die Automatische Spracherkennung (ASR). Diese benötigt seit jeher große Mengen an Rechenleistung um Systeme zu erstellen, welche so fehlerresistent wie möglich sind. Heutzutage sind auch komplexe Aufgaben wie die Echtzeit Sprach-zu-Sprach Übersetzung möglich, wobei sich Sprache hierbei nicht nur auf geschriebene, sondern vor allem auch auf gesprochene Sprache bezieht. Dies ist ein Beispiel bei dem die Gebiete der Automatischen Spracherkennung und der Maschinellen Übersetzung zusammen arbeiten.

Vor allem die höhere Rechenleistung von Smartphones, Tablets und anderen mobilen Geräten eröffnet neue Anwendungsgebiete für die Automatische Spracherkennung und den darauf zum Teil aufbauenden Systemen der Maschinellen Übersetzung. Ein Traum für viele Menschen wäre es, wenn man einen Satz in ein Handy sprechen könnte und dieser in jede auf der Welt gesprochene Sprache in Echtzeit fehlerlos übersetzt werden könnte. Dies würde die Kommunikation zwischen Menschen auf eine vollkommen andere Ebene heben. Momentan wird die Echtzeit Sprach-zu-Sprach Übersetzung nur für einige wenige Sprachpaare erforscht. Um alle Sprachpaare abzudecken ist noch viel Forschung zu betreiben.

Da es sich bei modernen Spracherkennungssystemen um statistische und wahrscheinlichkeitsbasierte Systeme handelt und dadurch von vielen verschiedenen Parametern, die trainiert werden müssen, abhängig ist, spielen die Trainingsdaten eine sehr große Rolle. Nicht umsonst bringt man das berühmte Zitat von Bob Mercer

> "There is no data like more data." [Chu04]

oft mit der Automatischen Spracherkennung in Verbindung. Wenn mehr Daten zur Verfügung stehen, kann man auch genauere Aussagen über die Wahrscheinlichkeiten treffen. Bei der Wahl der Daten sollte jedoch nicht nur auf die Quantität, sondern auch auf die Qualität der Daten achten. Sprachdaten, die stark mit Hintergrundgeräuschen belastet sind, oder die mit schlechten Mikrofonen aufgenommen wurden, führen dabei zu schlechteren

Ergebnissen, wie etwas weniger Sprachdaten, die dafür eine bessere Qualität aufweisen können. Um eine genaue Aussage darüber treffen zu können, welche Sprachdaten bessere Ergebnisse erzielen, muss in jedem Einzelfall getestet werden und kann nicht basierend auf den Sprachdaten eindeutig entschieden werden.

Weiterhin muss beachtet werden, dass man nicht nur beliebige Audiodaten als Sprachdaten verwenden kann, sondern dass es für die Audiodaten auch entsprechende Transkriptionen geben muss. Der Spracherkenner muss schließlich wissen, welche Parameter er während des Trainings anpassen muss. Die Transkriptionen müssen dabei manuell von Menschen, die die entsprechende Sprache verstehen, erstellt werden. Deswegen ist das Erstellen von Transkriptionen eine sehr zeitaufwändige und damit auch kostspielige Arbeit.

Die Qualität sowie die Quantität der Sprachdaten stellen dabei vor allem in wenig verbreiteten und gesprochenen Sprachen, wie Haitianisches Kreolisch oder Zulu, eine große Herausforderung dar. Da diese Sprachen damit auch vor allem weniger wirtschaftlich interessant, und damit weniger finanziell gefördert sind als Sprachen mit einer größeren Verbreitung, werden automatisch weniger kostspielige Transkriptionen angefertigt. Aber auch die durchschnittliche Qualität der Audiodateien ist niedriger, da weniger verbreitete Sprachen oft ärmeren Regionen zugeordnet werden können und damit zumeist auch schlechtere Aufnahmetechnik für die Aufzeichnung zur Verfügung steht.
Um die Menge an benötigten Sprachdaten zu reduzieren, gibt es die Möglichkeit verschiedene Clusterverfahren einzusetzen, um die Menge der zu trainierenden Parameter zu reduzieren.

Weiterhin braucht man für die die Automatische Spracherkennung Wörterbücher für die entsprechenden Sprachen. Diese Wörterbücher enthalten dabei zum Einen eine Menge von Wörtern, die in der Sprache vorkommen und zum Anderen eine Repräsentation der Aussprache selbiger Wörter. Diese müssen von Experten der entsprechenden Sprachen, welche entsprechend in weniger verbreiteten Sprachen seltener sind, manuell angefertigt werden und stellen damit einen weiteren Kostenfaktor dar.

Eine Möglichkeit Wörterbücher für die Automatische Spracherkennung automatisch erstellen zu können, stellen dabei die graphembasierten Spracherkennungssysteme dar. Bei diesen wird keine Information über die Aussprache der Wörter benötigt, da die Repräsentation der Wörter der Aufteilung in ihre Grapheme entspricht.

Zusätzlich hat die Automatische Spracherkennung mit zusätzlichen grundsätzlichen Problemen zu kämpfen. Zum Einen machen andere akustische Signale, wie Hintergrundgeräusche, große Probleme, weil sie das akustische Signal verunreinigen. Dadurch wird es schwieriger, die entsprechenden Merkmale aus dem akustischen Signal zu extrahieren und für die Spracherkennung zu nutzen.

Weiterhin stellen auch Dialekte, beziehungsweise die unterschiedliche Aussprache verschiedener Menschen, die Spracherkennung vor weitere Probleme. Das führt dazu, dass nicht nur Spracherkenner für Deutsch erstellt werden, sondern auch für einzelne Dialekte oder sogar Spracherkennungssysteme, die auf einzelne Personen spezialisiert sind. Dabei stellt sich wieder das Problem der Quantität der Sprachdaten. Dabei muss man abschätzen, ob es genug Sprachdaten für eine einzelne Person oder einen Dialekt gibt, um ein Spracherkennungssystem zu trainieren, welches bessere Ergebnisse erzielt, als die Spracherkennungssysteme, die auf einer Sprache basieren und dadurch mehr Sprachdaten zur Verfügung haben. Auch Spracherkennungssysteme, die auf dem neuesten Stand der Technik basieren, werden von diesen Problemen noch vor große Herausforderungen gestellt.

1.1. Zielsetzung

In dieser Arbeit werden verschiedene Clustermethoden untersucht, die auf graphembasierte Spracherkenner angewendet werden können. Dabei sollen die Methoden vor allem bei Sprachen zu Verbesserungen führen, die weniger erforscht und verbreitet sind, und damit weniger Sprachdaten zur Verfügung stehen haben. Ziel ist es, Methoden zu finden, die auf möglichst viele Sprachen angewendet werden können und eine Verbesserung der Erkennungsrate (gemessen mit Hilfe der Word Error Rate (WER)) zur Folge haben.

1.2. Struktur

Die Arbeit besteht aus einem theoretischen Teil, der die Grundlagen und Konzepte der Automatischen Spracherkennung aufzeigt und einen praktischen Teil, der auf die Benutzung und den Einfluss der getesteten Methoden auf ein richtiges System eingeht.

Kapitel 2 führt in die Grundlagen der Automatischen Spracherkennung, sowie der Clusteranalyse ein. In Kapitel 3 wird die Problemstellung der Arbeit genauer beleuchtet und verschiedene Ansätze aufgezeigt. Schließlich werden in Kapitel 4 die verwendeten Algorithmen vorgestellt und erklärt.

Danach wird in Kapitel 5 die Implementierung der Spracherkennungssysteme zusammen mit den Sprachdaten vorgestellt. Die Evaluierung der Spracherkennungssysteme wird im folgenden Kapitel 6 vorgestellt und schlussendlich werden in Kapitel 7 die Ergebnisse der Arbeit zusammengefasst und Möglichkeiten für Verbesserungen und Weiterentwicklungen aufgezeigt.

1.3. Verwandte Arbeiten

Heutzutage wird in der Automatischen Spracherkennung immer mehr Wert auf die Erschließung von weniger verbreiteten Sprachen für die Spracherkennung gelegt. Dabei wurden schon verschiedene Methoden und Ansätze für die leichtere Erschließung neuer Sprachen, entwickelt, umgesetzt und getestet.

Im Folgenden werden wir auf drei verschiedene Ansätze eingehen und entsprechende Arbeiten vorstellen. Zuerst befassen wir uns dabei mit der Verwendung von Clusteralgorithmen um die Menge an Akustischen Modellen, die trainiert werden müssen, zu reduzieren. Des Weiteren gehen wir auf graphembasierte Systeme, welche den Vorteil haben, dass sie kein phonetisches Aussprachewörterbuch benötigen, ein. Schließlich schauen wir uns die Multilinguale und Crosslinguale Akustische Modellierung an, die versucht von Spracherkennungssystemen anderer Sprachen für Spracherkennungssysteme der Zielsprache zu profitieren .

Clusteralgorithmen

In der Automatischen Spracherkennung werden vor allem hierarchische Clusterverfahren verwendet. Zu den hierarchischen Clusterverfahren gehören zum einen die divisiven [RF97], als auch die agglomerativen [BIA00] Clusterverfahren. Weiterhin gibt es auch Verfahren, die beide Clusterverfahren verbinden. [RSS99]

Um Akustische Modelle sinnvoll zu clustern, benötigt man Distanzmaße um die Ähnlichkeit oder auch die Unähnlichkeit von Akustischen Modellen zueinander zu bestimmen.

Dabei ist das Finden von geeigneten Distanzmaßen für die Automatische Spracherkennung ein übliches Problem, da die Effizienz von Clusteralgorithmen sehr stark von den Distanzmaßen abhängt. [DIG12, JS06]

Graphembasierte Automatische Spracherkennung

Graphembasierte Spracherkennungssysteme nehmen in der Spracherkennung einen immer größeren Stellenwert ein, da man bei graphembasierten Spracherkennungssystemen das Aussprachewörterbuch, welches für gute Ergebnisse aufwändig von Hand erstellt werden muss, nicht benötigt.

Man kann auch die manuelle Erstellung von Aussprachewörterbüchern umgehen, indem man diese automatisch erstellen lässt. [BN08] Dabei sind natürlich dementsprechend viele Fehler im Aussprachewörterbuch zu finden. Verschiedene Arbeiten haben sich daher schon mit dem Vergleich zwischen graphembasierten Spracherkennungssystemen und phonembasierten Spracherkennungssystemen, die auf einem automatisch erstellten Aussprachewörterbuch basieren, beschäftigt. Dabei wurde zum Beispiel für die Sprache Thai festgestellt, dass das graphembasierte System leicht schlechtere Erkennungswerte aufweiset als das phonembasierte System mit einem manuell erstellten Aussprachewörterbuch, aber eine deutlich bessere Erkennungsrate als das phonembasierte System mit automatisch erstelltem Aussprachewörterbuch. [PCS06, SOS12]

Andere Arbeiten beschäftigen sich nur rein mit der Aufgabenstellung ob gewisse Sprachen für die graphembasierten Systeme geeignet sind oder nicht.
Beispiele hierfür sind das Paper zu graphembasierter Sracherkennung von Russisch [SS04], Englisch [MMDB11] oder eine ausführlichere Arbeit, die sich mit acht verschiedenen Sprachen beschäftigt.[Jan12]

In diesen Arbeiten wurde festgestellt, dass Sprachen wie Spanisch oder Türkisch, welche sehr enge Graphem-zu-Phonem Bindung haben, bei den graphembasierten Spracherkennungssystemen gleich gute, oder sogar leicht bessere Ergebnisse erzielen, als die phonembasierten. Weiterhin wurde festgestellt, dass Englisch für graphembasierte Systeme eher ungeeignet ist und deutlich schlechtere Ergebnisse erzielt, als die phonembasierten Systeme. Für die anderen Sprachen wurden leicht schlechtere Erkennungsraten erzielt.

Multilinguale und Crosslinguale Akustische Modellierung

Unter Multilingualen Spracherkennungssystemen versteht man Systeme bei denen die Akustischen Modelle eines Spracherkenners mit Trainingsdaten verschiedener Sprachen trainiert werden. Im Gegensatz dazu werden bei Crosslingualen Spracherkennungssystemen die Akustischen Modelle basierend auf Akustischen Modellen anderer Sprachen aufgebaut.

Bei der Multilingualen und Crosslingualen Akustischen Modellierung werden Spracherkennungssysteme für neue Sprachen auf Basis von vortrainierten Gaußschen Mischverteilungen erstellt und trainiert. Dabei stammen die vortrainierten Gaußschen Mischverteilungen aus anderen Sprachen, welche ähnliche oder die gleichen Phoneme vorzuweisen haben.

Eine Methode der Multilingualen und Crosslingualen Akustischen Modellierung sind Subspace Gaussian Mixture Models (SGMM). (siehe Kapitel 3.2.1) Bei konventionellen Akustischen Modellen, die auf Hidden Markov Modelle (HMM) aufbauen, werden die Emmisionswahrscheinlichkeiten jedes Zustands durch Gaußsche Mischverteilungsmodelle (GMM)

dargestellt. Bei SGMM wird auch jeder Zustand oder Unterzustand auf ein GMM abgebildet. Der Unterschied besteht darin, dass die Standardabweichungen und Gewichtungen der GMMs aus den Unterräumen, welche phonetische Informationen enthalten, die mit Hilfe von mehreren anderen Sprachen erstellt wird, abgeleitet wird. [LLR11]

Einen ähnlichen Ansatz wie die SGMM verfolgen die Subspace Mixture Modells (SMM). (siehe Kapitel 3.2.2) Der Unterschied zu den SGMM besteht darin, dass nicht auf einen Unterraum mit phonetischen Informationen sondern auf mehrere Unterräume zurückgegriffen wird. Dabei gibt es pro Quellsprache eine eigene Menge an Unterräumen, welche gewichtet in das SMM einfließen. [YMW13]

Sowohl das SGMM als auch das SMM beruhen dabei auf den phonetischen Informationen, die man über andere Sprachen besitzt und auf die neue Sprache anwendet.
Da die meisten Sprachen, welche wenig Daten zur Verfügung haben, auch einen Mangel an Aussprachewörterbüchern aufweisen können, wird auch an Multilingualen graphembasierten Spracherkennungssystemen geforscht. [RBD13]

2. Grundlagen

In diesem Kapitel werden die Grundlagen der Automatischen Spracherkennung und andere Grundlagen, die zum Verständnis der Arbeit beitragen, beschrieben.

Zuerst gibt es eine kurze Einführung in die Unterschiede zwischen Phonemen und Graphemen. Danach werden die Grundlagen der Automatischen Spracherkennung erklärt und schließlich wird noch die Bewertungsfunktion für die Evaluierung vorgestellt.

2.1. Grapheme und Phoneme

Phoneme

Ein **Phonem** ist die kleinste bedeutungsunterscheidende **akustische** Einheit der Sprache. [pho] Das heißt die gesprochene Sprache ist aus einer endlichen Anzahl von Phonemen zusammen gesetzt, aus der man Wörter oder Sätze bilden kann. In der Spracherkennung werden Phoneme in Phonemklassen eingeteilt. Zum Einen in die zwei großen Klassen Konsonanten und Vokale, sowie in die Untergruppen Nasale, Laterale, ... Dies kann man sich zum Beispiel beim Clustern von Phonemen zu Nutze machen.

Grapheme

Ein **Graphem** ist dagegen die kleinste bedeutungsunterscheidende **graphische** Einheit der Sprache. [gra] Grapheme entsprechen deswegen größtenteils den Buchstaben. Besondere sprachenspezifische Buchstabenkombinationen, wie das CH in der deutschen Sprache, werden aber auch manchmal als eigenständiges Graphem gezählt.

In der Spracherkennung unterscheidet man Systeme in phonembasierte und graphembasierte Systeme. Erstere sind hierbei die Norm. Phoneme sind für die Spracherkennung sehr gut geeignet, da sie dem entsprechen, was erkannt werden soll: Den gesprochenen Wortsequenzen die aus Phonemen zusammen gesetzt sind. Graphembasierte Spracherkenner haben dagegen einen anderen großen Vorteil. Aussprachewörterbücher für phonembasierte Systeme müssen aufwändig von Hand erstellt werden. Bei graphembasierten Systemen ist dies nicht der Fall. Dies kann vor allem ein großer Vorteil für weniger verbreitete Sprachen sein, für die noch keine großen Aussprachewörterbücher existieren.

2.2. Automatische Spracherkennung (ASR)

Die **Automatische Spracherkennung (ASR)** befasst sich mit der automatischen Übersetzung von gesprochener Sprache in eine textuelle Repräsentation derselben. Dabei gibt es bei der Automatischen Spracherkennung viele Schwierigkeiten: Rauschen, Hintergrundgeräusche und unterschiedliche Sprecher, um nur einige zu nennen.

Ein typisches Spracherkennungssystem besteht aus den Komponenten, die in Abbildung 2.1 gezeigt werden. Dabei wird der Vorgang der Spracherkennung in die Teile Vorverarbeitung und Dekodierung unterteilt.

