Automatische Generierung Akustischer Spracheinheiten

Studienarbeit

von

Keni Bernardin

Institut für Logik, Komplexität und Deduktionssysteme
Prof. Dr. Alexander Waibel
Fakultät für Informatik
Universität Karlsruhe

Betreuer:
Ivica Rogina
mit freundlicher Unterstützung von
Thomas Schaaf

23. März 2001

Zusammenfassung:

Bei der Erkennung kontinuierlicher spontaner Sprache werden heutzutage Systeme verwendet, deren akustisches Modell auf kontextabhängigen Phonemen basiert. Damit ein Wort vom System erkannt werden kann, muß seine phonetische Aussprache in einem Lexikon festgelegt sein. Diese Festlegung wird aber noch vom Menschen durchgeführt, anhand von linguistischem Vorwissen, meist nicht von tatsächlich in gesprochener Sprache enthaltener, akustischer Information. Daher ist die Verwendung von Phonemen oder Phonemteilen als kleinste erkennbare Spracheinheiten nicht unbedingt optimal. Auch kann somit für nichtsprachliche Geräusche nur schwer eine passende Modellierung gefunden werden.

In der vorliegenden Arbeit wird ein Verfahren vorgestellt, das geeignete akustische Grundeinheiten selbstständig, ohne linguistisches Wissen, aus Sprachdaten extrahiert. Diese Einheiten, die auch nonverbaler Natur sein können, werden benutzt, um automatisch ein Lexikon aufzubauen, in dem gleichermaßen für Wörter, wie auch für sonstige Geräusche, eine Aussprache festgelegt ist. Die dabei erzielten Erkennungsraten bleiben zwar noch deutlich unter denen eines konventionellen Systems, zeigen aber, daß solch ein Verfahren prinzipiell sehr wohl anwendbar ist.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Automatische Spracherkenner mußten im Laufe der Zeit mit einem immer größer werdenden Vokabular zurecht kommen. Vor einigen Jahren schon wurde erkannt, daß es deshalb sinnvoll ist, nicht ganze Wörter, sondern kleinere Einheiten wie z.B. Phoneme innerhalb eines Wortes zu erkennen, und diese dann zu ganzen Wörtern zusammenzufügen. Ein wichtiger Teil heutiger Spracherkenner ist das Aussprachelexikon, das für jedes zu erkennende Wort angibt, aus welchen Grundeinheiten es besteht. Da traditionell in der Linguistik Phoneme benutzt wurden, um die Aussprache eines Wortes wiederzugeben, werden auch Phoneme oder Teile von Phonemen als Grundeinheiten in heutigen Aussprachelexika benutzt. Für die meisten weitverbreiteten Sprachen sind phonembasierte Lexika schon in der Literatur erhältlich. Sie können dann einfach für die Spracherkennung übernommen werden. Anders ist es bei Sprachen, für die es noch keine Lexika gibt. Letztere müssen dann entweder von Experten manuell erstellt oder automatisch erzeugt werden, z.B. anhand von Regeln, die angeben, wie man von einer Buchstaben- zu einer Phonemfolge kommt. Für das Französische könnte eine solche Regel so lauten: Die Buchstabenfolge "an", wird mit den Phonemen /A/ und /N/ wiedergegeben, falls der nächste Buchstabe ein Vokal ist. Sie wird mit einem einzigen Phonem /AN/ wiedergegeben, falls der nächste Buchstabe ein Konsonant ist, oder falls das Wortende erreicht wurde.

Die Erstellung solcher Regeln ist eine aufwendige Arbeit, speziell bei Sprachen, bei denen es viele Ausnahmen gibt oder die Aussprache nicht auf Regeln basiert. Und es wäre wieder die Arbeit eines Experten, das Lexikon auf Fehler zu überprüfen.
Darüber hinaus wird eine Erkennung rein auf der Basis von Phonemen keine guten Ergebnisse liefern, da das gleiche Phonem, je nachdem in welchem Kontext es sich befindet, verschieden klingt. Das /A/ zwischen /T/ und /L/ klingt nicht so wie das /A/ zwischen /N/ und /S/. Deshalb arbeiten moderne Spracherkenner intern mit kontextabhängigen Phonemen als Grundeinheiten. [Lee90]

Selbst dann entstehen Probleme bei sprecherunabhängigen Systemen, da selbst bei sehr unterschiedlicher Aussprache oder starkem Dialekt Wortteile immer noch mit demselben (kontextabhängigen) Phonem modelliert werden. Es stellt sich also die Frage, inwieweit Phoneme überhaupt als akustische Einheiten geeignet sind.

Diese Studienarbeit verfolgt zwei Ziele, nämlich herauszufinden, ob

- die akustischen Einheiten, die bei der Erkennung benutzt werden sollen, automatisch aus den Sprachdaten ermittelt werden können,
- das Aussprachelexikon, aufbauend auf diesen neuen Einheiten, automatisch erzeugt werden kann, so daß alle Wörter mit ihren wichtigsten in den Sprachdaten vorhandenen Aussprechmöglichkeiten enthalten sind.

Wenn die akustischen Einheiten (im Folgenden auch Atome) ohne linguistische Berücksichtigungen nur aus dem Datenmaterial extrahiert werden, erwarten wir, je nach Häufigkeit ihres Auftretens auch Atome zu erhalten, die bei phonembasierten Systemen im Allgemeinen nicht benutzt werden. Zum Beispiel das "OH!", das bei starker Verwunderung ausgestoßen wird, und das sich akustisch deutlich von dem normalen "O" unterscheidet, wie es innerhalb eines Wortes, ohne spezielle Intonation, ausgesprochen wird. Oder der Nasallaut am Ende von "Restaurant", den es im offiziellen deutschen Phonemsatz gar nicht gibt.

Die datengetriebene Generierung des Lexikons bringt, neben der offensichtlichen Bequemlichkeit des Automatismus, auch andere Vorteile: In spontaner Sprache werden Wörter oft sehr unterschiedlich ausgesprochen und betont. Ein Wort wie "morgen" wird je nach Sprecher oder Situation "morgen", "mo'gen", "morchen", "mor'n", "moin" ausgesprochen. Die Eintragung aller denkbaren Varianten ist wohl nicht sinnvoll, und, bei manueller Erzeugung des Lexikons, auch sehr anstrengend. Andererseits kann das völlige Weglassen von Varianten die Erkennungsleistung stark beeinträchtigen. Ein Verfahren, das sich auf das Trainingsmaterial stützt, könnte alle vorhandenen Aussprachen eines Wortes berücksichtigen und die häufigsten als Varianten abspeichern.

Im folgenden Kapitel werden noch einmal die kontextabhängigen Phoneme vorgestellt, die z.Z. als akustische Spracheinheiten die größte Verbreitung genießen. Außerdem wird ein bisher nicht für kontinuierliche Sprache verwendeter Ansatz, der Successive State Splitting Algorithmus (SSS), vorgestellt, der versucht, rein datengetrieben neue akustische Untereinheiten von Phonemen zu ermitteln. In Kapitel 3 wird der neue Algorithmus zur automatischen Generierung akustischer Atome vorgestellt, der auf dem Aufspalten (Splitten) und anschließenden Gruppieren (Clustern) von akustischen Modellen beruht. Kapitel 4 stellt die so erzielten Ergebnisse dar und Kapitel 5 gibt einen kurzen Ausblick.

Kapitel 2

Akustische Grundeinheiten

Bei der Suche nach geeigneten akustischen Atomen zur Beschreibung der Aussprache von Wörtern für Spracherkennungssysteme, wurde früh entdeckt, daß eine Erkennung rein auf Basis von Phonemen zahlreiche Probleme mit sich bringt, die zu schlechten Erkennungsraten führen. Besonders bei Systemen für kontinuierliche spontane Sprache mit großem Vokabular. Bei der Aussprache der Phoneme treten so viele Varianten auf, bedingt durch ihre Stellung innerhalb eines Worts, daß eine feinere Unterteilung notwendig wird, um alle vorkommenden Nuancen besser zu erfassen.

2.1 Kontextabhängige Phoneme

Ein kontextabhängiges Phonem ist ein Phonem, das um zusätzliche Information über seine unmittelbare Umgebung erweitert wurde. Diese Umgebung (der Kontext) kann sich nur auf das vorangehende oder nachfolgende Phonem beziehen, sie kann aber auch die nächsten 2 oder 3 Phoneme umfassen, oder gar wortübergreifend sein.

Die Benutzung dieser größeren Anzahl, feinerer akustischer Einheiten bedeutet, daß das System über detailliertere Modelle verfügt, um einen Teil eines Wortes zu erkennen. Statt nur über ein einfaches /I/ verfügt es auch über ein /I/ zwischen /D/ und /R/, ein /I/ zwischen /B/ und /A/, eines zwischen /A/ /F/ und /N/ /I/, usw. Je mehr Modelle das System besitzt, desto genauer ist seine Repräsentation der realen Welt. Aber je größer die Anzahl der Modelle ist, desto weniger Trainingsdaten haben wir auch, um deren Parameter zu schätzen. Deshalb wird in einem kontextabhängigen System die Anzahl der Modelle oft beschränkt. Mit einem geeigneten Distanzmaß, wie z.B. der gewichteten Entropie-Distanz, werden die Modelle miteinander verglichen und die ähnlichsten vereint. Die am häufigsten in den Trainingsdaten vorkommenden kontextabhängigen Phoneme erhalten so ein eigenes Modell.