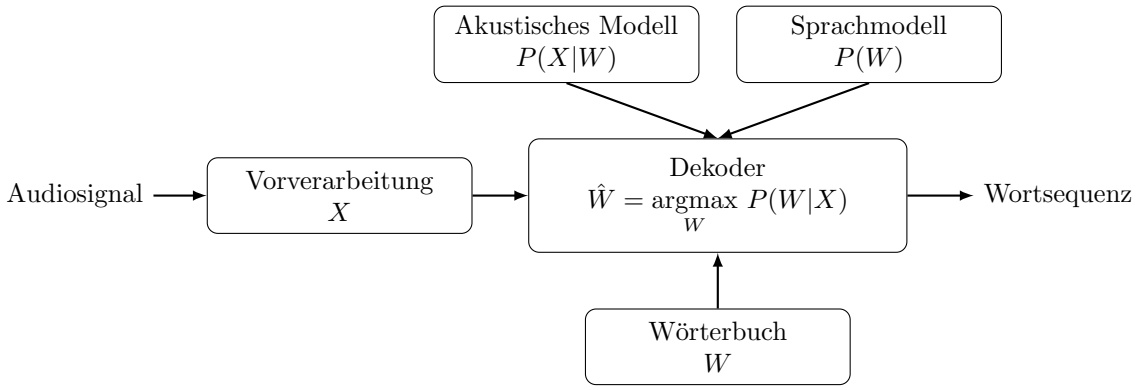


Abbildung 2.1.: Ablaufdiagramm eines typischen Spracherkennungssystems

Durch die Benutzung eines Mikrophons wird eine menschliche Äußerung aufgenommen und man erhält ein akustisches analoges Signal. Dieses analoge Signal wird abgetastet um eine digitale Repräsentation des Signals zu bekommen. In der Vorverarbeitung wird die digitale Repräsentation ausgewertet und eine Sequenz von Merkmalsvektoren berechnet, die die wichtigsten Eigenschaften des Signals für die Erkennung der menschliche Sprache in sich tragen. Dies geschieht indem das digitale Signal gefenstert, gefiltert und anschließend in den Frequenzbereich transformiert wird. Das Ziel dabei ist es, das Eingangssignal auf die wichtigsten Merkmale des Signals zu reduzieren. Dadurch wird jedes Phonem durch eine kurze Abfolge von Merkmalsvektoren dargestellt.

Durch die Vorverarbeitung haben wir nun eine eine Sequenz X von Merkmalsvektoren bekommen, die die Äußerung repräsentieren. Der Dekoder findet, mit Hilfe des Wahrscheinlichkeitsmaßes $P(W|X)$, die wahrscheinlichste Wortsequenz $\hat{W}$, welche man als Hypothese bezeichnet. Durch Benutzung der Bayesregel kann dies in die **Fundamentalformel der Spracherkennung** zerlegt werden: [ST95, p. 165]

$$P(W|X) = \frac{P(X|W) \cdot P(W)}{P(X)} \tag{2.1}$$

Dabei entspricht $P(X|W)$ der Wahrscheinlichkeit, dass eine Sequenz X von Merkmalsvektoren beobachtet wird, unter der Voraussetzung einer Sequenz W von Wörtern. Dies nennt man das **Akustische Modell**. $P(W)$ ist die a-priori-Wahrscheinlichkeit, unabhängig von anderen Wahrscheinlichkeiten, W zu beobachten, und wird bezeichnet als **Sprachmodell**. $P(X)$ ist die a-priori-Wahrscheinlichkeit die Sequenz X von Merkmalsvektoren zu beobachten.

Der Dekoder versucht nun

$$\begin{aligned}\hat{W} &= \underset{W}{\operatorname{argmax}}\ P(W|X) \\ &= \underset{W}{\operatorname{argmax}}\ \frac{P(X|W)\cdot P(W)}{P(X)} \\ &= \underset{W}{\operatorname{argmax}}\ P(X|W)\cdot P(W)\end{aligned} \tag{2.2}$$

zu bestimmen. [ST95, p. 165]

Dafür durchsucht der Dekoder Phonem und Wortsequenzen. Diese werden durch ein **Wörterbuch**, welches eine Menge von Wörtern mit der entsprechenden Aussprache enthält, eingegrenzt. In modernen Systemen werden die drei Komponenten Akustisches Modell, Sprachmodell und Wörterbuch parallel genutzt.
Die vorliegende Arbeit beschäftigt sich mit Clusterverfahren, welche auf das Sprachmodell keinen Einfluss haben. Deswegen werden im Folgenden das Wörterbuch und das Akustische Modell genauer beschrieben.

2.2.1. Wörterbuch

Das Wörterbuch ist die Komponente die das Akustische Modell und das Sprachmodell verbindet, indem es die **orthographische** Darstellung mit der **phonetischen** Darstellung verbindet.

Das Akustische Modell arbeitet mit Sequenzen von Phonemen. Um diese in Wörter verwandeln zu können, werden im Wörterbuch beide Formen gespeichert. Dies führt dazu, dass Wörter, die nicht im Wörterbuch vorkommen, nicht vom Akustischen Modell erkannt werden können. Dies führt zu einer Begrenzung des Suchraums, aber kann auch dazu führen, dass Wörter, die **out of vocabulary** (OOV) sind, nicht erkannt werden.

`Ani`	`{{A WB} N {I WB}}`
`Chino`	`{{C WB} H I N {O WB}}`
`Jabulani`	`{{J WB} A B U L A N {I WB}}`
`Nxamalala`	`{{N WB} X A M A L A L {A WB}}`
`izosuke`	`{{I WB} Z O S U K {E WB}}`
`mpama`	`{{M WB} P A M {A WB}}`

Tabelle 2.1.: Auszug aus einem Wörterbuch der Sprache Zulu

In Tabelle 2.1 sieht man ein Beispiel für ein Wörterbuch der Sprache Zulu. Dabei stehen die eingeklammerten Grapheme mit dem Kennzeichen `WB` für Wortanfang und -ende.

2.2.2. Akustisches Modell

Die Aufgabe des Akustischen Modells ist es für eine Sequenz von Phonemen die Wahrscheinlichkeit anzugeben, dass sie den Merkmalsvektoren des Eingangssignals entsprechen. Dies wird durch unterschiedliche Sprecher, Hintergrundgeräusche und andere Schwierigkeiten noch mehr erschwert.

In den meisten modernen Spracherkennern werden für das Akustische Modell Hidden Markov Models (siehe Kapitel 2.2.2.1) eingesetzt. Dabei setzt man für die Emissionen in modernen Systemen meistens Gaußsche Mischverteilungen (siehe Kapitel 2.2.2.2) ein. Weiterhin sind die Grundlagen der Clusteranalyse (siehe Kapitel 2.2.2.3) wichtig für die vorliegende Arbeit, weswegen darauf kurz eingegangen wird.

2.2.2.1. Hidden Markov Model (HMM)

Ein **Hidden Markov Model** (siehe Abbildung 2.2) ist ein 5-Tupel $\lambda = (S, \pi, A, B, V)$ mit der Zustandsmenge S, Initialwahrscheinlichkeiten π, die für jeden Zustand angeben, wie wahrscheinlich es ist, dass dieser Zustand der Anfangszustand ist, der Zustandsübergangsmatrix A, den Emmisionswahrscheinlichkeiten B, und dem Vokabular V, aus dem die Beobachtungen stammen. [Rab89]

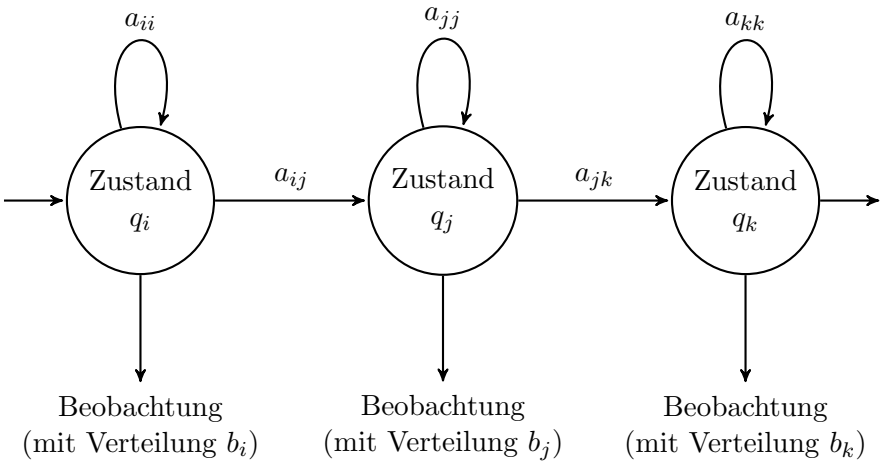


Abbildung 2.2.: Aufbau eines Hidden Markov Models

Der Begriff "hidden" bedeutet bei HMM, dass es unbekannt ist, welche Zustände durchlaufen werden. Man kennt nur die Beobachtungen und kann dann mit Hilfe der Übergangs- und Emmissionswahrscheinlichkeiten die wahrscheinlichste durchlaufene Zustandsfolge berechnen. Dabei basiert das Modell auf der 1. Markov-Eigenschaft: [Rab89]

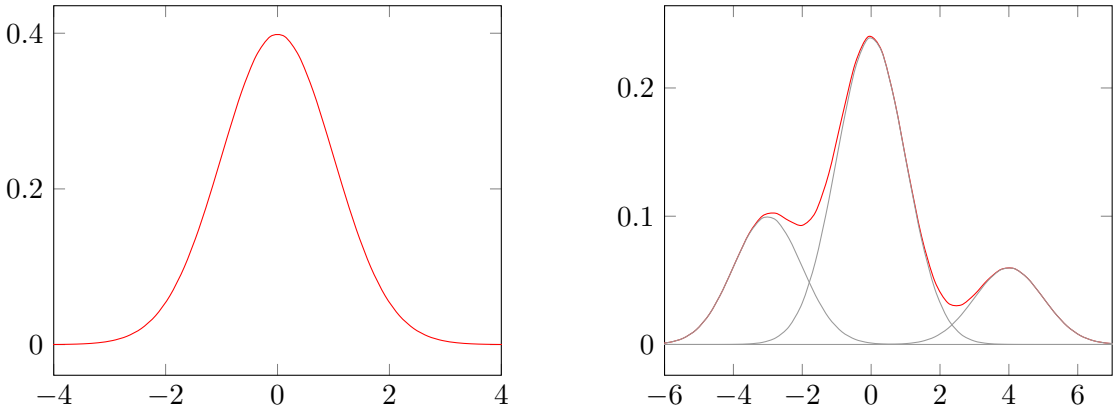
$$\begin{aligned} &P(q_t|q_1, ..., q_{t-1}) = P(q_t|q_{t-1}) \\ &q_t : \text{Zustand in dem sich das System in Zeitpunkt t befindet} \end{aligned} \tag{2.3}$$

Die 1. Markov-Eigenschaft bedeutet, dass für die Wahrscheinlichkeit des aktuellen Zustands nur der direkt davor liegende Zustand betrachtet werden muss und nicht alle anderen Zustände.
In der Spracherkennung entsprechen die Zustände des Hidden Markov Models einem Laut und die Emissionen in den hier verwendeten Spracherkennungssystemen Gaußschen Mischverteilungen.

2.2.2.2. Gaußsches Mischverteilungsmodell (GMM)

Ein **Gaußsches Mischverteilungsmodell** ist eine gewichtete Summe von **Gauß-Verteilungen**(auch Normalverteilung). Gauß-Verteilungen $\mathcal{N}$ (siehe Abbildung 2.3(a)) sind wie folgt definiert: [Rey08]

$$\begin{aligned} &\mathcal{N}(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \cdot exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \\ &\mu : Standardabweichung \\ &d : Dimensionen \\ &\Sigma : Kovarianzmatrix \end{aligned} \tag{2.4}$$



(a) GV mit $\mu = 0$ und $\Sigma = 1$ (b) GVM mit 3 Gauß-Verteilungen

Abbildung 2.3.: Gauß-Verteilung (GV) und Gaußsche Mischverteilung (GMM)

Eine Gaußsche Mischverteilung Γ (siehe Abbildung 2.3(b)) ist eine gewichtete Summe von Gauß-Verteilungen und ist wie folgt definiert: [Rey08]

$$\Gamma(x) = \sum_{m=1}^{M} \omega_m \cdot \mathcal{N}(x|\mu_m, \Sigma_m)$$
$$\omega_m : \text{Verteilungsgewichte mit } \sum_{m=1}^{M} \omega_m = 1 \text{ und } 0 \leq \omega_m \leq 1 \ \forall m \in \{1 \ ... \ M\} \tag{2.5}$$

In der Spracherkennung werden Gaußsche Mischverteilungen eingesetzt um die Wahrscheinlichkeit zu errechnen, ob ein Merkmalsvektor einem Phonem oder einem Teil eines Phonems entspricht. Abbildung 2.4 zeigt, welchen Einfluss die Gewichtungen auf die Gaußschen Mischverteilungen haben. Die Gewichtungen wurden dabei auf die Standardnormalverteilung angewandt. Diese hat die Besonderheit, dass sie für die Standardabweichung $\mu = 0$ und für die Kovarianzmatrix $\Sigma = 1$ annimmt. (siehe Abbildung 2.3(a))

In der Spracherkennung hat man aber häufig das Problem, dass für manche GMM nur wenige Trainingsdaten zur Verfügung stehen. Eine Lösung für dieses Problem ist es, Gauß-Verteilung mehrfach zu benutzen, um die Menge an Trainingsdaten pro Gauß-Verteilung zu erhöhen. Dies führt dazu, dass die Gauß-Verteilung nicht mehr auf genau ein Phonem oder Teil eines Phonems spezialisiert sind. Man unterscheidet hier zwischen: [ST95, p. 144]

- **Vollkontinuierliches Markovmodell:** Eine Normalverteilung wird mit einer Gewichtung verknüpft. Kann die besten Ergebnisse erzielen, wenn genug Trainingsdaten zur Verfügung stehen.
- **Semikontinuierliches Markovmodell:** Eine Normalverteilung wird mit mehreren Gewichtungen verknüpft. Dadurch müssen weniger Parameter trainiert werden und es werden weniger Trainingsdaten benötigt. Dieser Ansatz wird in Kapitel 3.3.3 aufgegriffen.

Eine Menge von Normalverteilungen wird im Folgenden Codebook genannt und die dazugehörigen Gewichtungen Distributionen. Zusammen ergeben Codebook und Distribution eine Menge von Gaußschen Mischverteilungen.

2.2.2.3. Clusteranalyse

In diesem Kapitel gibt es eine kurze Einführung in die Clusteranalyse. Die genauen Algorithmen die verwendet wurden werden in Kapitel 4 genauer vorgestellt.

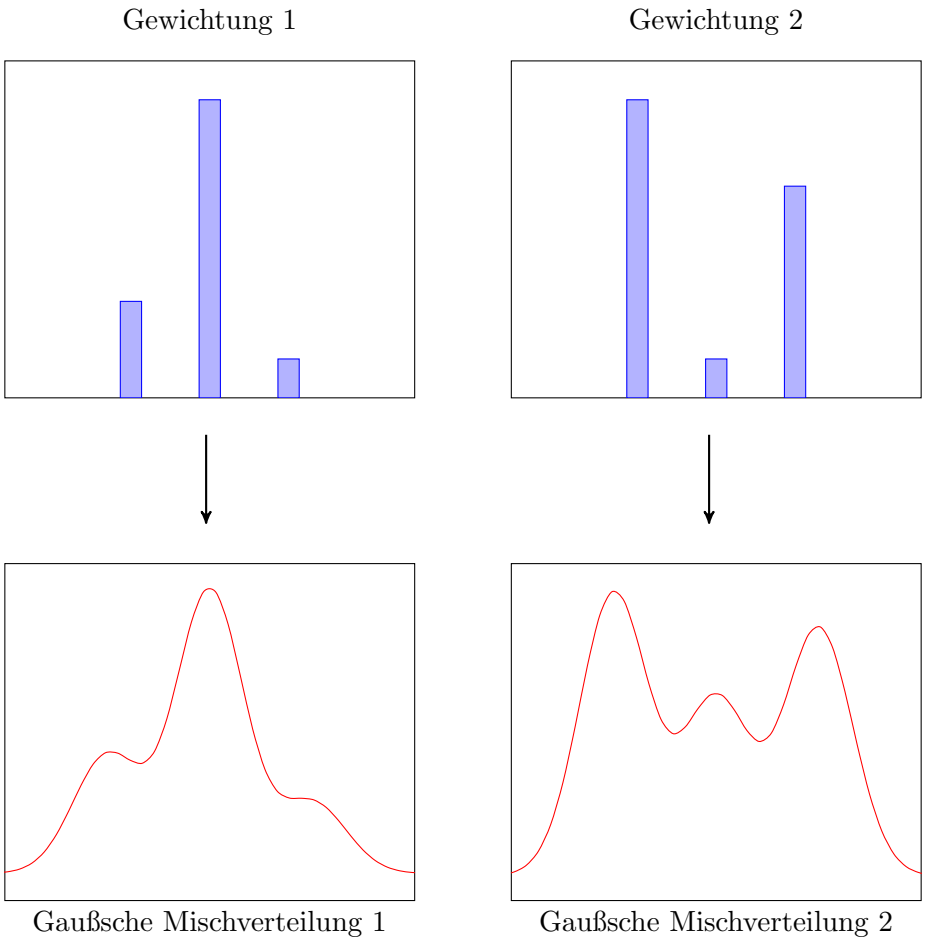
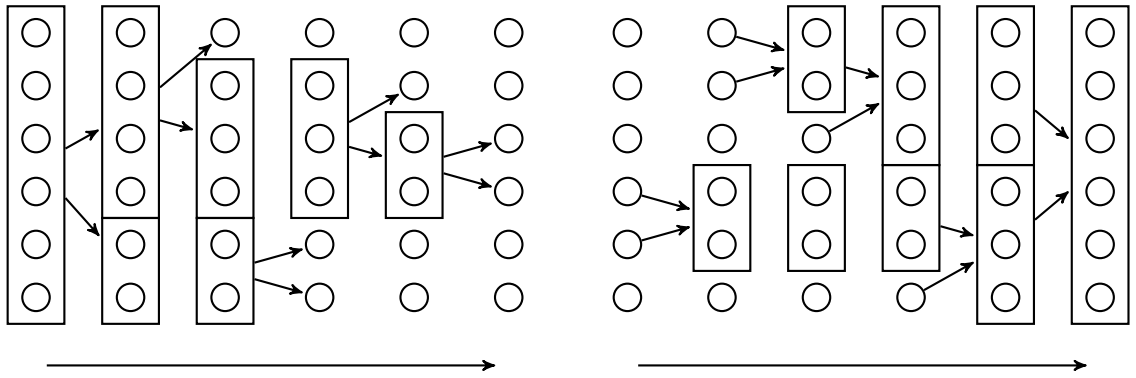