Kontextabhängige Systeme erzielen schon eine viel bessere Erkennung als kontextunabhängige, jedoch berücksichtigen sie immer noch nicht eine Reihe von Phänomenen, die in der natürlichen Sprache vorkommen. Ein Wort kann von verschiedenen Sprechern ganz unterschiedlich ausgesprochen werden. Diese Variationen, z.B. das Anheben der Tonlage am Ende des Wortes "Sicher" bei einer Frage im Gegensatz zu ihrer Ab-

senkung bei einer Bestätigung, können oft nicht mit Phonemen beschrieben werden. Darüber hinaus kommen in spontaner Sprache Laute vor wie z.B. Lachen, Schnalzen, Husten, oder andere Artefakte, wie das Verschlucken von Wortteilen, usw. Für diese speziellen Laute müssen auch passende Modelle gefunden werden, wenn sie in einem spontan gesprochenen Satz auch als solche erkannt werden sollen. Das Problem dabei ist: Wie sieht ein Lexikoneintrag für "Schnalzen" oder "Grunzen" aus? Da diese Laute nicht mit Phonemen beschreibbar sind, können sie nicht als Folge von solchen dargestellt werden. Heißt das nun, daß man nur jeweils ein Modell für ihre Erkennung verwenden soll? Gewiß sind solche Laute auch in kleinere Einheiten unterteilbar, solange dafür auch geeignete gefunden werden können. Sie müßten gegebenenfalls automatisch aus den Trainingsdaten ermittelt werden. [Schultz94]

2.2 Successive State Splitting

Der Successive State Splitting Algorithmus (SSS) versucht, gerade dies zu erreichen: eine automatische Aufteilung von akustischen Modellen in kleinere Einheiten. Der Algorithmus, wie er in [Takami-Sagay] vorgestellt wird, basiert auf Hidden Markov Modellen (HMMs) und arbeitet nach folgendem Prinzip:

Ein einzelner Wortteil (hier ein Phonem), wird anfangs durch ein HMM mit nur einem Zustand beschrieben. Dieser Zustand soll nun weiter aufgeteilt werden, um zeitliche Variationen, sowie verschiedene mögliche Aussprachen des Buchstaben zu berücksichtigen. Um unter allen Zuständen aller Buchstaben herauszufinden, welche aufgeteilt werden sollen, werden sie erst mit den verfügbaren Daten trainiert. Dann wird die Varianz im Modell eines Zustandes betrachtet und die n Zustände mit den größten Varianzen werden gesplittet. Um zu bestimmen, ob die Aufteilung (der Split) räumlicher oder temporaler Natur sein soll, wird der Zustand nochmals trainiert: Einmal, indem er selbst durch zwei Verteilungen modelliert wird, ein andermal, indem er durch zwei parallel geschaltete Zustände mit jeweils einer Verteilung ersetzt wird. (Abb. 2.1)

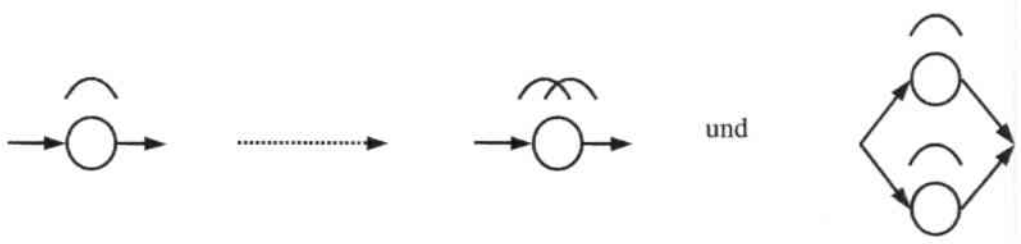

Abbildung 2.1: Trainieren mit gesplitteten Modellen beim SSS (Die Kreisbögen über den Zuständen repräsentieren Gaußverteilungen.)

Mit einem Entropie-Maß wird die Distanz zwischen den so trainierten Verteilungen

gemessen, um herauszufinden, welcher der Splits den größten Informationsgewinn bringt. Wenn sich das zugehörige Sprachsignal stark über die Zeit ändert, der temporale Split also vorzuziehen wäre, wird der alte Zustand durch zwei neue, seriell angeordnete ersetzt, denen jeweils eine der trainierten Verteilungen zugeteilt wird. Ist die Änderung räumlicher Natur, d.h. wird sie durch unterschiedliche Aussprachen bedingt, so wird ein räumlicher Split vorgenommen, der alte Zustand wird durch die zwei parallelen ersetzt. (Abb. 2.2, 2.3)