Abbildung 2.4.: Gewichtungen und dazugehörige Gaußsche Mischverteilungen

In der Clusteranalyse unterteilt man Objekte in eine bestimmte Menge von Klassen, welche auch Cluster genannt werden. Dabei sollten die Objekte in der selben Klasse möglichst ähnlich zueinander sein und Objekte in unterschiedlichen Klassen sollten relativ unterschiedlich sein. Um die Ähnlichkeit von Objekten zueinander abschätzen zu können, werden verschiedene Metriken verwendet. Eine Metrik die in dieser Arbeit verwendet wird ist die **Kullback-Leibler-Divergenz**. (siehe Kapitel 2.2.2.3)

In der Clusteranalyse kann man zwischen sehr vielen verschiedenen Clusterverfahren unterscheiden. In der vorliegenden Arbeit haben wir uns auf die **Hierarchische** Clusteranalyse beschränkt. Die Hierarchische Clusteranalyse kann man in zwei Grundlegende Verfahren unterteilen:

- **Divisives Clusterverfahren:** Auch Top-Down-Verfahren genannt. Alle Objekte befinden sich am Anfang in einer großen Klasse und werden danach in mehrere Klassen unterteilt, bis man die richtige Menge an Clustern hat. (siehe Abbildung 2.5(a)) [RF97]
- **Agglomeratives Clusterverfahren:** Auch Bottom-up-Verfahren genannt. Jedes Objekt befindet sich am Anfang in einem eigenen Cluster. Danach werden so lange die Cluster, die sich am ähnlichsten sind, in ein neues Cluster zusammen gefasst, bis man die richtige Menge an Clustern hat. Dieses Verfahren wurde in dieser Arbeit angewendet. (siehe Abbildung 2.5(b)) [BIA00]



(a) Divisives Clusterverfahren (b) Agglomeratives Clusterverfahren

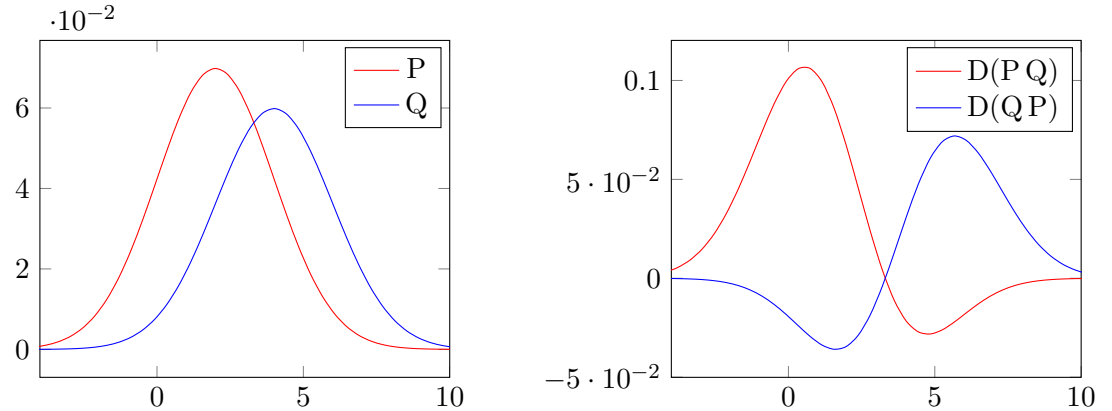
Abbildung 2.5.: Visualisierung der Clusterverfahren

Kullback-Leibler-Divergenz

Die **Kullback-Leibler-Divergenz** kann als Unähnlichkeitsmaß zwischen zwei Wahrscheinlichkeitsfunktionen gesehen werden und ist wie folgt definiert: [KL51]

$$d_{KL}(P,Q) = \int_{x \in X} P(x) \cdot \log \frac{P(x)}{Q(x)} \tag{2.6}$$

Abbildung 2.6(a)) zeigt zwei Gauß-Verteilungen auf die die Kullback-Leibler-Divergenz angewendet werden soll und Abbildung 2.6(b) zeigt die resultierenden Kullback-Leibler-Divergenzen. Wie man hier leicht beobachten kann, ist die Kullback-Leibler-Divergenz nicht symmetrisch und befolgt damit nicht die Dreieckungsungleich. Damit kann die Kullback-Leibler-Divergenz nicht als Distanzmaß eingesetzt werden.



(a) Zwei normale Verteilungen P und Q (b) KL-Divergenz zwischen P und Q

Abbildung 2.6.: Veranschaulichung der Kullback-Leibler-Divergenz

Um die Kullback-Leibler-Divergenz dennoch als Distanzmaß einsetzten zu können, kann man die Kullback-Leibler-Divergenz symmetrisch machen indem man sie zwischen P und Q mit der zwischen Q und P aufaddiert.

$$d_{KL-sym}(P,Q) = d_{KL}(P,Q) + d_{KL}(Q,P) \tag{2.7}$$

Wenn man nun die Kullback-Leibler-Divergenz für Gaußsche Mischverteilungen mit diagonalen Kovarianzmatrizen einsetzt, bekommt man folgende Formel:

$$d_{KL-sym}(\Gamma_1, \Gamma_2) = \frac{1}{2} \sum_{i=1}^{d} \frac{\Sigma_1}{\Sigma_2} + \frac{\Sigma_2}{\Sigma_1} + \left(\frac{1}{\Sigma_1} + \frac{1}{\Sigma_2} \right) (\mu_{1,i} - \mu_{2,i})^2 \tag{2.8}$$

Diese Form der Kullback-Leibler-Divergenz wurde in dieser Arbeit eingesetzt.

2.3. Evaluierung von Spracherkennungssystemen

Die Qualität von Spracherkennungssystemen wird praktisch immer an ihrer Genauigkeit gemessen.
Dabei schaut man sich für eine gegebene Äußerung die erwünschte Wortsequenz an, welche man Referenz nennt. Diese ist meistens eine manuelle Niederschrift der Wortsequenz, die in der Äußerung vorkommt. Dadurch bietet die Referenz eine Möglichkeit die Genauigkeit eines Spracherkenners zu bestimmen, indem man die Referenz mit der Hypothese, also der Wortsequenz, die vom Spracherkenner ausgegeben wird, vergleicht.

Für die Quantifizierung der Genauigkeit wurde in dieser Arbeit die **Word Error Rate (WER)** verwendet, welche eine Modifikation der **Levenshtein Distanz** [Lev66] darstellt.

$$\begin{aligned} \text{WER} &= \frac{s + d + i}{n} \\ s &: \text{Anzahl an Ersetzungen (Substitutions)} \\ d &: \text{Anzahl an Löschungen (Deletions)} \\ i &: \text{Anzahl an Einfügungen (Insertions)} \\ n &: \text{Anzahl der Wörter in der Referenz} \end{aligned} \tag{2.9}$$

Um die Word Error Rate zu berechnen, wird die minimale Anzahl an Ersetzungen, Einfügungen und Löschungen gesucht, die benötigt wird um von der Hypothese zur Referenz zu gelangen.

Wie schon erwähnt, ist die Word Error Rate eine Modifikation der Levenshtein Distanz. Diese berechnet auf Basis zweier Zeichenketten (der Hypothese h und der Referenz r) eine Distanzmatrix D, die wie folgt definiert ist:

$$\begin{aligned} &D_{0,0} = 0 \\ &D_{i,0} = i, 1 \leq i \leq |h| \\ &D_{0,j} = j, 1 \leq j \leq |r| \\ &D_{i,j} = \min \begin{cases} D_{i-1,j-1} & +0 \text{ falls } h_i = r_i \\ D_{i-1,j-1} & +1 \text{ (Ersetzung)} \\ D_{i,j-1} & +1 \text{ (Einfügung)} \\ D_{i-1,j} & +1 \text{ (Löschung)} \end{cases}, 1 \leq i \leq |h|, 1 \leq i \leq |r| \end{aligned} \tag{2.10}$$

Wenn die Distanzmatrix D berechnet ist, bekommt man die Levenshtein Distanz, indem man den Wert ausliest der sich in Zelle $D_{|h|,|r|}$ befindet. Dieser stellt die minimale Anzahl an Veränderungen dar, die nötig sind, um von der Hypothese zur Referenz zu kommen. Wenn man die resultierende Distanz durch die Anzahl an Zeichen in der Referenz teilt, erhält man die **Character Error Rate (CER)**

Um die Veränderungen nachvollziehen zu können, gibt es die Möglichkeit in der Distanzmatrix ein Backtrace durchzuführen. Dabei geht man ausgehend von der Zelle $D_{|h|,|r|}$ durch die Matrix immer zu den anliegenden Minima, bis man bei der Zelle $D_{0,0}$ angekommen ist.

Abbildung 2.7 zeigt ein Beispiel für eine Levenshtein Distanzmatrix, bei der die Levenshtein Distanz 4 beträgt. Die entsprechende CER würde $\frac{4}{11} = 37\%$ betragen.

	ϵ	L	E	V	E	N	S	H	T	E	I	N
ϵ	0	1	2	3	4	5	6	7	8	9	10	11
M	1	1	2	3	4	5	6	7	8	9	10	11
E	2	2	1	2	3	4	5	6	7	8	9	10
I	3	3	2	2	3	4	5	6	7	8	8	9
L	4	3	3	3	3	4	5	6	7	8	9	9
E	5	4	3	4	3	4	5	6	7	7	8	9
N	6	5	4	4	4	3	4	5	6	7	8	8
S	7	6	5	5	5	4	3	4	5	6	7	8
T	8	7	6	6	6	5	4	4	4	5	6	7
E	9	8	7	7	6	6	5	5	5	4	5	6
I	10	9	8	8	7	7	6	6	6	5	4	5
N	11	10	9	9	8	7	7	7	7	6	5	4

Abbildung 2.7.: Beispiel für eine Levenshtein Distanzmatrix mit den Beispielwörtern Levenshtein und Meilenstein

Wenn man die Levenshtein Distanz modifiziert, sodass nicht mehr einzelne Zeichen sondern ganze Wörter in den einzelnen Schritten verglichen werden und die resultierende Distanz durch die Anzahl der Wörter in der Referenz teilt, bekommt man die Word Error Rate.

3. Analyse

In diesem Kapitel wird die Problemstellung der Arbeit genauer erläutert und danach werden existierende Ansätze und Ansätze dieser Arbeit, zur Lösung der Probleme vorgestellt.

3.1. Problemstellung

Lange Zeit wurde in der Automatischen Spracherkennung hauptsächlich an den Weltsprachen (siehe Abbildung 3.1) geforscht. An der Erforschung von Spracherkennungssystemen für genannte Weltsprachen, hat vor allem die Wirtschaft und auch die Politik ein großes Interesse. Weiterhin sind für diese Sprachen die meisten Daten vorhanden. Die ausgeprägte Forschung für diese Sprachen hat dazu geführt, dass man in der Qualität der Spracherkennung in diesen Sprachen schon weit fortgeschritten ist.

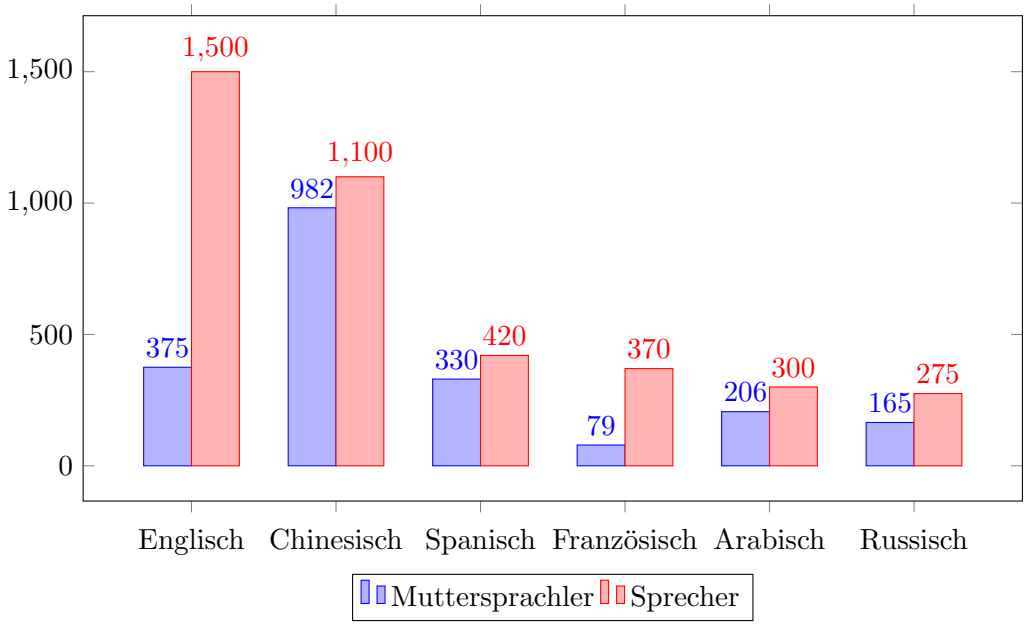


Abbildung 3.1.: Die Weltsprachen mit der Anzahl der Sprecher (in Millionen) [ws1, ws2]

Heutzutage wird weiterhin an den Weltsprachen geforscht, aber die Spracherkennung hat es sich auch zur Aufgabe gemacht die Spracherkennung für weniger verbreitete Sprachen voranzutreiben.

Da die Anzahl der Sprachen, die man erforschen will, immer weiter zunimmt, wird nach Methoden gesucht, die das Erstellen von Spracherkennungssystemen für neue Sprachen erleichtert, beziehungsweise nach Methoden, die universell auf verschiedene Sprachen angewandt werden können, um die Erkennungsrate zu verbessern.

Bei wenig verbreiteten Sprachen tauchen dabei noch mehrere Probleme auf. Zum Einen stehen für besagte Sprachen deutlich weniger Trainingsdaten zur Verfügung, und zum Anderen ist ein Mangel an geeigneten Aussprachewörterbüchern zu beobachten. Der Mangel an geeigneten Aussprachewörterbüchern lässt sich dabei auf die Tatsache zurückführen, dass gute Aussprachewörterbücher manuell von Sprachexperten, welche für weniger verbreitete Sprachen dementsprechend seltener sind, erstellt werden müssen.

3.2. Existierende Ansätze

Für die Lösung der dargestellten Probleme gibt es verschiedene Ansätze. Eine der Möglichkeiten die Menge an benötigten Trainingsdaten für Spracherkennungssysteme zu reduzieren, ist die Multilinguale und Crosslinguale Akustische Modellierung.
Unter Multilingualen Spracherkennungssystemen versteht man Systeme bei denen die Akustischen Modelle eines Spracherkenners mit Trainingsdaten verschiedener Sprachen trainiert werden. Im Gegensatz dazu werden bei Crosslingualen Spracherkennungssystemen die Akustischen Modelle basierend auf Akustischen Modellen anderer Sprachen aufgebaut.
Im Folgenden wird auf zwei Ansätze eingegangen, die sich mit Multilingualen und Crosslingualen Spracherkennungssystemen beschäftigen.

3.2.1. Subspace Gaussian Mixture Models (SGMM)

Subspace Gaussian Mixture Models (SGMM) sind ähnlich wie konventionelle Gaußsche Mischverteilungsmodelle (GMM) aufgebaut. Beide nutzen dabei Gaußsche Mischverteilungsmodelle für die Emissionswahrscheinlichkeiten der Hidden Markov Models. Der Unterschied besteht darin, dass die Standardabweichungen und Gewichtungen aus Unterräumen, die aus phonetischen Informationen bestehen, abgeleitet werden. Dabei wird jedem Unterraum eine Gauß-Verteilung zugeordnet.
Weiterhin werden die Kovarianzmatrizen zwischen allen Zuständen des HMM geteilt.

Die grundsätzliche Idee von Subspace Gaussian Mixture Models wird in Abbildung 3.2 veranschaulicht. Die Parameter der Unterräume werden in der Quellsprache abgeschätzt. Aus den Parametern der Unterräume der Quellsprachen werden die Parameter der Unterräume der Zielsprache abgeleitet. Sollte das SGMM ohne Daten von anderen Systemen trainiert werden, spricht man von einem SGMM. Sollte das SGMM aber auf die Parameter von anderen Systemen aufbauen spricht man von einem Crosslingualen SGMM.
Beim Trainieren des Crosslingualen SGMM wird auch der Unterraum der Quellsprachen modifiziert, da die neue Sprache nun auch Teil des Crosslingualen SGMM ist.

Formal sind Subspace Gaussian Mixture Models wie folgt definiert: [DPA11]

$$p(x|j,s) = \sum_{m=1}^{M_j} c_{jm} \sum_{i_1}^{I} w_{jmi} \mathcal{N}(x; \mu_{jmi}^{(s)}, \Sigma_i) \tag{3.1}$$

Dabei entsprechen j dem Zustandsindex des HMM, m dem Index eines Unterzustands, I ist die Anzahl an Gauß-Verteilungen, Σ_i entspricht der i-ten Kovarianzmatrix und c_{jm} den

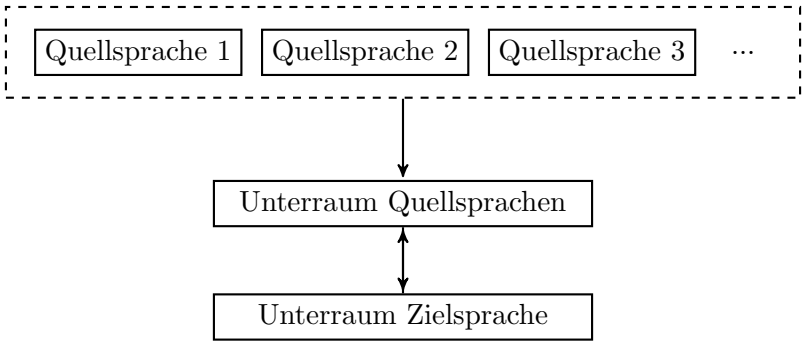


Abbildung 3.2.: Aufbau eines Crosslingualen SGMM

Gewichtungen der Unterzustände.