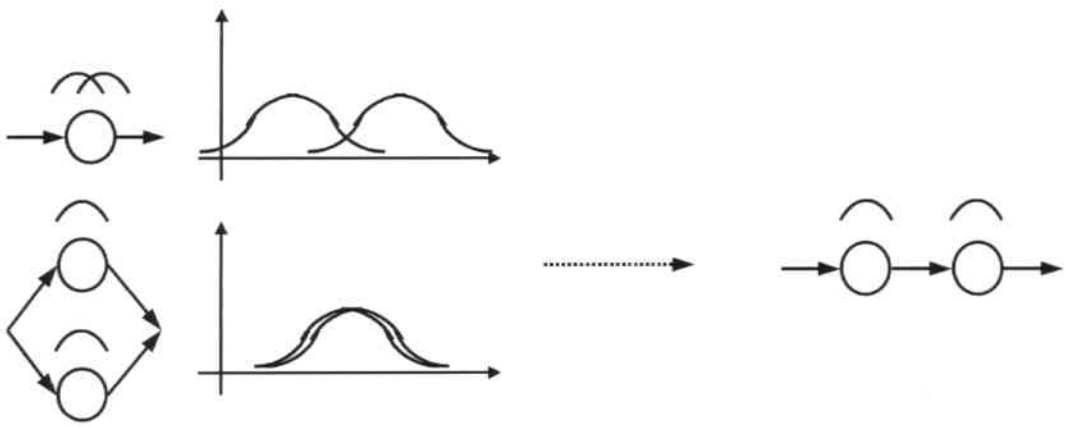

Abbildung 2.2: Zeitliches Aufsplitten eines Zustandes. Die Gaußverteilungen des zeitlichen Splits unterscheiden sich stärker als die des räumlichen.

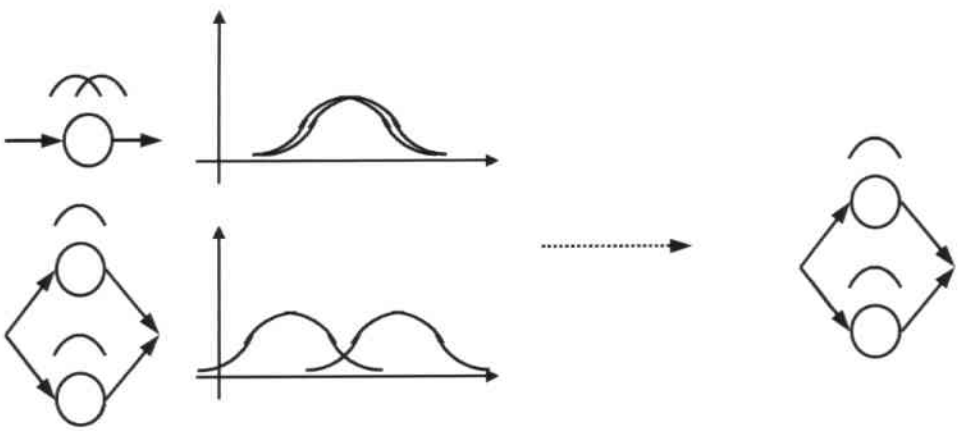

Abbildung 2.3: Räumliches Aufsplitten eines Zustandes. Ein zeitlicher Split bringt hier weniger Informationsgewinn.

In [Singer-Ost] wird eine andere Variante dieses Algorithmus vorgestellt, der ML-SSS

(Maximum Likelihood SSS), diesmal für eine Unterteilung von Phonemen. Im Gegensatz zum normalen SSS wird hier nicht zuerst der zu splittende Zustand gewählt und dann der für ihn beste Split gesucht. Vielmehr werden hier unter allen möglichen Splits die n Splits ausgesucht, die die Wahrscheinlichkeit der Beobachtung maximieren, seien sie temporaler oder räumlicher Natur. Nur diese Splits werden dann auch durchgeführt.

In beiden Varianten wird das ursprüngliche HMM durch wiederholtes Aufteilen seiner Zustände ständig erweitert. Dies resultiert in einem sogenannten HM-Netz. Dabei muß ein klares Abbruchkriterium definiert werden, da sich diese Netzte sonst unbeschränkt weiter verfeinern. (Abb. 2.4)

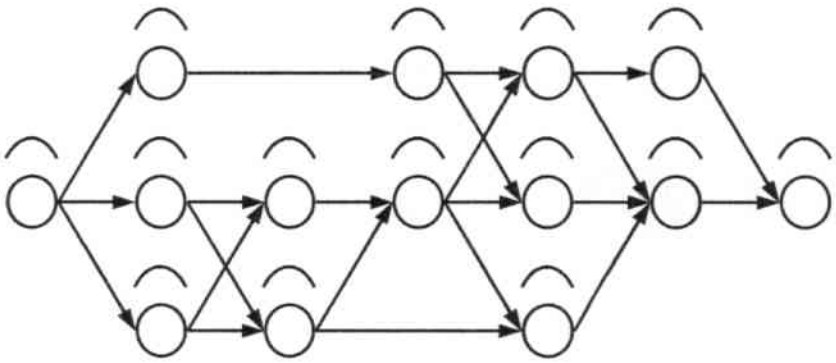

Abbildung 2.4: HM-Netz

Im Endeffekt können die Modelle der Zustände eines solchen HM-Netzes als akustische Grundeinheiten verstanden werden. Die Aufteilung erfolgt vollautomatisch und die erhaltenen akustischen Atome haben nichts mehr mit Phonemen gemeinsam. Dieser Algorithmus hat bei der Einzelerkennung kleiner Mengen von Phonemen gute Ergebnisse geliefert [Takami-Sagay] [Singer-Ost]. Es wird nun untersucht, ob er nach einigen Modifikationen geeignet ist, auch für große Mengen von Wörtern bei der Erkennung kontinuierlicher natürlicher Sprache, mit all ihren spontanen Effekten, eingesetzt zu werden.

Kapitel 3

Automatische Generierung akustischer Einheiten

Bei einen Erkenner auf HMM-Basis wird ein zu erkennendes Wort durch eine Menge von Zuständen beschrieben. Die Zustände selbst repräsentieren die akustischen Einheiten, aus denen das Wort aufgebaut ist. Eine Menge von Zustandsübergangswahrscheinlichkeiten gibt an, wie wahrscheinlich es ist, von einem Zustand des Wortes in einen anderen zu springen. (Abb. 3.1)

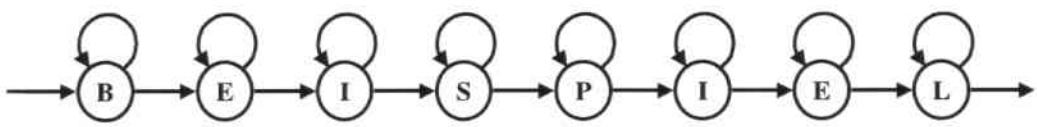

Abbildung 3.1: Beispiel eines HMMs. Die Buchstaben in den Zuständen repräsentieren hier nicht Phoneme, sondern sind nur symbolischer Natur. (In allen folgenden Bildern werden reflexive Verbindungen zur besseren Anschaulichkeit ausgelassen.)

Es wird eine von der Vorverarbeitung des Sprachsignals gelieferte Merkmalsfolge X untersucht und ermittelt, um welche Wortfolge es sich wahrscheinlich handelt. Nach Bayes ist die Wahrscheinlichkeit eines Wortes oder Wortfolge W, bei beobachteter Merkmalsfolge X (stark vereinfacht):

$$P(W \mid X) = \frac{P(X \mid W) \cdot P(W)}{P(X)}$$

Die Berechnung von $P(X|W)$, also der Wahrscheinlichkeit, X zu beobachten, angenommen es handelt sich um das Wort W, hängt vom akustischen Modell ab. Sie wird z.B. mit einem Forward- oder einem Viterbi-Algorithmus, basierend auf dem HMM für dieses Wort, berechnet. Die A-priori-Wahrscheinlichkeit $P(W)$, das Wort W überhaupt zu beobachten, wird hingegen vom Sprachmodell bestimmt. Im folgenden

beschäftigen wir uns nur mit dem akustischen Modell.

Wie beim SSS-Algorithmus sollen die Wort-HMMs zu HM-Netzen (im folgenden auch Zustandsgraphen) erweitern werden. Dafür werden die einzelnen Zustände des Wortes iterativ räumlich oder zeitlich gesplittet.

Dies wirft einige Fragen auf:

- Wie viele Splits sollen pro Iteration zulassen werden?
- Welche Größe darf der Zustandsgraph für ein Wort maximal erreichen?
- Wie wird mit einem sehr großen Vokabular umgegangen? Die bisherigen Realisierungen des SSS-Algorithmus hatten es nur mit kleinen Mengen von Buchstaben oder Phonemen zu tun. Wenn wir bei großen Wortmengen jedem Zustandsgraphen erlauben, eigene Atome zu besitzen, erhalten wir eine ungeheure Menge zum größten Teil sehr ähnlicher Atome. Zustandsgraphen müssen also imstande sein, sie gemeinsam zu nutzen. Ähnliche Atome sollten verschmolzen werden.

Der hier vorgestellte Ansatz versucht, all diese Probleme gemeinsam zu lösen. Das Finden und Trainieren geeigneter akustischer Grundeinheiten, die Wahl der besten Splits, das Clustern von Atomen und der Aufbau von Wort-Zustandsgraphen sollen in einem Algorithmus realisiert werden.