Die Standardabweichungen μ_{jmi} und Gewichtungen w_{jmi} sind folgendermaßen definiert:

$$\begin{aligned} \mu_{jmi}^{(s)} &= \mathrm{M}_i \mathrm{v}_{jm} \\ w_{jmi} &= \frac{\exp\ \mathrm{w}_i^T \mathrm{v}_{jm}}{\sum_{i_1'}^{I} \exp\ \mathrm{w}_{i'}^T \mathrm{v}_{jm}} \end{aligned} \tag{3.2}$$

Dabei entsprechen M_i den Unterräumen, v_{jm} den dazugehörenden Merkmalsvektoren und w_i^T den prognostizierten Gewichtungen.

Die zu trainierenden Parameter von Subspace Gaussian Mixture Models kann man in 2 Gruppen aufteilen. Zum Einen die Parameter, die mit anderen Sprachen geteilt werden und damit auch mit "out-of-language" Daten, also Daten anderer Sprachen, trainiert werden können, und zum Anderen die sprachspezifischen Parameter, welche mit Daten der Zielsprache trainiert werden müssen, um den Spracherkenner für eine spezifische Sprache zu spezialisieren.

Dabei liegt die Anzahl an Parametern um ungefähr 25% niedriger, als bei vergleichbaren HMM-GMM Systemen. Ein großer Teil (˜80%) der Parameter des SGMM sind Parameter, die auf anderen Sprachen aufbauen. Die Anzahl der sprachspezifischen Parameter, die trainiert werden müssen, liegt damit im Vergleich zu einem herkömmlichen HMM-GMM System bei ungefähr 15%. [DPA11, pp. 408,409]

3.2.2. Subspace Mixture Models (SMM)

Subspace Mixture Models (SMM) haben einen ähnlichen Ansatz wie Subspace Gaussian Mixture Models. Der grundsätzliche Unterschied zwischen Subspace Mixture Models und Subsbace Gaussian Mixture Models besteht darin, dass der Quellunterraum pro Gauß-Verteilung des SGMM durch mehrere gewichtete Unterräume pro Gauß-Verteilung im SMM ersetzt wird. Dabei basiert jeder Quell-Unterraum auf einer Sprache und nicht wie bei einem Crosslingualen SGMM auf verschiedenen Sprachen. (siehe Abbildung 3.3)

Um die Unterräume M_i der Zielsprache zu berechnen, werden die Unterräume der Quellsprachen gewichtet aufaddiert:

$$\mathrm{M}_i = \sum_{l=1}^{L} a_i^{(g)} \mathrm{M}_i^{(g)} \tag{3.3}$$

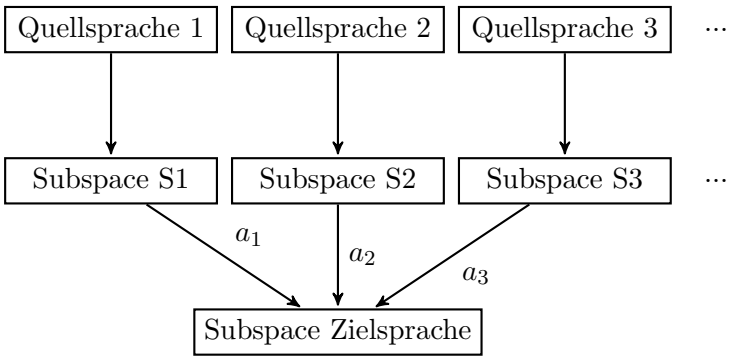


Abbildung 3.3.: Aufbau der Subspaces. a_1, a_2 und a_3 stellen die Gewichtungen der Sprachen dar.

Dabei entspricht $a_i^{(g)}$ der Gewichtung für den Unterraum $M_i^{(g)}$ und L entspricht der Anzahl der Quellsprachen.

Ein großer Vorteil gegenüber Spracherkennungssystemen, die auf SGMM aufgebaut sind, ist die größere Variabilität. Bei Spracherkennungssystemen, die auf Subspace Mixture Models aufgebaut sind, können zum Beispiel auch Quellsprachen relativ einfach für ein neues System ausgeschlossen werden, indem die entsprechenden Gewichtungen auf 0 gesetzt werden. Bei SGMM ist das nicht so einfach möglich, da die Unterräume nicht auf einzelne Sprachen spezialisiert sind, sondern alle Sprachen Einfluss auf die Unterräume nehmen. Welchen Einfluss jede Sprache auf die Parameter genommen hat, ist im Nachhinein nicht nachvollziehbar.

Bei Experimenten mit der Zielsprache Deutsch und den Quellsprachen Spanisch, Portugiesisch und Schwedisch wurden dabei mit dem SMM-basierten Spracherkennungssystem bessere Ergebnisse erzielt, als mit dem SGMM-basierten Spracherkennungssystem. [LLR11]

3.3. Ansatz dieser Arbeit

Die Ansätze, die in dieser Arbeit umgesetzt wurden, werden im Folgenden beschrieben. Zuerst wird die Idee von graphembasierten Spracherkennungssystemen vorgestellt. Anschließend wird das Graphemmerging vorgestellt und abschließend werden noch zwei Clusterverfahren vorgestellt.

3.3.1. Graphembasierte Systeme

Eine Möglichkeit um die Probleme von fehlenden beziehungsweise fehlerbehafteten Aussprachewörterbüchern zu umgehen, ist es, graphembasierte Systeme für die Automatische Spracherkennung einzusetzen. (Siehe Kaptiel 2.1 und Kapitel 2.2.1)

Dabei kann man auf die Sprachspezialisten für die entsprechenden Aussprachewörterbücher verzichten, weil die Phonemsequenz von Wörtern bei graphembasierten Spracherkennern nicht beachtet wird. In dieser Arbeit sind alle Spracherkennungssysteme graphembasiert aufgebaut.

3.3.2. Graphemmerging

Eine Methode um die Ergebnisse des Spracherkenners zu verbessern besteht darin, Graphempaare, die sehr häufig vorkommen, zu neuen Graphemen zusammenzufügen. Dies hat verschiedene Vorteile:

1. Grapheme, welche häufig in Kombination vorkommen und zusammen anders ausgesprochen werden als die einzelnen Grapheme, können besser modelliert werden. Nehmen wir zum Beispiel das Deutsche Wort Junge: Wenn wir das Wort in seine Grapheme aufteilen, bekommen wir die Grapheme J, U, N, G und E, oder wir könnten das sehr häufig vorkommende NG als neues Grapheme einführen, womit der Junge nun durch die Grapheme J, U, NG und E abgebildet wird.
2. Die Anzahl der Merkmalsvektoren pro Graphem erhöht sich. Durch die höhere Anzahl an Merkmalsvektoren pro Graphem, können die Parameter der Modelle robuster geschätzt werden. Dadurch steigt die Erkennungsleistung. Dieser Sachverhalt wird in Kapitel 4.1 genauer erläutert.

Wenn man sich nun die Vorteile anschaut, würde es normalerweise Sinn machen, so viele Grapheme wie möglich zusammenzufügen. Dabei entsteht aber das Problem, dass auch für jedes dieser Graphempaare genug Trainingsdaten vorhanden sein müssen. Das heißt, es ist immer eine Abwägungssache, wie viele Grapheme zusammenfügt werden. Deswegen werden hier nur die Graphempaare zusammengefügt, die in den Trainingsdaten häufig vorkommen

Dieser Ansatz wird in Kapitel 4.1 aufgefasst.

3.3.3. Semikontinuierliches System

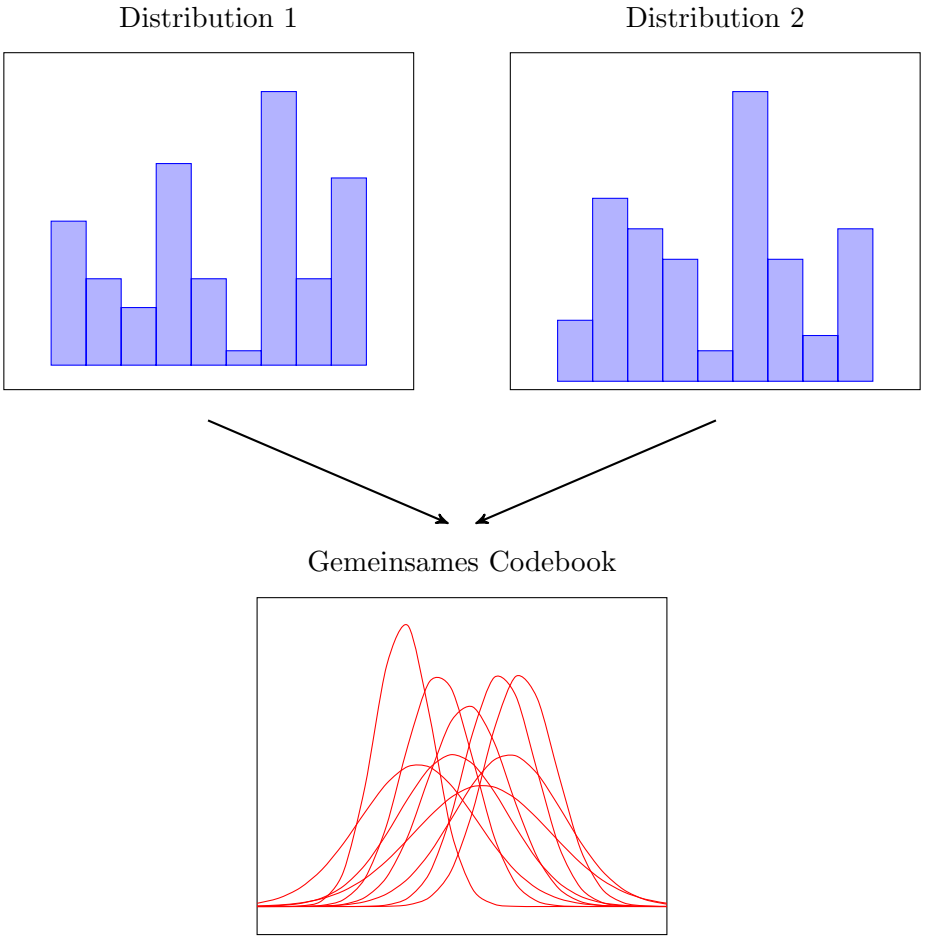


Abbildung 3.4.: Aufbau eines Semikontinuierlichen Systems

Eine Möglichkeit um die Menge der benötigten Trainingsdaten zu verringern besteht darin ein semikontinuierliches System zu erstellen, bei dem mehrere Distributionen einem Codebook zugeordnet werden. Damit bekommen selbige Codebooks mehr Trainingsdaten ab

und sind dementsprechend besser trainiert. Ein Nachteil besteht darin, dass die Codebooks nicht mehr für eine Distribution spezialisiert sind. (siehe Abbildung 3.4)

Dieser Ansatz wird in Kapitel 4.2 aufgefasst.

3.3.4. Phonemclustering

Eine andere Möglichkeit die benötigten Trainingsdaten zu verringern, besteht darin zwei oder mehr Phoneme zu einem neuen Phonem zusammenzufassen. Durch die geringere Anzahl an Phonemen und dazugehörigen Gaußmischverteilungen werden dementsprechend weniger Trainingsdaten verwendet. Dies hat natürlich auch den Nachteil, dass das neue Phonem nicht mehr so stark spezialisiert ist.

Dieser Ansatz wird in Kapitel 4.3 aufgefasst.

4. Konzepte

In diesem Kapitel werden die Methoden, die in dieser Arbeit eingesetzt wurden, genauer beschrieben. Zuerst wird das Merging von Graphemen vorgestellt und danach werden die zwei Clusterverfahren beschrieben.

4.1. Merging von Graphemen

Für das Merging von Graphemen wurde zuerst das FLP Baseline System, (siehe Kapitel 6.1) das auch als Referenzsystem in dieser Arbeit dient, genauer untersucht. Dabei wurde die Anzahl an Merkmalsvektoren pro Vorkommen pro Graphem während des Trainings untersucht. Dabei hat sich heraus gestellt, dass man die Grapheme in zwei Gruppen einteilen kann:

Zum Einen in die Gruppe der Grapheme, die sehr viele Vorkommen mit Merkmalsvektoranzahl 3 haben und danach stark abfallen. (Siehe Abbildung 4.1)

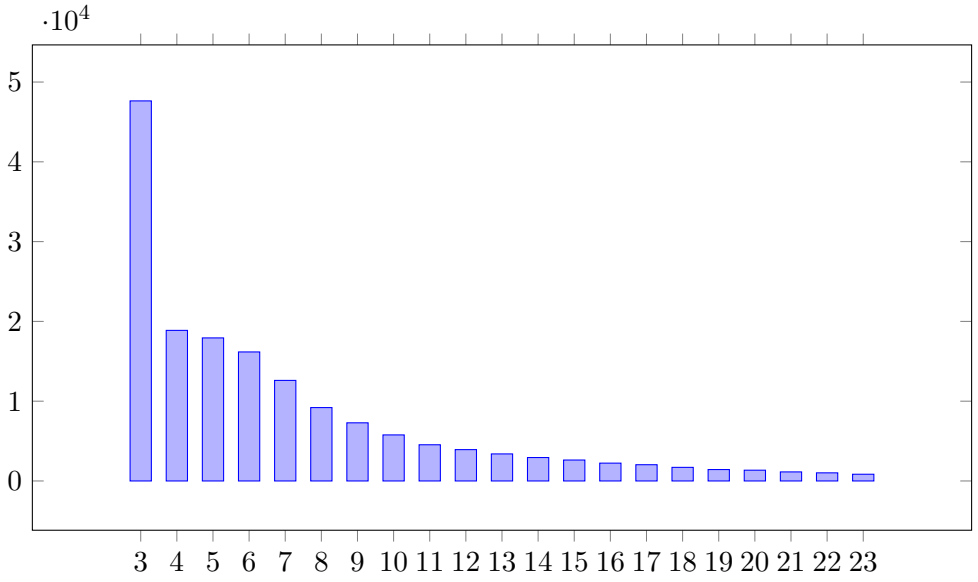


Abbildung 4.1.: Verteilung der Merkmalsvektoren des Graphems E der Sprache Haitianisches Kreolisch

Zum Anderen die zweiten Gruppe, deren Vorkommen an Merkmalsvektoren pro Graphem ungefähr einer Normalverteilung entspricht, wenn Vorkommen der Länge 3 ausgenommen werden. (Siehe Abbildung 4.2)

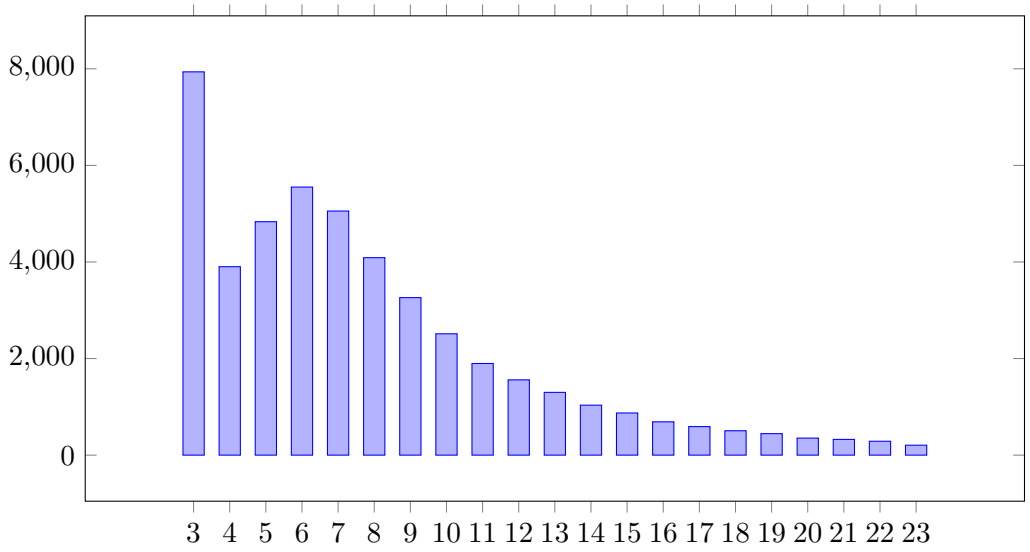


Abbildung 4.2.: Verteilung der Merkmalsvektoren des Graphems È der Sprache Haitianisches Kreolisch

Die Erwartungshaltung vor der Untersuchung war, dass die Verteilungen ungefähr einer Normalverteilung entsprechen, weil Menschen unterschiedlich schnell sprechen oder auch manche Grapheme unterschiedlich schnell in ihrem Kontext ausgesprochen werden. Man muss aber auch beachten, dass in den meisten Sprachen Grapheme existieren, die immer kurz ausgesprochen werden. Als Beispiel kann man hier das Deutsche L heranziehen. Weiterhin legt die Häufung der der Zustandsanzahl drei pro Graphem die Annahme nahe, dass der Spracherkenner das Graphem an dieser Stelle nur durchläuft weil er es muss und nicht weil das Graphem für den Spracherkenner an diese Stelle gehört.

Weiterhin liegt die Annahme nahe, dass die Fehleranfälligkeit mit weniger Merkmalsvektoren zunimmt. Um die Anzahl an Merkmalsvektoren für möglichst viele Vorkommen zu erhöhen, liegt die Idee nahe, zwei Grapheme zu einem neuen zusammenzufassen und damit automatisch eine größere Menge an Merkmalsvektoren zu haben.

Dies hat auch noch eine weitere positive Auswirkung. In vielen Sprachen gibt es Graphemkombinationen, welche anders ausgesprochen werden, als die einzelnen Grapheme. Beispiele hierfür sind die deutschen Buchstabenkombinationen CH oder EI oder im Italienischen der Buchstabe C, der je nachdem welcher Vokal ihm nachfolgt, anders ausgesprochen wird.
Im Folgenden haben wir zwei verschiedene Ansätze für das Merging von Graphemen getestet:

- **Vorkommenbasiertes Merging:** Die häufigsten in den Trainingsdaten auftretenden Graphempaare, werden zu neuen Graphemen zusammengefügt. Dabei wird die Anzahl an Graphempaaren, die zusammengefügt wird, variiert.
- **Merging von Konsonanten:** Wenn man sich verschiedene Sprachen anschaut, fällt auf, dass es viele Sprachen gibt, bei denen die Graphempaare, welche anders ausgesprochen werden, als die einzelnen Grapheme, oft Konsonantenpaare sind und nicht Vokalpaare oder Vokal/Konsonantenpaare. Zulu ist zum Beispiel eine dieser Spra-

chen, welche sehr viele Konsonantenpaare mit anderer Aussprache besitzt. Deswegen muss man je nach Sprache abwägen, ob es sinnvoll ist, nur Konsonaten zusammenzufügen oder auch Vokale in das Merging einzubeziehen.

Bei beiden Verfahren muss man aufpassen, dass man nicht zu viele neue Grapheme einführt, weil sonst die einzelnen Grapheme wegen wenigen Trainingsdaten nicht mehr genug trainiert werden können und damit die Erkennungsrate des Spracherkenners verschlechtern. Deswegen wurden auch beim Merging von Konsonanten die am häufigsten vorkommenden Graphempaare zusammengefügt, welche in diesem Fall Konsonantenpaaren entsprechen.

4.2. Codebook-Clustering

Für das Codebook-Clustering wird ein trainiertes System als Grundlage benötigt. Beim Codebook-Clustering wurden dafür die LLP Baseline Systeme (siehe Kapitel 6.1) verwendet.
Es wurden in dieser Arbeit zwei verschiedene Codebook-Clusterverfahren getestet. Zum Einen das rein distanzbasierte Clustering, bei dem auf Basis der Kullback-Leibler-Divergenz (siehe Kapitel 2.2.2.3) die ähnlichsten Codebooks geclustert werden, und zum Anderen ein auf den Vorkommen der Codebooks in den Trainingsdaten basiertes Clustering. Dabei werden für die Codebooks, die am seltensten trainiert wurden, entsprechend besser trainierte, ähnliche Codebooks gesucht, mit denen sie geclustert werden.

Distanzbasiertes Clustering

Beim distanzbasierten Codebook-Clustering werden mit Hilfe der Kullback-Leibler-Divergenz (siehe Kapitel 2.2.2.3) die Distanzen zwischen allen Distributionen des Baseline Systems berechnet und anschließend das Minimum bestimmt. Das Minimum entspricht dabei, den beiden Distributionen, die am ähnlichsten zueinander sind.

Algorithm 1 Distanzbasiertes Codebook-Clustering

1: $clusterAmount \leftarrow$ amount of codebooks to be clustered
2: **for** $i \leftarrow 1, clusterAmount$ **do**
3: **for all** distributions **do** ▷ ds1
4: **for all** distributions **do** ▷ ds2
5: $distanceArray[ds1][ds2] \leftarrow kldist(d1, d2)$ ▷ kldist(ds1, ds2) method to calculate the distance between 2 distributions
6: **end for**
7: **end for**
8: Get Distributions(ds1,ds2) with Minimum in $distanceArray$
9: $cb1 \leftarrow$ codebook for $ds1$
10: $cb2 \leftarrow$ codebook for $ds2$
11: $cbNew.parameter \leftarrow \frac{1}{2} cb1.parameter \cdot cb2.parameter$
12: map $ds1$ to $cbNew$
13: map $ds2$ to $cbNew$
14: **end for**
15: save distributions and codebooks

Für die entsprechenden Distributionen wird ein neues Codebook erstellt, auf das die Distributionen verweisen. Die Parameter des neuen Codebooks werden dabei basierend auf den Parametern der alten Codebooks initialisiert. Dabei wird für die Berechnung der neuen

Parameter folgende Formel verwendet:

$$\begin{aligned} &p(C_{neu}) = \frac{p(C_1) + p(C_2)}{2} \\ &p(C_{neu}): \text{ Parameter des neuen Codebooks} \\ &p(C_1), p(C_2): \text{ Parameter der alten Codebooks} \end{aligned} \tag{4.1}$$

Dabei entsprechen die Parameter der Codebooks den vorher trainierten Gauß-Verteilungen. Die Gewichtungen, die in den Distributionen gespeichert werden, werden nicht verändert. Sie werden auf die neu berechneten Parameter angewandt.

Das vorgestellte Verfahren wird dabei so oft wiederholt bis die Anzahl an Codebooks, die geclustert werden sollen, erreicht ist. Auf die neuen Codebooks und die darauf verweisenden Distributionen aufbauend, wird das neue semikontinuierliche System trainiert. Algorithmus 1 enthält den Pseudocode zum distanzbasierten Codebook-Clustering.

Vorkommenbasiertes Clustering

Beim vorkommenbasierten Clustering werden die Distributionen und dazugehörigen Codebooks bestimmt, die am wenigsten in den Trainingsdaten vorkommen. Diese Codebooks wurden damit offensichtlich am wenigsten trainiert. Die Menge der Codebooks, die geclustert werden sollen, wurde dabei auf die Hälfte der Codebooks begrenzt, um zu verhindern, dass schlecht trainierte Codebooks miteinander geclustert werden und somit keine Verbesserung eintreten kann.

Algorithm 2 Vorkommenbasiertes Codebook-Clustering

```
clusterCodebooks ← codebooks to be clustered.
for all clusterCodebooks do                                   ▷ clusterCodebook
    ds1 ← inherent distribution
    minDistance ← ∞
    for all distributions not mapping to clusterCodebooks do   ▷ distribution
        distance ← kldist(ds1, distribution)   ▷ kldist(ds1, ds2) method to calculate
    the distance between 2 distributions
        if distance ≤ minDistance then
            minDistance ← distance
            ds2 ← distribution
        end if
    end for
    newCodebook ← codebook of ds2
    map ds1 to newCodebook
    delete clusterCodebook
end for
save distributions and codebooks
```

Anschließend wird für jede der ausgewählten Distributionen die Distanz zu allen andren Distributionen (Distributionen, die geclustert werden sollen, werden ausgenommen) berechnet. Grund hierfür ist wie schon zuvor erwähnt, dass wir vermeiden wollen, dass zwei schlecht trainierte Codebooks geclustert werden. Die Distanzberechnung wurde wieder mit Hilfe der Kullback-Leibler-Divergenz 2.2.2.3 durchgeführt.

Im Clusterschritt werden die Distributionen der schlecht trainierten Codebooks auf die Codebooks gemappt, welche zu den Distributionen gehören, die basierend auf der Distanzberechnung die kleinste Distanz vorweisen.

Dabei wurden bewusst im Gegensatz zum distanzbasierten Clustering, die Parameter des neuen geclusterten Codebooks nicht verändert, weil wir nicht die Parameter eines gut trainierten Codebooks mit dem eines schlecht trainierten Codebooks manipulieren wollen. Wir wollen uns schließlich das besser trainierte Codebook zu Nutze machen.
Auf die Codebooks und die dazugehörigen Distributionen aufbauend, wird das neue semikontinuierliche System trainiert. Algorithmus 2 enthält den Pseudocode zum Verfahren.

4.3. Graphem-Clustering

Für das Graphem-Clustering wird ein trainiertes System als Grundlage benötigt. Beim Graphem-Clustering haben wir dafür die LLP Baseline Systeme (siehe Kapitel 6.1) verwendet.
Zuerst werden beim Graphem-Clustering die Distanzen zwischen allen Graphemen berechnet. Dabei muss man beachten, dass es keine Distanzberechnung gibt, die man direkt auf die Grapheme anwenden könnte, da jedes Graphem aus drei Distributionen zusammengesetzt ist.

Algorithm 3 Graphem-Clustering

1: $clusterAmount \leftarrow$ graphemes to be clustered
2: **for all** graphemes **do** ▷ grapheme1
3: $maG1 \leftarrow$ all distributions of grapheme1 mapped together
4: **for all** graphemes **do** ▷ grapheme2
5: $maG2 \leftarrow$ all distributions of grapheme2 mapped together
6: $distanceArray[maG1][maG2] \leftarrow kldist(maG1, maG2)$ ▷ kldist(maG1, maG2) method to calculate the distance between the mapped distributions
7: **end for**
8: **end for**
9: **for** $i \leftarrow 1, clusterAmount$ **do**
10: $grapheme1, grapheme2 \leftarrow$ graphemepair, with lowest distance
11: delete both graphemes from the phonesSet and add the new combined grapheme
12: add new distributions and codebooks for the new combined grapheme
13: **for all** new distributions **do** ▷ ds
14: $cb \leftarrow$ codebook for ds
15: $cb.parameter \leftarrow \frac{1}{2} cb1.parameter \cdot cb2.parameter$
16: $ds.weights \leftarrow \frac{1}{2} ds1.weights \cdot ds2.weights$
17: **end for**
18: delete all old codebooks and distributions
19: save the codebooks, graphemes and distributions.
20: **end for**

Die Grapheme bestehen dabei aus drei Distributionen, da für die Spracherkennungssysteme 3-state HMMs eingesetzt wurden. 3-state HMMs modellieren dabei Grapheme/Phoneme mit drei Zuständen. Dem Anfang (b), der Mitte (m) und dem Ende (e) der Grapheme/Phoneme.
Die Distanz zwischen den Graphemen wurde basierend auf den aufaddierten Distanzen der Distributionen berechnet.

$$d_{KL-Grapheme}(\mathrm{g}_1, \mathrm{g}_2) = d_{KL}(\mathrm{da}_1, \mathrm{da}_2) + d_{KL}(\mathrm{dm}_1, \mathrm{dm}_2) + d_{KL}(\mathrm{de}_1, \mathrm{de}_2) \tag{4.2}$$

Durch die Vereinfachung auf die Distanz zwischen den Distributionen können wir nun die Distanzen zwischen allen Graphemen mit Hilfe der Kullback-Leibler-Divergenz (siehe Kapitel 2.2.2.3) berechnen. Anschließend werden die Grapheme mit den geringsten Distanzen zueinander geclustert.

Clusterschritt

Im Clusterschritt werden neue Grapheme, Distributionen und Codebooks für die entsprechenden Graphempaare erstellt. Dabei werden den Codebooks und Distribution neue Parameter zugeordnet die wie folgt berechnet werden:

$$\begin{aligned} &p(C_{neu}) = \frac{p(C_1) + p(C_2)}{2} \\ &p(C_{neu}): \text{ Parameter des neuen Codebooks} \\ &p(C_1), p(C_2): \text{ Parameter der alten Codebooks} \end{aligned} \tag{4.3}$$

$$\begin{aligned} &w(D_{neu}) = \frac{w(D_1) + w(D_2)}{2} \\ &w(D_{neu}): \text{ Gewichtungen der neuen Distribution} \\ &w(D_1), w(D_2): \text{ Gewichtungen der alten Distributionen} \end{aligned} \tag{4.4}$$

Danach werden die alten Phoneme, Distributionen und Codebooks gelöscht und alle anderen gespeichert. Aufbauend auf die geclusterten Grapheme wird ein neues System aufgesetzt. Dabei muss beachtet werden, dass das Wörterbuch überarbeitet werden muss, da die jeweilige Repräsentation der Wörter nicht mehr mit den Graphemen übereinstimmt. Algorithmus 3 enthält den Pseudocode zum Graphem-Clustering.

5. Implementierung

Dieses Kapitel beschäftigt sich mit der praktischen Umsetzung der Konzepte, die in Kapitel 4 vorgestellt werden.
Dabei wird in diesem Kapitel das System vorgestellt, das als Grundstock für die Experimente dient. Danach gehen wir auf Einstellungen ein die für das System getroffen wurden und schließlich gibt es noch einen kurzen Überblick über den Ablauf der Experimente und die verwendeten Sprachdaten.

5.1. System

Für die Experimente in dieser Arbeit wurde das JANUS Recognition Toolkit (JRTk)[JRT] verwendet. Es wurde vom Interactive Systems Labs (ISL) mit Sitz am Karlsruher Institut für Technologie (KIT), Deutschland und der Carnegie Mellon University, USA, entwickelt. Das JRTk ist ein Teil des JANUS Speech-to-Speech Translation Systems.

Das JRTk stellt eine flexible, TcL/Tk Script basierte Umgebung zur Verfügung, welche es ermöglicht, Spracherkenner auf dem neuesten Stand der Technik zu bauen und erlaubt neue Methoden zu Entwickeln, Implementieren und diese auch zu Testen. In der Arbeit wurde die Version 5.1.1 des JRTk verwendet.

5.2. Systemeinstellungen

Für alle Systeme wurden sowohl ein kontextunabhängiges System, auf welchem die eigentlichen Experimente durchgeführt wurden, sowie darauf aufbauend noch ein kontextabhängiges System trainiert. Die kontextabhängigen Systeme wurden trainiert um abschätzen zu können, ob die Verbesserungen/Verschlechterungen auch noch im kontextabhängigen System messbar sind. Allgemein wurden für die Systeme folgende Einstellungen gewählt:

- **PhonesSet:** Alle Systeme wurden graphembasiert aufgebaut. Dafür wird im Gegensatz zum phonembasierten Spracherkenner nicht unterschieden zwischen den PhoneSets Konsonanten, Vokalen, ..., sondern jedes Graphem bekommt ein eigenes PhoneSet und stellt sozusagen eine eigene Klasse dar.
- **Anzahl Systeme:** Für das Training der Systeme wurden sechs Trainingszyklen durchgeführt. Das heißt, das System wurde sechs mal trainiert, wobei es in jedem Trainingsschritt auf das vorige System aufbaut.

- **Distanzmaß:** Das Distanzmaß, dass in dieser Arbeit verwendet wird um die Distanz bei den Clusterverfahren zu berechnen, ist die Kullback-Leibler-Divergenz. (siehe Kapitel 2.2.2.3)
- **Anzahl an Gaußschen Mischverteilungen:** Die Anzahl der Gaußschen Mischverteilungen pro Distribution/Codebook wurde auf 128 festgelegt.
- **Anzahl an Merkmalsvektoren:** Für die Anzahl an Merkmalsvektoren, die von den Gaußschen Mischverteilungen modelliert werden, wurden 42 ausgewählt. Das heißt, die Gaußschen Mischverteilungen bestehen aus 42 Gauß-Verteilungen.
- **Sampling Rate:** Das Signal wurde mit einer Sampling Rate von 8 ms abgetastet.

Für das kontextabhängige System wurden noch folgende zusätzliche Einstellungen vorgenommen:

- **Anzahl an Splits:** Die Anzahl an Splits für das Clustering des kontextabhängigen Systems wurde auf 500, 1000, 1500, 2000 und 2500 festgelegt. Alle kontextabhängigen Systeme mit den entsprechenden Splits wurden getestet und bewertet. Die kontextabhängigen Systeme mit mehr Splits wurden weggelassen, da keine nennenswerten Verbesserungen erzielt werden konnten.
- **Mindestanzahl an Samples pro Split:** Bevor ein Split ausgeführt wird, wird geprüeft ob noch genug Samples zum Training der jeweiligen Splits übrig bleiben. Die Grenze wurde dabei auf 100 Samples pro Split gesetzt.

5.3. Ablauf der Experimente

Mit den zuvor genannten Systemeinstellungen wurden die Experimente durch folgende Schritte trainiert und getestet. Dabei wird zwischen den verschiedenen Methoden unterschieden:

Merging

1. **Graphemanalyse:** Auf Basis des FLP Baseline Systems werden die Grapheme untersucht. Dabei werden die Vorkommen der Graphempaare bestimmt.
2. **Graphemmerging:** Auf Basis der Vorkommen werden die Graphempaare ausgewählt, die gemergt werden sollen. Die ausgewählten Graphempaare werden zu den anderen Graphemen hinzugefügt.
3. **Modifizierung des Wörterbuchs:** Das Wörterbuch muss auf die neuen Graphempaare angepasst werden, ansonsten könnte der Spracherkenner keine Worte erkennen, in denen die neuen Graphempaare vorkommen, beziehungsweise die neuen Graphempaare würden nicht trainiert werden und es würde dem Baseline System mit ein paar ungenutzten, untrainierten Graphemen entsprechen.
4. **Training eines kontextunabhängigen Systems:** Auf Basis des neuen Wörterbuchs wird ein komplett neues System aufgesetzt, bei dem alle Codebooks und Distributionen zunächst untrainiert sind. Zum Training des Systems werden die FLP Trainingsdaten verwendet.
5. **Training eines kontextabhängigen Systems:** Aufbauend auf das kontextunabhängige System wird ein kontextabhängiges System erstellt und trainiert. Zum Training des Systems werden die FLP Trainingsdaten verwendet.
6. **Test beider Systeme:** Das kontextunabhängige, sowie das kontextabhängige System werden mit Hilfe der Testdaten getestet und mit Hilfe der Word Error Rate bewertet.

Codebook-Clustering

1. **Distanzberechnung:** Auf Basis des LLP Baseline Systems wird eine Distanzberechnung zwischen den Distributionen ausgeführt.
2. **Clustering:** Auf Basis der Distanzberechnung wird das Codebook-Clustering ausgeführt. Die neuen Distributionen und Codebooks werden anschließend gespeichert.
3. **Training eines kontextunabhängigen Systems:** Auf die neuen Codebooks und Distributionen aufbauend, wird ein neues kontextunabhängiges System trainiert. Dabei ist wichtig, dass alle Skripte die verwendet werden, mit einer 1:n Beziehung zwischen Codebooks und Distributionen umgehen können. Gegebenenfalls müssen diese modifiziert werden. Zum Training des Systems werden die LLP Trainingsdaten eingesetzt.
4. **Training eines kontextabhängigen Systems:** Aufbauend auf das kontextunabhängige System wird noch ein kontextabhängiges System trainiert. Dabei muss wie beim kontextunabhängigen System aufgepasst werden, dass die Skripte mit 1:n Beziehungen zwischen Codebooks und Distributionen umgehen können. Gegebenenfalls müssen diese modifiziert werden. Zum Training des Systems werden die LLP Trainingsdaten eingesetzt.
5. **Test beider Systeme:** Das kontextunabhängige, sowie das kontextabhängige System werden mit Hilfe der Testdaten getestet und mit Hilfe der Word Error Rate bewertet.