Im Initialzustand sollen alle Wörter durch ein einziges, für alle gleiches Atom beschrieben werden. Dieses sollte so initialisiert sein, daß es im Merkmalsraum ungefähr im Mittelpunkt aller vorkommenden Merkmale steht. Es "modelliert" so gut es geht alle Worte gleichzeitig. Der anschließende Algorithmus ist in vier Phasen eingeteilt:

- Splitten der vorhandenen Atome.
- Trainieren der so entstandenen neuen Atome.
- Clustern dieser neuen Atome.
- Auswählen der geeigneten Splits und Aufbauen von neuen Zustandsgraphen.

3.1 Splitten

Jeder Zustand von jedem Wort wird zunächst räumlich und zeitlich aufgesplittet. Ein räumlicher Split bedeutet, daß an seiner Stelle im Zustandsgraph zwei neue Zustände mit eigenen Modellen U und D parallel eingebaut werden. Bei einem zeitlichen Split hingegen, wird etwas anders vorgegangen, als beim in Kapitel 2 vorgestellten SSS-Algorithmus. Wie schon in [Singer-Ost] vorgeschlagen wird dort nicht ein Zustand mit zwei akustischen Modellen eingefügt, sondern zwei Zustände mit einzelnen Modellen L und R in Serie geschaltet. Es werden also in jedem Wort pro Zustand vier neue Atome erzeugt. (Abb. 3.2)

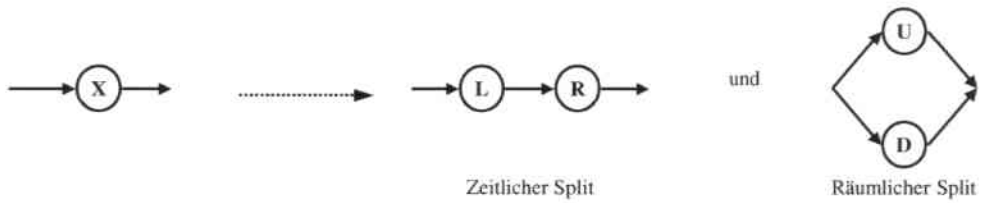

Abbildung 3.2: Aufsplitten eines Zustands vor dem Trainieren

Die neuen Modelle werden anhand der Information aus dem ursprünglichen Atom initialisiert. Bei einem räumlichen Split wird zunächst die Richtung der größten Varianz des alten Modells im Merkmalsraum ermittelt. Entlang dieser Richtung werden nun die neuen Modelle um einen kleinen Wert $\pm e$ gegenüber dem alten verschoben. (Abb. 3.3)

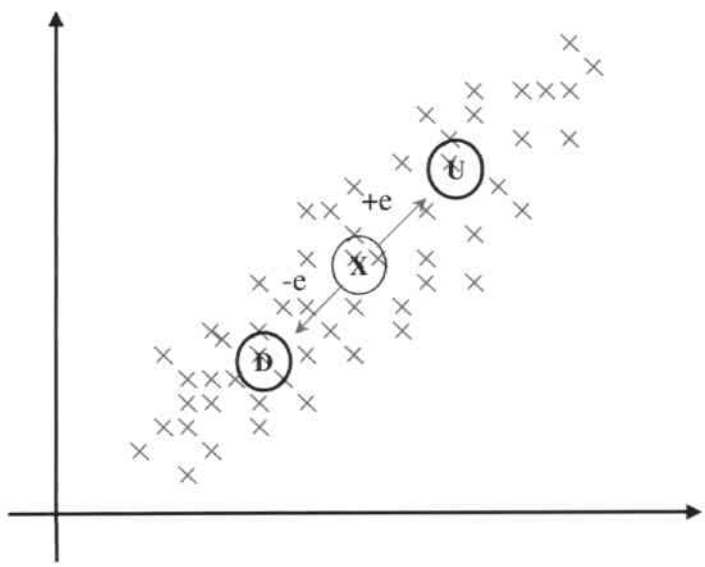

Abbildung 3.3: Räumlicher Split: Initialisierung entlang der Richtung der größten Varianz. (Hier wurde der Merkmalsraum zur besseren Anschaulichkeit auf zwei Dimensionen reduziert.)

Bei einem zeitlichen Split hingegen, werden sie zunächst mit den Werten des alten Atoms initialisiert. Es wird darauf verzichtet, sie im Merkmalsraum zu verschieben. Dies liegt daran, daß nicht eindeutig zu bestimmen ist, ob die Varianz des ursprünglichen Modells auf eine räumliche Variation, also auf unterschiedliche Aussprechmöglichkeiten, oder auf eine zeitliche, also auf das sukzessive Aussprechen zweier Wortteile, zurückzuführen ist. Deshalb wird beim Training der zeitlichen Splits ein

zusätzlicher Anpassungsschritt durchgeführt, der sicherstellt, daß jedes Modell auch mit den richtigen Mustern trainiert wird, so daß eventuelle Unterschiede in den trainierten Modellen auch wirklich auf eine zeitliche Varianz zurückzuführen sind. Dieser Anpassungsschritt macht eine Verschiebung bezüglich des alten Modells überflüssig. Auf diese Problematik wird noch genauer in einem separaten Abschnitt eingegangen.