Graphem-Clustering

1. **Distanzberechnung:** Auf Basis des LLP Baseline Systems wird eine Distanzberechnung zwischen den Graphemen ausgeführt.
2. **Clustering:** Auf Basis der Distanzberechnung wird das Graphem-Clustering ausgeführt. Die neuen Distributionen, Codebooks und Grapheme werden anschließend gespeichert.
3. **Modifizierung des Wörterbuchs:** Das Wörterbuch muss auf die neuen Grapheme angepasst werden. Ansonsten würden alle Wörter, die aus Graphemen die geclustert wurden, bestehen, nicht mehr erkannt werden, da die entsprechenden Grapheme nicht mehr existieren, sondern nur noch das geclusterte Graphem.
4. **Training eines kontextunabhängigen Systems:** Aufbauend auf die neuen Grapheme, Distributionen, Codebooks und das neue Wörterbuch wird ein neues kontextunabhängiges System trainiert. Zum Training werden die LLP Trainingsdaten eingesetzt.
5. **Training eines kontextabhängigen Systems:** Aufbauend auf das kontextunabhängige System wird noch ein kontextabhängiges System trainiert. Zum Training werden die LLP Trainingsdaten eingesetzt.
6. **Test beider Systeme:** Das kontextunabhängige, sowie das kontextabhängige System werden mit Hilfe der Testdaten getestet und mit Hilfe der Word Error Rate bewertet.

5.4. Daten

Die Experimente wurden auf zwei verschiedenen Sprachen getestet. Zum Einen Haitianisches Kreolisch und zum Anderen Zulu. Beide Sprachen wurden auf Konversationssprache

trainiert und getestet. Dabei wurden jeweils zwei verschiedene Datenpakete verwendet. Ein Datenpaket mit den gesamten Sprachdaten (Full Language Pack - FLP) und ein Datenpaket mit reduzierten Sprachdaten. (Limited Language Pack - LLP) Die Datenpakete werden vom IARPA BABEL [IAR] Programm zu Verfügung gestellt. Das Acknowledgment für die Trainingsdaten befindet sich in Anhang A.

Haitianisches Kreolisch

Haitianisches Kreolisch ist eine auf Französisch basierte Kreolische Sprache. Daher verwundert es auch nicht, dass Haitianisches Kreolisch und Französich die zwei offiziellen Sprachen in Haiti sind. Da Haitianisches Kreolisch zum Großteil auf Französisch aufbaut, ist ein Großteil des Lexikons aus dem Französischen abgeleitet. [MA14] Sie wird von zwölf Millionen Menschen gesprochen. [HaK]

Haitianisches Kreolisch wurde wegen seiner Ähnlichkeit zum Französischen für diese Arbeit verwendet. Wie schon erwähnt wurden zwei verschiedene Datenpackete verwendet. Zum Einen den FLP mit 100 Stunden Trainingsdaten und zum Anderen den LLP mit 10 Stunden Trainingsdaten. Die Trainingsdaten enthalten dabei drei verschiedene Dialekte. Northern HC, Western HC und Soutern HC.

Zulu

Zulu ist eine von elf offiziellen Sprachen Südafrikas. Ungefähr 23% der Südafrikaner haben Zulu als ihre Muttersprache, was den größten Anteil in Südafrika ausmacht. Weiterhin wird Zulu nicht nur in Südafrika gesprochen. Auch in Botswana, Lesotho, Malawi, Mozambique und Swasiland wird Zulu gesprochen. In diesen Ländern aber hauptsächlich als Zweitsprache und nicht als Muttersprache. [dS06] Zulu wird von ungefähr elf Millionen Menschen gesprochen. [Zul] Zulu wurde wegen seiner Klicklaute für diese Arbeit verwendet. Wir wollten unter andrem schauen was für einen Einfluss die Klicklaute [Hal12] auf die Spracherkennung haben.

Wie schon erwähnt wurden zwei verschiedene Datenpackete verwendet. Zum Einen den FLP mit 100 Stunden Trainingsdaten und zum Anderen den LLP mit 10 Stunden Trainingsdaten. Alle Trainingsdaten stammen dabei von Muttersprachlern.

Bei Untersuchungen der Sprachdaten wurde bei beiden Sprachen festgestellt, dass die Sprachdaten sehr stark verrauscht sind und deswegen keine sehr guten Ergebnisse zu erwarten sind.

6. Evaluierung

In diesem Kapitel werden die Ergebnisse vorgestellt, die durch die Experimente erzielt wurden. Für die Bewertung der Systeme wurde die Word Error Rate (siehe Kapitel 2.3) verwendet.
Zuerst werden die Ergebnisse der Baseline Systeme vorgestellt, welche als Referenzsysteme für die anderen Systeme dienen. Danach wird auf die Ergebnisse der Experimente eingegangen, wobei diese immer in Relation zu den Baseline Systemen gesetzt werden.

6.1. Baseline Systeme

Insgesamt wurden zwei Baseline Systeme für jede Sprache erstellt: Ein System mit vielen Sprachdaten (FLP Baseline System) und ein System mit wenig Sprachdaten. (LLP Baseline System) Weitergehende Informationen in Kapitel 5.4.
Die Baseline Systeme dienen dabei zum Einen als Referenzsysteme für die modifizierten Systeme, um abschätzen zu können, wie gut die Ergebnisse unserer Experimente sind, und zum Anderen als Grundstock, um unsere anderen Systeme darauf aufzubauen. Unser Ziel in dieser Arbeit ist es, mit unseren Experimenten bessere Ergebnisse zu erzielen, als es die Baseline Systeme tun.

Haitianisches Kreolisch

Bei Haitianisches Kreolisch liegt die Word Error Rate für das FLP Baseline System bei 86,1% für das kontextunabhängige System und bei 78,6% für das kontextabhängige System. Für das LLP System wurde eine Word Error Rate von 88,6% für das kontextunabhängige System und von 83,4% für das kontextabhängige Systeme erzielt.

Zulu

Bei Zulu liegt die Word Error Rate für das FLP Baseline bei 90,7% für das kontextunabhängige System und bei 80,4% für das kontextabhängige System. Für das LLP System wurde eine Word Error Rate von 93,4% für das kontextunabhängige und von 87,0% für das kontextabhängige System erzielt.

Die schlechten Word Error Rates sind dabei auf die stark verrauschten Trainingsdaten zurückzuführen. (siehe Kapitel 5.4)

6.2. Merging von Graphemen

Für das Merging von Graphemen wurden die FLP Trainingsdaten verwendet. Das heißt, dieses System wird mit dem FLP Baseline System verglichen.
Es wurden insgesamt vier verschiedene Systeme erstellt. Zum Einen wurden drei Systeme für das vorkommenbasierte Merging erstellt, bei denen die Anzahl an neuen Graphemen 5, 10 und 15 beträgt (Systemnamen: Merge5, Merge10, Merge15) und zum Anderen wurde ein System für das Merging von Konsonanten erstellt, bei dem fÃ¼nf Konsonantenpaare zu neuen Graphemen gemergt wurden (Systemnamen: MergeKons). Die Grapheme für die entsprechenden Systeme befinden sich im Anhang B.

Haitianisches Kreolisch

Die Testergebnisse für die Sprache Haitianisches Kreolisch sind in Tabelle 6.1 dargestellt. CI (context independent) steht dabei für die kontextunabhängingen Systeme und CD (context dependent) für die kontextabhängingen Systeme, mit der entsprechenden Anzahl an Splits.

System	CI	CD 500	CD 1000	CD 1500	CD 2000	CD 2500
Baseline	86,1	81,8	80,1	79,3	78,9	**78,6**
Merge5	84,7	81,4	79,7	78,9	78,3	**78,0**
Merge10	84,4	80,3	79,2	78,5	78,4	**77,8**
Merge15	85,2	81,6	80,1	79,2	78,8	**78,7**
MergeKons	87,6	82,5	80,8	80,2	79,9	**79,4**

Tabelle 6.1.: WER für Haitianisches Kreolisch. Alle Angabe in %

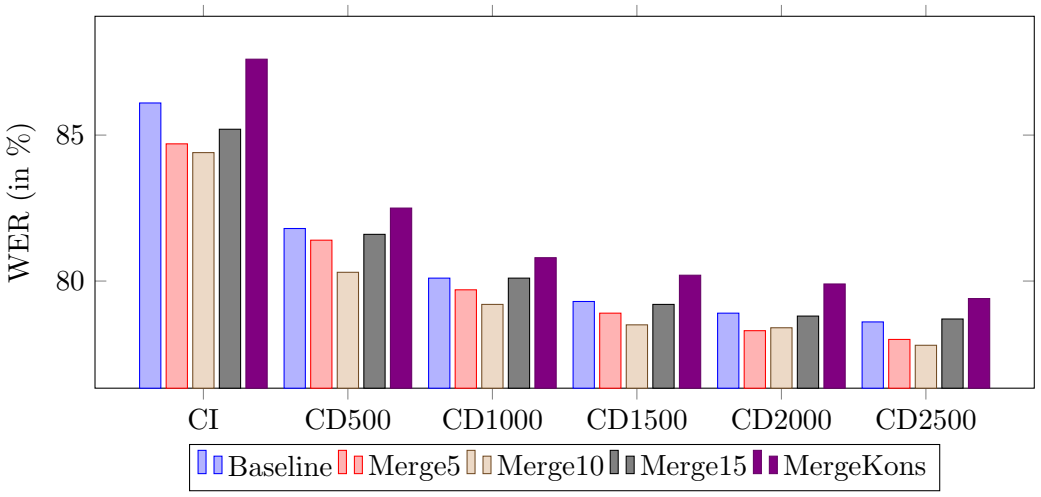


Abbildung 6.1.: Blockdiagramm der WER für Haitianisches Kreolisch

Wie man im Blockdiagramm 6.1 ablesen kann, hat sich das System, bei dem zehn Graphempaare zusammengefügt wurden, als am Besten herausgestellt. Dabei muss man aber beachten, dass die Verbesserung der Erkennungsrate mit der höheren Anzahl an Splits im kontextabhängigen System abnimmt, aber dennoch bessere Ergebnisse liefert als das Baseline System. Eine weitere interessante Beobachtung ist, dass die Erkennungsrate mit 15 Graphempaaren schon wieder abnimmt. Das bedeutet die Anzahl der neu eingeführten Grapheme muss sorgfältig gewählt werden.

Das System, dass nur Konsonantenpaare zusammenfügt, ist dagegen beim Haitianischen Kreolisch als ungeeignet zu bezeichnen. Bei diesem Merging nimmt die Erkennungsrate sogar im Vergleich zum Baseline System ab. Für die kontextunabhängingen Systeme ist noch anzumerken, dass alle rein vorkommenbasierten Systeme eine bessere Erkennungsrate haben, als das Baseline System.

Zulu

Die Testergebnisse für die Sprache Zulu sind in Tabelle 6.2 dargestellt. CI (context independent) steht dabei für die kontextunabhängingen Systeme und CD (context dependent) für die kontextabhängingen Systeme, mit der entsprechenden Anzahl an Splits.

System	CI	CD 500	CD 1000	CD 1500	CD 2000	CD 2500
Baseline	90,7	84,2	81,9	81,2	80,5	**80,4**
Merge5	89,8	84,2	82,2	81,2	80,7	**80,5**
Merge10	88,9	84,1	82,2	81,1	80,9	**80,8**
Merge15	88,7	85,2	82,7	82,3	82,0	**81,3**
MergeKons	89,3	83,4	81,1	80,4	79,7	**79,5**

Tabelle 6.2.: WER für Zulu. Alle Angabe in %

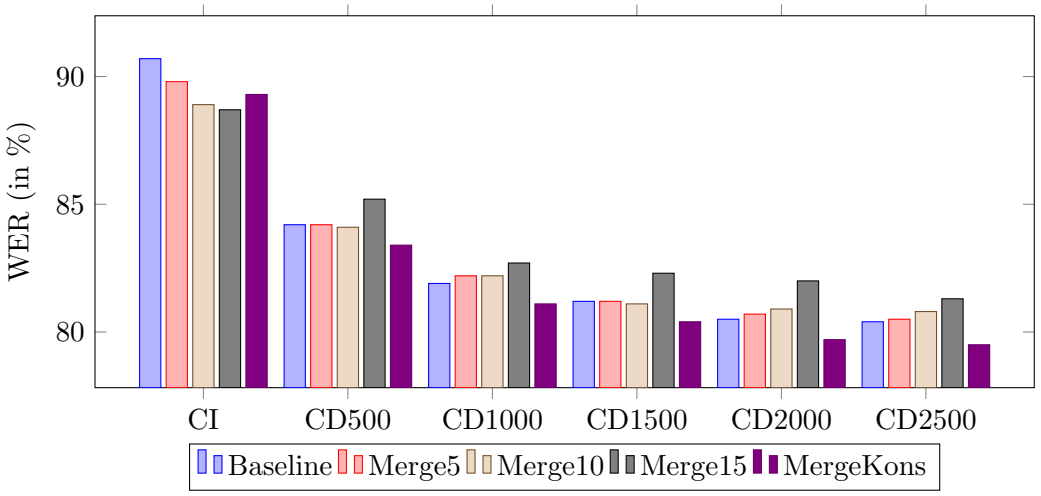


Abbildung 6.2.: Blockdiagramm der WER für Zulu

Wie man im Blockdiagramm 6.2 ablesen kann, haben alle kontextunabhäningen Systeme eine bessere Erkennungsrate erreicht, als das Baseline System. Wobei hier das Merging mit 15 Graphemen am Besten abschneidet. Bei den kontextabhängigen Systemen verändert sich das Bild. Dabei sind die Systeme des vorkommenbasierten Merging alle im Bereich des Baseline Systems anzuordnen, wobei das Konsonantenmerging besser abschneidet.

Generell kann man sagen, dass das Merging von Graphemen bessere Ergebnisse als die Baseline Systeme geliefert hat, wenn man die richtige Methode mit der richtigen Anzahl an Graphempaaren, für die neuen Systeme kombiniert. Ein großer Unterschied lässt sich dabei zwischen den beiden Sprachen feststellen. Haitianisches Kreolisch profitiert scheinbar hauptsächlich vom rein vorkommenbasierten Merging, wohingegen bei Zulu das konsonantenbasierte Merging die besseren Ergebnisse erzielt.

Dies kann durchaus mit dem Aufbau der Sprachen zu tun haben, da Zulu einige Konsonantenpaare besitzt, bei denen die einzelnen Konsonanten anders ausgesprochen werden, als die Paare zusammen. Beispiele hierfür sind KL, NG, TH und PH. Dies ist im Haitianischen Kreolisch nicht der Fall. Weiterhin werden die Klicklaute in Zulu durch ein Konsonantenpaar modelliert. Bei unseren Graphempaaren kommen diese aber nicht vor, da die Klicklaute zu selten auftreten.

6.3. Codebook-Clustering

Für das Codebook-Clustering wurden die LLP Trainingsdaten verwendet. Das heißt, das System wird auch mit dem LLP Baseline System verglichen.
Es wurden insgesamt vier Systeme erstellt: Drei mit dem rein distanzbasierten Clustering, bei denen die Anzahl der Clusterschritte 5, 10 und 20 beträgt (Systemnamen: Cluster5, Cluster10, Cluster20) und ein System, dass die Codebooks mit den wenigsten Trainingsdaten clustert, wobei zehn Clusterschritte durchgeführt wurden (Systemname: Cluster10.2). Die geclusterten Codebooks für die Systeme befinden sich im Anhang C.

Haitianisches Kreolisch

Die Testergebnisse für die Sprache Haitianisches Kreolisch sind in Tabelle 6.3 dargestellt. CI (context independent) steht dabei für die kontextunabhängingen Systeme und CD (context dependent) für die kontextabhängingen Systeme mit der entsprechenden Anzahl an Splits.

System	CI	CD 500	CD 1000	CD 1500	CD 2000	CD 2500
Baseline	88,6	85,0	83,9	83,6	83,6	**83,4**
Cluster5	89,1	84,9	84,2	84,2	84,1	**84,0**
Cluster10	89,1	85,3	84,6	**83,9**	84,0	84,2
Cluster20	89,5	85,5	85,2	**84,8**	**84,8**	**84,8**
Cluster10.2	88,7	85,1	84,1	83,9	**83,8**	**83,8**

Tabelle 6.3.: WER für Haitianisches Kreolisch. Alle Angabe in %

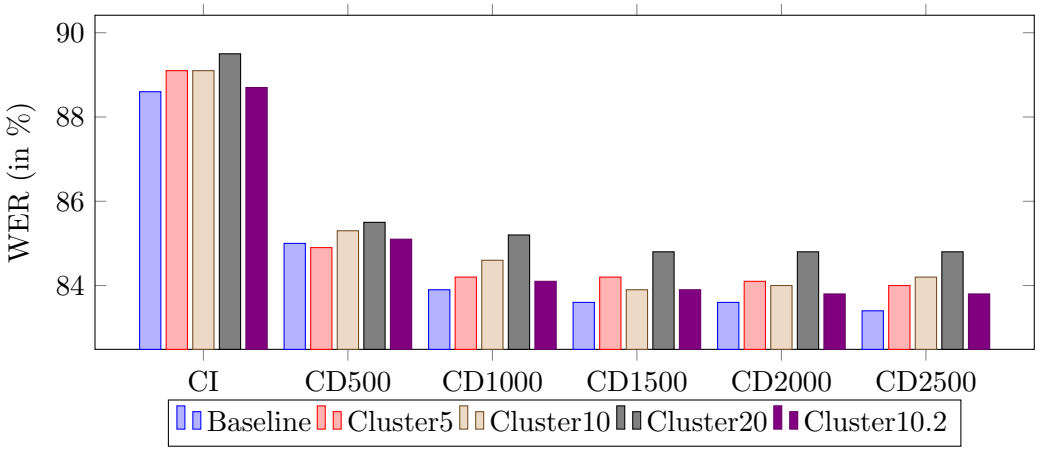


Abbildung 6.3.: Blockdiagramm der WER für Haitianisches Kreolisch

Wie man im Blockdiagramm 6.3 ablesen kann, haben alle Systeme eine schlechtere Erkennungsrate, als das Baseline System, wobei das System bei dem die Codebooks, die wenig Trainingsdaten abbekommen, geclustert werden, noch am Besten abschneidet.

Zulu

Die Testergebnisse für die Sprache Zulu sind in Tabelle 6.4 dargestellt. CI (context independent) steht dabei für die kontextunabhängingen Systeme und CD (context dependent) für die kontextabhängingen Systeme, mit der entsprechenden Anzahl an Splits.

System	CI	CD 500	CD 1000	CD 1500	CD 2000	CD 2500
Baseline	93,4	88,5	89,0	87,2	**86,9**	87,0
Cluster5	92,3	88,0	86,8	86,2	**86,0**	**86,0**
Cluster10	92,8	87,9	86,7	86,3	**86,2**	86,3
Cluster20	92,3	87,9	86,8	86,3	86,6	**86,2**
Cluster10.2	92,2	88,1	87,0	86,4	**86,1**	86,2

Tabelle 6.4.: WER für Zulu. Alle Angabe in %

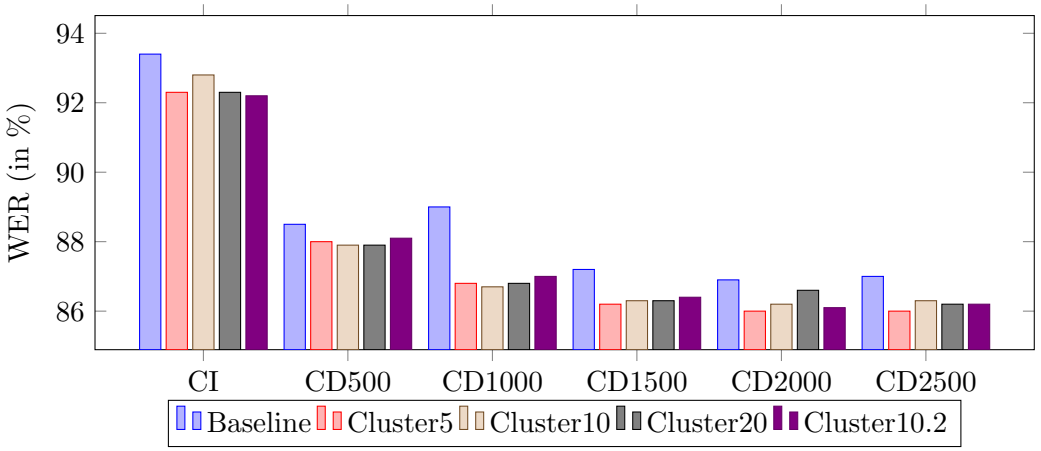


Abbildung 6.4.: Blockdiagramm der WER für Zulu

Wie man im Blockdiagramm 6.4 ablesen kann, sind die Clustering Systeme durchgehend leicht besser, als das Baseline System. Erstaunlicherweise sind dabei die Ergebnisse zwischen den ganzen semikontinuierlichen Systemen sehr ähnlich.

Zusammengefasst kann man sagen, dass das Codebook-Clustering beim Haitianischen Kreolisch keine besseren Ergebnisse geliefert hat, bei Zulu dagegen schon. Gründe hierfür können vielfältig sein. Es kann an den stark verrauschten Daten liegen, es kann daran liegen, dass für Zulu sinnvollere Codebooks zusammen geclusert worden sind, oder dass beim Berechnen der Gauß-Verteilungen der neuen Codebooks die Durchschnittswerte bei Zulu besser zu den perfekten Werten der Codebooks passen, oder bei Haitianischen Kreolisch die Parameterberechnung einen negativen Einfluss hatte.

Generell ist es aber schwierig zu sagen, warum Zulu besser abschneidet als Haitianisches Kreolisch. Generell kann man bei anderen Sprachen durchaus versuchen ein Codebook-Clustering einzusetzen, wenn wenig Trainingsdaten vorhanden sind, da es zu einer Verbesserung der Erkennungsrate führen kann, und auch in unserem semikontinuierlichen System noch viele Möglichkeiten zur Verbesserung bestehen. Eine Möglichkeit wäre zum Beispiel die Berechnung der Parameter des neuen Codebooks zu verfeinern.

6.4. Graphem-Clustering

Für das Graphem-Clustering wurden die LLP Trainingsdaten verwendet. Das heißt, das System wird auch mit dem LLP Baseline System verglichen.
Es wurde ein System für das Graphem-Clustering erstellt, bei dem die Zahl der Grapheme insgesamt um drei reduziert wurde. (Systemname: GraphemCluster) Im Anhang B befinden sich die Grapheme des Systems.

Haitianisches Kreolisch

Die Testergebnisse für die Sprache Haitianisches Kreolisch sind in Tabelle 6.5 dargestellt. CI (condext independent) steht dabei für die kontextunabhängingen Systeme und CD (context dependent) für die kontextabhängingen Systeme mit der entsprechenden Anzahl an Splits.

System	CI	CD 500	CD 1000	CD 1500	CD 2000	CD 2500
Baseline	88,6	85,0	83,9	83,6	83,6	**83,4**
GraphemCluster	89,4	85,8	84,7	84,3	**84,2**	**84,2**

Tabelle 6.5.: WER für Haitianisches Kreolisch. Alle Angabe in %

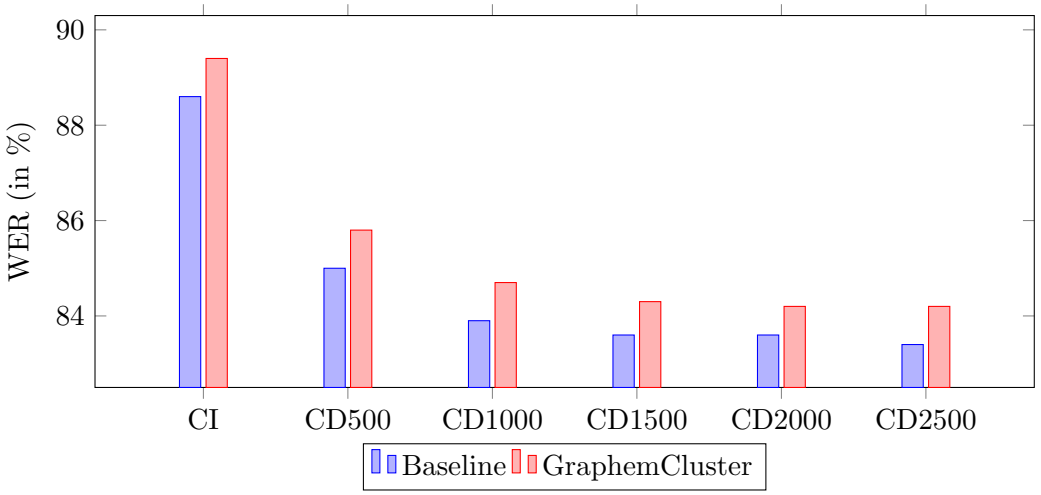


Abbildung 6.5.: Blockdiagramm der WER für Haitianisches Kreolisch

Wie man im Blockdiagramm 6.5 ablesen kann, liegt die Erkennungsrate etwas niedriger als die des Baseline Systems.

Zulu

System	CI	CD 500	CD 1000	CD 1500	CD 2000	CD 2500
Baseline	93,4	88,5	89,0	87,2	**86,9**	87,0
GraphemCluster	93,6	88,9	88,1	87,4	87,4	**87,2**

Tabelle 6.6.: WER für Zulu. Alle Angabe in %

Die Testergebnisse für die Sprache Zulu sind in Tabelle 6.6 dargestellt. CI (context independent) steht dabei für die kontextunabhängingen Systeme und CD (context dependent) für die kontextabhängingen Systeme mit der entsprechenden Anzahl an Splits.

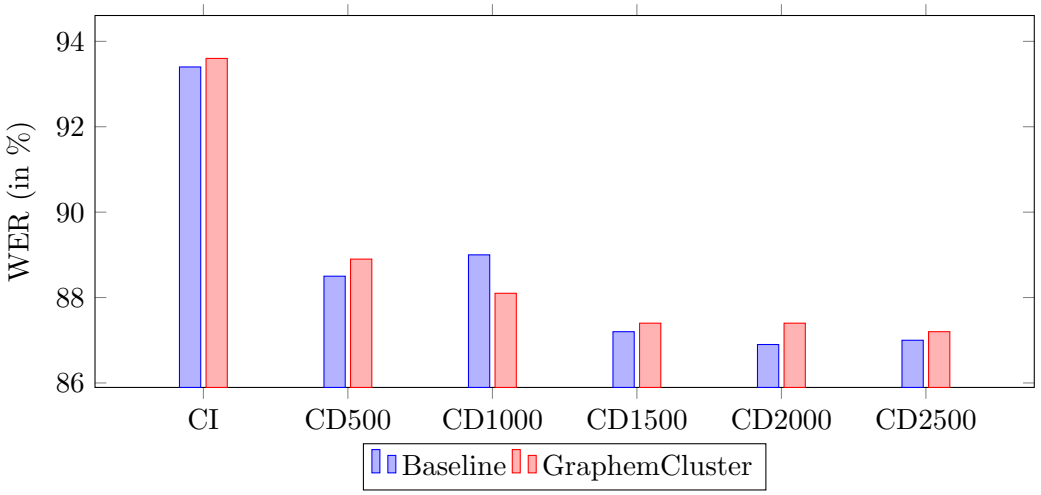


Abbildung 6.6.: Blockdiagramm der WER für Zulu

Wie man im Blockdiagramm 6.6 ablesen kann, liegt die Erkennungsrate etwas niedriger als die des Baseline Systems.

Das Graphem-Clustering hat bei beiden Sprachen schlechtere Ergebnisse erzielt, als die Baseline Systeme. Dies ist durchaus auch erwartet worden. Die Ähnlichkeit zwischen zwei verschiedenen Graphemen muss schon sehr stark sein, damit dieser Ansatz erfolgversprechend ist. In manchen Sprachen, kann dieser Ansatz durchaus Sinn machen, wenn wie erwähnt, manche Grapheme sehr ähnlich sind und wenige Trainingsdaten vorhanden sind. Dies ist bei Zulu und Haitianischem Kreolisch nicht der Fall.

7. Zusammenfassung und Ausblick

7.1. Zusammenfassung

In dieser Arbeit wurden verschiedene Methoden für graphembasierte Spracherkenner entwickelt, implementiert und schließlich getestet. Dabei haben wir drei verschiedene Methoden mit verschiedenen Modifikationen getestet. Zum Ersten ein Merging von Graphemen, bei dem Graphempaare als neue Grapheme eingeführt werden und darauf basierend Systeme trainiert wurden. Dabei werden zwei verschiedene Ansätze verwendet. Einmal der rein vorkommenbasierte Ansatz und zum Anderen der konsonantenbasierte Ansatz.

Zum Zweiten ein Codebook-Clustering, bei dem semikontinuierliche Systeme getestet werden. Dabei werden zwei verschiedene Ansätze verwendet. Zum Einen der rein distanzbasierten Ansatz und zum Anderen einen Ansatz, der auf den Vorkommen der Codebooks basiert. Zum Dritten ein Graphem-Clustering, bei dem zwei oder mehr Grapheme zu einem einzelnen Graphem geclustert werden.

Wie wir in der Evaluierung in Kapitel 6 sehen, hat vor allem das Merging von Graphemen die besten Ergebnisse geliefert. Dabei wurde eine absolute Verbesserung von bis zu 1,7% erreicht. Dabei ist anzumerken, dass beim Haitianischen Kreolisch das vorkommenbasierte Merging bessere Ergebnisse erzielt hat und bei Zulu das konstantenbasierte Merging.

Beim Codebook-Clustering konnte eine absolute Verbesserung von bis zu 1,2% erreicht werden, wobei dies nur bei der Sprache Zulu der Fall war. Beim Haitianischen Kreolisch war dies nicht der Fall. Dort hat das Codebook-Clustering durchweg leicht schlechtere Ergebnisse erzielt. Bei der dritten Methode, dem Graphem-Clustering, wurden leicht schlechtere Erkennungsraten erreicht als in den Referenzsystemen.

7.2. Diskussion

Am Anfang der Diskussion müssen wir zuerst auf die Trainingsdaten eingehen. Die Trainingsdaten, mit denen in dieser Arbeit gearbeitet wurde, sind sehr stark verrauscht und erschweren es gute Ergebnisse zu erzielen. Wie schon in der Einleitung erwähnt wurde, kann ein Spracherkenner immer nur so gut sein, wie die Daten die er für das Training zur Verfügung gestellt bekommt. Wie auch schon erwähnt kann das für viele weniger verbreitete Sprachen, wie Zulu und Haitianisches Kreolisch ein großes Problem darstellen. Es

stehen einfach deutlich weniger Daten zur Verfügung, als in stark verbreiteten wie zum Beispiel Englisch. Dadurch leidet auch die Qualität der Daten, da man zum Einen aus weniger Daten auswählen kann und zum Anderen auch noch das zweite Problem dazu kommt, dass zum Beispiel in Haiti nicht so gute Aufnahmetechnik vorhanden ist, wie dies zum Beispiel im Europäischen Parlament der Fall ist.

Durch die schlechte Qualität der Trainingsdaten ist es auch schwierig verlässliche Aussagen über die Ergebnisse der Experimente zu treffen. Sind die leichten Verbesserungen oder Verschlechterungen der Erkennungsraten auf wirkliche Verbesserung oder Verschlechterung der Spracherkenner zurückzuführen, oder würde der Trend bei besseren Daten genau umgekehrt aussehen. Gerade bei Ergebnissen wie 93,4% für das LLP Baseline System von ZULU und die darauf aufbauenden Systeme ist es dabei schwer zu sagen, ob wirklich eine sinnvolle Erkennung gemacht wird, oder ob einige Treffer durch "Zufall" oder hauptsächlich durch das Language Model zustande kommen.

Wenn wir unter den Gesichtspunkten der problembehafteten Daten unsere getesteten Methoden genauer betrachten, kann man trotzdem ein paar Aussagen treffen.
Bei genaueren Betrachten des Mergings von Graphemen, fällt auf, dass durchgehend positive Ergebnisse durch das Merging erzielt wurden, wobei die Anzahl an neuen Graphemen nicht zu groß werden darf, weil sonst die Erkennungsrate wieder abnimmt. Dies kann zwei verschiedene Gründe haben. Die erste Möglichkeit ist diejenige, dass durch Zufall, die am besten geeigneten Graphempaare die meisten Vorkommen haben und dadurch bessere Ergebnisse erzielt werden. Die zweite Möglichkeit ist, dass das Merging von Graphemen nur einen Vorteil liefert, wenn für die entsprechenden Graphempaare auch relativ viele Trainingsdaten vorhanden sind.

Weiterhin wurde festgestellt, dass für manche Sprachen das Merging nur auf Konsonanten bessere Ergebnisse erzielt, als das reine vorkommenbasierte Merging. Die Entscheidung zu treffen, welches Verfahren für eine Sprache angewendet werden soll, kann man dabei scheinbar schon an grundlegenden Informationen über Sprachen festmachen. Zulu zum Beispiel ist eine Sprache, die relativ viele Konsonantenpaare besitzt, die eine eigene Aussprache haben. Dies ist beim Haitianischen Kreolisch nicht der Fall. Dementsprechend sind auch die Ergebnisse für die entsprechenden Sprachen ausgefallen.

Wenn nun ein neuer Spracherkenner für eine neue Sprache trainiert werden soll, können wir uns diese Information zu Nutze machen. Nehmen wir zum Beispiel die Sprache Italienisch. Im Italienischen gibt es nur relativ wenige Graphempaare, die anders ausgesprochen werden als die einzelnen Teile. Bei den Paaren die es gibt, sind dies meistens Kombinationen von einem Vokal und einem Konsonant und keine Konsonantenpaare. Dementsprechend würde es sich für Italienisch anbieten das rein vorkommenbasierte Merging zu verwenden und nicht das konsonantenbasierte Merging.

Beim Codebook-Clustering ist die Sache schwieriger zu bewerten. Auch wenn die Ergebnisse für das Codebook-Clustering eher durchwachsen sind, kann man trotzdem nicht sagen, dass das Codebook-Clustering eine schlechte Idee ist. Zum Einen ergab das Codebook-Clustering für Zulu sogar eine leicht bessere Erkennungsrate und zum Anderen gibt es beim Codebook-Clustering viele Möglichkeiten die Ergebnisse direkt oder indirekt zu beeinflussen.

Dies fängt zum Beispiel schon damit an, dass die Distributionen, auf denen die Distanzberechung stattgefunden hat, wegen der schlechten Daten, schlecht trainiert sind. Das heißt, die eigentlichen Gewichtungen, die manche Codebooks und Distributionen haben sollten

um den entsprechenden Phonemteilen zu entsprechen haben mit den Parametern, die zur Verfügung stehen, vielleicht nicht viel gemeinsam.