3.2 Trainieren

Nach dem Splitten erhalten wir also für jeden Zustand von jedem Wort vier neue Modelle. Diese werden nun trainiert, um sie optimal an die Sprachdaten anzupassen. Im allgemeinen wird beim Training von Erkennern für kontinuierliche Sprache folgendermaßen vorgegangen:

Erst wird für den zu trainierenden Satz ein HMM aufgebaut. Für jedes im Satz vorkommende Wort wird mit Hilfe des Lexikons ermittelt, mit welchen akustischen Einheiten es modelliert wird. Für jedes Wort wird ein HMM erstellt, das dann (evtl. mit Einfügen von optionaler Stille zwischen zwei Wörtern) zum Satz-HMM zusammengefügt wird. Dann wird in diesem Satz-HMM ein Viterbi-Pfad (oder ein Forward-Backward-Pass) berechnet und die Parameter der akustischen Modelle, z.B. mittels des Baum-Welch-Algorithmus, angepaßt.

In dem hier vorgestellten Algorithmus werden beim Training als Wort-HMMs die Zustandsgraphen verwendet. Einmal wird mit zeitlich gesplitteten Zuständen, einmal mit räumlich gesplitteten trainiert. Nach dem Training sollten die neuen Atome so eingestellt sein, daß die Splits in Raum und Zeit die Wahrscheinlichkeit, das jeweilige Wort zu beobachten, maximieren.

3.3 Clustern

Da sich die Wörter nach dem Splitten die neu entstandenen Atome nicht mehr teilen, entsteht nach dem Training eine ungeheure Menge an neuen Modellen. Bei einem Vokabular von 13000 Wörtern und einer (eher unter-) durchschnittlichen Anzahl von 10 Zuständen pro Wort sind es schon auf $13000 \cdot 10 \cdot 4 = 520000$ neue Atome. Es ist nicht sinnvoll, mit den üblicherweise vorhandenen Trainingsdaten, einen Erkenner mit einer derart großen Anzahl akustischer Modelle zu bauen. Es ist sowieso zu erwarten, daß viele Atome ungefähr das gleiche akustische Phänomen modellieren. Bei den Wörtern "versuchen", "vereinen", "verteidigen", "vereidigen", "Vernunft", z.B. dürften die Modelle, die den Wortanfang beschreiben, ziemlich ähnlich, wenn nicht identisch sein. Es bietet sich daher an, die neuen Atome zu clustern und die ähnlichsten Modelle zu vereinigen. Dies wird mit einer Variante des k-Mittelwerte (k-Means) Algorithmus erreicht, der zusätzlich die Gewichtung der einzelnen Atome berücksichtigt. Dabei sei unter Gewichtung eines Atoms die Anzahl der Trainingsmuster verstanden, die ihm vom Viterbi-Algorithmus insgesamt zugeordnet wurden. Das Bottom-Up Clustern wird hier nicht eingesetzt, da ein Verfahren mit quadratischem Aufwand bei dieser Anzahl

Modelle nicht brauchbar ist. Der Cluster-Algorithmus sieht folgendermaßen aus:

1. Erzeuge k neue Zentren. Diese sollten nach Möglichkeit mit den Werten der alten Atome initialisiert werden, und ansonsten gut über den von den Trainingsmustern besetzten Raum verteilt sein.

2. Berechne für jedes Atom seine Distanz zu den k Zentren und ordne es dem Zentrum zu, zu dem es die kleinste Distanz besitzt.

3. Berechne die Zentren neu, als die gewichteten Mittelwerte aller Atome, die ihnen zugeordnet wurden.

4. Gehe zurück zu 2, falls Abbruchbedingung noch nicht erreicht.

Als Distanzmaß zwischen Atomen wird die Entropie-Distanz benutzt. Je nachdem, ob es sich bei den Modellen um Gauß-Verteilungen aus Codebüchern oder lediglich um Gauß-Mixtur-Gewichte handelt, sehen die Formeln für die Entropie-Distanz anders aus.
Für Gauß-Verteilungen (Kullback-Leibler-Distanz):

$$d(f,g) = \int f(\mathbf{x}) \log \frac{f(x)}{g(\mathbf{x})} d\mathbf{x}$$

Für Mixtur-Gewichte:

$$d(f,g) = H(f+g) - \frac{1}{2}H(f) - \frac{1}{2}H(g)$$

mit $H(f) = -\sum_i f(i) \log_2 f(i)$ und $f+g$ Mittelwert der Mixturen f und g.

Es können auch einfachere Distanzmaße benutzt werden, wie z.B. der euklidische Abstand bei Mixtur-Gewichten.