Dies kann natürlich dazu führen, dass eigentlich unähnliche Codebooks zusammen geclustert werden. Das ist auch das Problem des Codebook-Clusterings, welches nur die Codebooks mit den geringsten Vorkommen geclustert hat. Hier ist aber genau die Frage, wie sinnvoll es ist, eine Distanzberechnung auf eine Distribution auszuführen, welche nur sehr schlecht trainiert ist, weil hier die Wahrscheinlichkeit groß ist, dass eigentlich sehr unterschiedliche Codebooks geclustert werden.

Weiterhin ist die nächste Frage, ob die Berechnung der neuen Parameter wirklich sinnvoll durchgeführt wird, oder ob man eigentlich alle Gaußmixturen einzeln miteinander vergleichen müsste, und immer aus den ähnlichsten Gaußmixturen die neuen Gaußmixturen für das neue Codebook zu berechnen, anstatt einfach die neuen Gaußmixturen aus den Gaußmixturen mit den gleichen Indizes zu berechnen. Generell gibt es beim Codebook-Clustering noch viele Möglichkeiten genauere Experimente zu tätigen, um das Potenzial genauer abschätzen zu können.

Beim Graphem-Clustering ist das Potenzial auf eine Verbesserung der Erkennungsrate deutlich niedriger als bei den anderen beiden Methoden. Bei unseren Experimenten hat das Graphem-Clustering auch zu keinen Verbesserungen in der Erkennungsrate geführt. Dabei hat das Graphem-Clustering mit den gleichen Problemen wie das Codebook-Clustering zu kämpfen. Auch hier ist die Frage, wie gut die Distanzberechnung auf den schlechten Daten funktioniert und wie gut die Berechnung der neuen Parameter funktioniert, oder ob man diese verbessern könnte.

Generell macht aber die Idee des Graphem-Clusterings in deutlich weniger Fällen Sinn, als dies beim Codebook-Clustering der Fall ist. Es kann einen guten Einfluss haben, wenn es zwei Grapheme gibt, die sehr ähnlich sind. Dann kann es Sinn machen diese zu clustern und so ein besser trainiertes Graphem zu erhalten. Ein Beispiel dafür könnten zum Beispiel die Deutschen Grapheme F und V sein, welche in vielen Fällen gleich ausgesprochen werden. Dies ist aber eher bei wenigen Graphemen der Fall, wodurch die Reduzierung der zu trainierenden Parameter bessere Aussichten auf Erfolg mit Hilfe des Codebook-Clusterings haben.

7.3. Ausblick

Es gibt verschiedene Möglichkeiten diese Arbeit weiterzuführen. Zum Einen gibt es die interessanteste Möglichkeit Multilinguale und Crosslinguale Systeme einzusetzen um für Sprachen mit einer geringen Verbreitung und wenigen Trainingsdaten gute Systeme zu erstellen. Dabei würde es sich zum Beispiel anbieten für Haitianisches Kreolisch, Französisch als Quellsprache zu verwenden.

Eine andere Möglichkeit besteht darin, die Methoden auf anderen Sprachen und Trainingsdaten zu testen, um genauere Aussagen über den allgemeinen Nutzen der Methoden treffen zu können. Vor allem wären die Ergebnisse mit weniger verrauschten Daten interessant. Weiterhin gibt es beim Codebook-Clustering, wie schon in der Diskussion erwähnt, Möglichkeiten die Systeme zu verbessern, indem die Parameterberechung oder die Auswahl der Codebooks modifiziert wird. Dabei sollten die Ergebnisse aber auf weniger verrauschten Daten stattfinden, damit die Distanzberechnung sinnvoller funktioniert. Dabei könnten auch verschiedene Techniken untersucht werden um die Daten besser zu filtern und damit weniger Probleme mit den verrauschten Daten zu haben.

Ein weiterer Ansatz der überprüft werden könnte ist, wie gut die Sprachen mit phonembasierten Systemen funktionieren um Aussagen darüber treffen zu können, ob die Sprachen für graphembasierte Spracherkennungssysteme geeignet sind. Schlussendlich gibt es noch die Möglichkeit Neuronale Netze für die Spracherkennungssysteme einzusetzen. Dabei ist vor allem auch interessant zu wissen, wie gut Neuronale Netze mit dem Graphemclustering funktionieren.

Literaturverzeichnis

[BIA00] B. Horvat B. Imperl, Z. Kacic and A.Zgank: *Agglomerative vs. tree-based clustering for definition of multilingual set of triphones.* In *IEEE International Conference of Acoustics, Speech and Signal Processing*, pages 1273–1276, 2000.

[BN08] Maximilian Bisani und Hermann Ney: *Joint-sequence models for grapheme-to-phoneme conversion.* Speech Communication, 50(5):434–451, 2008.

[Chu04] K. Church: *Speech and Language Processing: Can We Use the Past to Predict the Future?* In: *Text, Speech and Dialogue*, Band 3206 der Reihe *Lecture Notes in Computer Science*, Seiten 3–13. Springer Berlin Heidelberg, 2004.

[DIG12] Herve Bourlard David Imseng and Philip N. Garner: *Using kl-divergence and multilingual information to improve asr for underresourced languages.* In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012*, pages 4869–4872, 2012.

[DPA11] L. Burget D. Povey and M. Agarwal: *Computer Speech and Language*, volume 25, chapter The subspace Gaussian mixture model A structured model for speech recognition. Elsevier Ltd., 2011.

[dS06] Van der Spuy: *Zulu.* Encyclopedia of Language and Linguistics second edition, Seiten 759–761, 2006.

[gra] *Definition grapheme.* `http://www.oxforddictionaries.com/definition/english/grapheme`, Abrufdatum: 08.05.2014.

[HaK] `http://en.wikipedia.org/wiki/Haitian_Creole`, Abrufdatum: 09.05.2014.

[Hal12] C. Halpert: *Overlap-driven consequences of Nasal place assimilation.* Consonant Clusters and Structural Complexity, 26:345, 2012.

[IAR] *IARPA-BAA-11-02.* `http://www.iarpa.gov/index.php/research-programs/babel/baa`, Abrufdatum: 10.05.2014.

[Jan12] M. Janda: *Grapheme based speech recognition.* In *Proceedings of the 18th Conference STUDENT EEICT 2012*, pages 441–445, 2012.

[JRT] `http://isl.anthropomatik.kit.edu/english/1406.php`, Abrufdatum: 10.05.2014.

[JS06] S. Narayanan J. Silva: *Upper bound kullback-leibler divergence for hidden markov models with application as discrimination measure for speech recognition.* In *IEEE International Symposium on Infomation Theory, 2006*, pages 2299–2303, 2006.

[KL51] S. Kullback and R. A. Leibler: *On information and sufficiency.* In *The Annals of Mathematical Statistics*, volume 22, pages 79–86. Institute of Mathematical Statistics, 1951.

[Lev66] V. I. Levenshtein: *Binary codes capable of correction deletions, insertions and reversals.* In *Soviet Physics Doklady*, pages 707–710, 1966.

[LLR11] A. Ghoshal L. Lu and S. Renals: *Regularized subspace gaussian mixture models for cross-lingual speech recognition.* In *IEEE Workshop on Automatic Speech Recognition and Understanding (ARSU),2011*, pages 365–370, 2011.

[MA14] M. Mason und J. Allen: *Standardized Spelling as a Localization Issue.* http://www.multilingual.com/articleDetail.php?id=589, 2014.

[MMDB11] G. Aradilla M. Magimai-Doss, R. Rasipuram and H. Bourlard: *Grapheme-based automatic speech recognition using kl-hmm.* In *INTERSPEECH 2011*, 2011.

[PCS06] S. Hewavitharana P. Charoenpornsawat and T. Schulz: *Thai grapheme-based speech recognition.* In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 17–20, 2006.

[pho] *Definition phonem.* `http://www.oxforddictionaries.com/definition/english/phoneme`, Abrufdatum: 08.05.2014.

[Rab89] L. Rabiner: *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE, 77(2):257–286, 1989.

[RBD13] Ramya Rasipuram, Peter Bell und Mathew Magimai Doss: *Grapheme and multilingual posterior features for under-resourced speech recognition: A study on Scottish Gaelic.* In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, Seiten 7334–7338. IEEE, 2013.

[Rey08] D. A. Reynolds: *Gaussian mixture models.* Encyclopedia of Biometric Recognition, 2008.

[RF97] I. Rogina and M. Finke: *Wide context acoustic modeling in read vs. spontaneous speech.* In *IEEE International Conference on Acoustics, Speech, and Signal Processing 1997*, pages 1743–1746, 1997.

[RSS99] B. Rag R. Singh and R. Stern: *Automatic clustering and gernation of contextual questions for tied states in hidden markov models.* In *IEEE International Conference on Acoustics, speech, and signal Processing, 1999*, pages 117–120, 1999.

[SOS12] Tim Schlippe, Sebastian Ochs und Tanja Schultz: *Grapheme-to-phoneme model generation for Indo-European languages.* In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, Seiten 4801–4804. IEEE, 2012.

[SS04] S. Stüker and T. Schulz: *A grapheme based speech recognition system for russian.* In *SPECOM 2004: 9th Conference, Speech and Computer*, pages 297–303, 2004.

[ST95] E.G. Schukat-Talamazzini: *Automatische Spracherkennung - Grundlagen, statistische Modelle und effiziente Algorithmen.* Vieweg, 1995.

[ws1] `http://de.statista.com/statistik/daten/studie/150407/umfrage/die-zehn-meistgesprochenen-sprachen-weltweit/`, Abfragedatum: 09.05.2014.

[ws2] `http://www2.ignatius.edu/faculty/turner/languages.htm`, Abfragedatum: 09.05.2014.

[YMW13] F. Metze Y. Miao and A. Waibel: *Subspace mixture model for low-resource speech recognition in cross-lingual settings.* In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2013*, 2013.

[Zul] `http://de.wikipedia.org/wiki/IsiZulu`, Abrufdatum: 09.05.2014.

Anhang

A. Trainingsdaten - Acknowledgement

Supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

This effort uses the IARPA Babel Program Haitian_Creole language collection release babel201b-v0.2b and the IARPA Babel Program Zulu language collection release babel206b-v0.1d. In our experiments, we used the FullLP (FLP) and LimitedLP (LLP) datasets as indicated.

B. Phoneme Sets

Die Liste zeigt die Phoneme Sets, die für die einzelnen Systeme verwendet wurden. Dabei werden normalerweise die Phoneme in verschiedene Klassen eingeteilt wie Vokale, Konsonanten, Nasale, ... Dies ist bei graphembasierten Spracherkennungssystemen anders, da wir von Graphemen genau diese Eigenschaften nicht wissen. Die Unterscheidung in Vokale und Konsonanten könnte man noch einführen, hat man hier aber nicht gemacht.

B.1. Haitianisches Kreolisch

Zuerst wird das komplette Phoneme Set mit den einzelnen Untergliederungen vom LLP Baseline System und damit auch der semikontinuierlichen Systeme aufgeführt. Danach wird für die anderen Systeme nur noch die Klasse VOICED notiert. Daraus kann man leicht erschließen, wie die anderen Klassen auszusehen haben.

Klasse	**Phoneme**
PHONES	PAD +BREATH+ +CLICK+ +COUGH+ +FOREIGN+ +GARBAGE+ +HESTITATION+ +LAUGH+ +NOISE+ +RING+ +SMACK+ A B C D E F G H I J K L M N O P R S SIL T U V W X Y Z À È Ò
A	A
B	B
C	C
D	D
E	E
F	F
G	G
H	H
HUMAN-NOISE	+BREATH+ +COUGH+ +FOREIGN+ +GARBAGE+ +HESTITATION+ +LAUGH+ +SMACK+
I	I
J	J
K	K
L	L
M	M
N	N
NOISES	+BREATH+ +CLICK+ +COUGH+ +FOREIGN+ +GARBAGE+ +HESTITATION+ +LAUGH+ +NOISE+ +RING+ +SMACK+
NON-HUMAN	+NOISE+ +RING+
O	O
P	P
R	R
S	S
SILENCES	SIL
T	T
U	U
V	V
VOICED	A B C D E F G H I J K L M N O P R S T U V W X Y Z À È Ò
W	W
X	X
Y	Y
Z	Z
À	À
È	È
Ò	Ò

Änderungen bei anderen Systemen

System	VOICED - Klasse des Systems
FLP Baseline	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z À È Ò
Merge5	A AN B C D E EN F G H I J K L M N O ON OU P PA Q R S T U V W X Y Z À È Ò
Merge10	A AN AP B C D E EN F G H I J K KO L LA M N NA NM O ON OU P PA Q R S T U V W X Y Z À È Ò
Merge15	A AL AN AP B C D E EN EY F G H I J K KO L LA M N NA NM O ON OU P PA Q R S SA T U V W X Y YO Z À È Ò
MergeKons	A B C D E F G H I J K L M MP N NM NP NS O P PR Q R S T U V W X Y Z À È Ò
PhoneCluster	A B C D EI F G HR J K L M NU O P S T V W X Y Z À È Ò

Anmerkung: In den Trainingsdaten des FLP Baseline Systems kommt noch das Grapheme Q vor, welches in den Trainingsdaten des LLP Baseline Systems nicht vorkommt. Deswegen stimmen die Phoneme Sets der beiden Systeme nicht überein.

B.2. Zulu

Zuerst wird das komplette Phoneme Set mit den einzelnen Untergliederungen vom LLP Baseline System, FLP Bseline System und damit auch der semikontinuierlichen Systeme aufgeführt. Danach wird für die anderen Systeme nur noch die Klasse VOICED notiert. Daraus kann man leicht erschließen, wie die anderen Klassen auszusehen haben.

Klasse	**Phoneme**
PHONES	PAD +BREATH+ +CLICK+ +COUGH+ +FOREIGN+ +GARBAGE+ +HESTITATION+ +LAUGH+ +NOISE+ +RING+ +SMACK+ A B C D E F G H I J K L M N O P Q R S SIL T U V W X Y Z
A	A
B	B
C	C
D	D
E	E
F	F
G	G
H	H
HUMAN-NOISE	+BREATH+ +COUGH+ +FOREIGN+ +GARBAGE+ +HESTITATION+ +LAUGH+ +SMACK+
I	I
J	J
K	K
L	L
M	M
N	N
NOISES	+BREATH+ +CLICK+ +COUGH+ +FOREIGN+ +GARBAGE+ +HESTITATION+ +LAUGH+ +NOISE+ +RING+ +SMACK+
NON-HUMAN	+NOISE+ +RING+
O	O
P	P
Q	Q
R	R
S	S
SILENCE	SIL
T	T
U	U
V	V
VOICED	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
W	W
X	X
Y	Y
Z	Z

Änderungen bei anderen Systemen

System	VOICED - Klasse des Systems
Merge5	A AN B C D E F G H I IN J K L LA M N NA NG O P Q R S T U V W X Y Z
Merge10	A AN AY B C D E F G GI H HA I IN J K KU L LA M N NA NG O P Q R S T TH U V W X Y Z
Merge15	A AN AY B C D E EL F G GI H HA HE HI I IN J K KU L LA LE M N NA NG O P Q R S T TH U V W X Y Z
MergeKons	A B C D E F G H I J K KH L M N NG O P PH Q R S SH T TH U V W X Y Z
PhoneCluster	A B C D E F GI H J KP LN M O Q R S T U V W X Y Z

C. Codebooks

Im folgenden werden die Codebooks, die in den semikontinuierlichen Systemen zusammen geclustert wurden, werden im Folgenden für beide Sprachen aufgelistet.

C.1. Haitianisches Kreolisch

Cluster5

Neues Codebook	Alte Codebooks
E-e	E-e È-e
E-m	E-m È-m
N-b	N-b N-m
O-e	O-e U-b
T-b	T-b K-b

Cluster10

Neues Codebook	Alte Codebooks
A-e	A-e A-m
D-e	D-e I-b
E-e	E-e È-e
E-m	E-m È-m
I-e	I-e N-e
K-e	K-e T-e
M-m	M-m M-e
N-b	N-b N-m
O-e	O-e U-b
T-b	T-b K-b

Cluster20

Neues Codebook	Alte Codebooks
A-e	A-e A-m
D-e	D-e I-b
E-e	E-e È-e I-e
E-m	E-m È-m
I-e	I-e N-e
K-e	K-e T-e T-m
N-b	N-b N-m
T-b	T-b K-b
O-m	O-m W-e
U-m	U-m O-e U-b
L-m	L-m L-b M-m M-e
D-b	D-b G-b
S-b	S-b F-b
È-b	È-b R-e
G-e	G-e E-b

Cluster10.2

Neues Codebook	Alte Codebooks
T-e	T-e C-e
N-b	N-b À-b X-b Q-b
A-m	A-m X-m Q-m À-m
E-e	E-e X-e Q-e À-e

C.2. Zulu

Cluster5

Neues Codebook	Alte Codebooks
A-e	A-e O-e
E-m	E-m I-m
L-m	L-m L-b
L-e	L-e N-e
T-b	T-b P-b

Cluster10

Neues Codebook	Alte Codebooks
A-m	A-m W-e
A-e	A-e O-e
E-m	E-m I-m
G-e	G-e I-b
I-e	I-e U-e
L-m	L-m L-b
L-e	L-e N-e
T-b	T-b P-b
W-m	O-m W-m
Z-b	Z-b N-m

Cluster20

Neues Codebook	Alte Codebooks
A-m	A-m W-e A-b
A-e	A-e O-e
B-b	B-b V-b
B-e	B-e G-m
E-m	E-m I-m
G-e	G-e I-b
I-e	I-e U-e
K-b	K-b E-e
L-m	L-m L-b
L-e	L-e N-e
R-b	R-b H-e
T-b	T-b P-b X-b
T-m	T-m F-b
T-e	T-e P-e
W-m	O-m W-m
Y-e	Y-e Y-m
Z-b	Z-b N-m
Z-m	Z-m Z-e

Cluster10.2

Neues Codebook	Alte Codebooks
H-m	H-m C-m Q-m
H-e	H-e C-e X-e Q-e
L-m	L-m V-m
L-e	L-e V-e
P-b	Q-b P-b X-b
S-m	S-m X-m