Der Vorteil beim k-Means Clustern von Modellen ist, daß über die Anzahl der Zentren k direkt angegeben werden kann, wie viele neue Atome pro Iteration (des Gesamtalgorithmus) gewünscht sind. Es wird also nicht wie beim SSS-Algorithmus die Anzahl der erlaubten Splits festgelegt, sondern die Anzahl daraus entstehender Atome. Die Suche nach den n besten Splits ist so auf elegante Weise umgangen. Da nur k neue Atome für die gesamte Wortmenge zur Verfügung stehen, kann es passieren, daß zwei für einen Split vorgeschlagene Atome in dieselbe Klasse fallen. Dieser Split wird dann nicht vorgenommen. Nur die Splits, deren Atome sich beim Clustern unter allen vorgeschlagenen Atomen als besonders verschieden "bewährt" haben, werden dann tatsächlich durchgeführt.

Nach einer gewissen Anzahl Iterationen, oder wenn die Anzahl neuer Atome ein Maximum erreicht hat, können wir entscheiden, die Atommenge nicht weiter wachsen zu lassen. Wörter können dann immer noch ihre Zustände aufsplitten, unter Verwendung der aktuellen Menge von Atomen, ob diese sich im Vergleich zur letzten Iteration nun geändert hat oder nicht. Nur eine Frage muß jetzt noch geklärt werden: Ob ein Zustand im Raum oder in der Zeit aufgeteilt wird. Dies geschieht in der letzten Phase.

3.4 Auswählen der Splits und Aufbauen neuer Zustandsgraphen

Bei dem Clustern wurde jedes der vier von einem Zustand eines Wortes erzeugten Atome einer Klasse zugeteilt. Der Split eines Zustands soll nun nicht mit diesen Atomen durchgeführt werden, sondern mit den Repräsentanten (Zentren) der Klassen dieser Atome. Auf diese Art teilen sich die aufzubauenden Zustandsgraphen der Wörter automatisch die neu erzeugten Modelle. Da aber das Zentrum einer Klasse aus dem gewichteten Mittelwert aller Atome, die sie enthält, gebildet wird, liegt es im allgemeinen nicht an dem vom Split vorgeschlagenen Ort im Merkmalsraum. (Abb. 3.4)

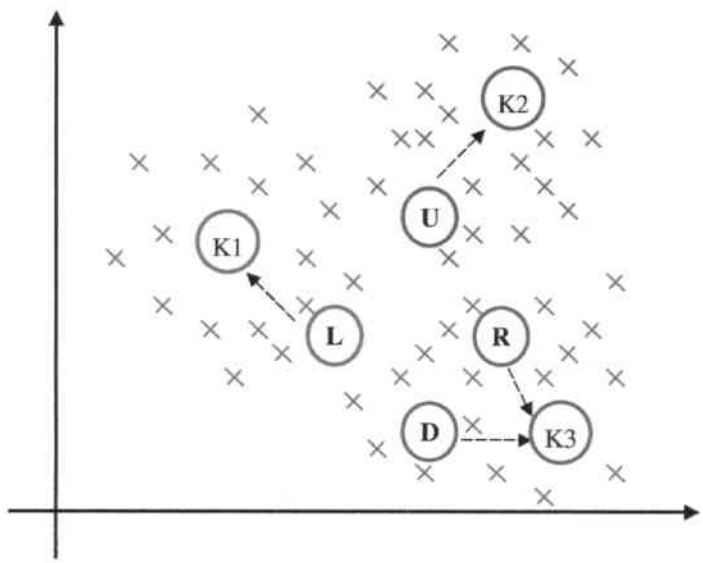

Abbildung 3.4: Die trainierten Atome L, R, U, D und ihre Klassen. Zur Einfachheit wird angenommen, die Atome seien Gaußverteilungen (symbolisch durch einen Kreis im zweidimensionalen Merkmalsraum dargestellt. Kreuze repräsentieren andere Atome derselben Klasse.)

Es muß also entschieden werden, ob für den Zustand unter diesen Bedingungen überhaupt noch ein Split durchgeführt wird und wenn ja, ob ein räumlicher oder ein zeitlicher Split vorteilhafter wäre.

3.4.1 Auswahl der Splits

Falls die aus dem zeitlichen Split gewonnen Atome L und R, sowie die aus dem räumlichen Split gewonnenen Atome U und D zusammenfallen, so werden beide Splits verworfen. Offensichtlich war in allen Vorkommen dieses Wortes nicht genügend

zeitliche Fluktuation und auch zu wenig Unterschied in der Aussprechweise, um eine Aufteilung dieses Zustands zu rechtfertigen. An dieser Stelle des Graphen bleibt deshalb ein einziger Zustand, der z.B. mit dem Atom L modelliert wird. (Abb. 3.5)

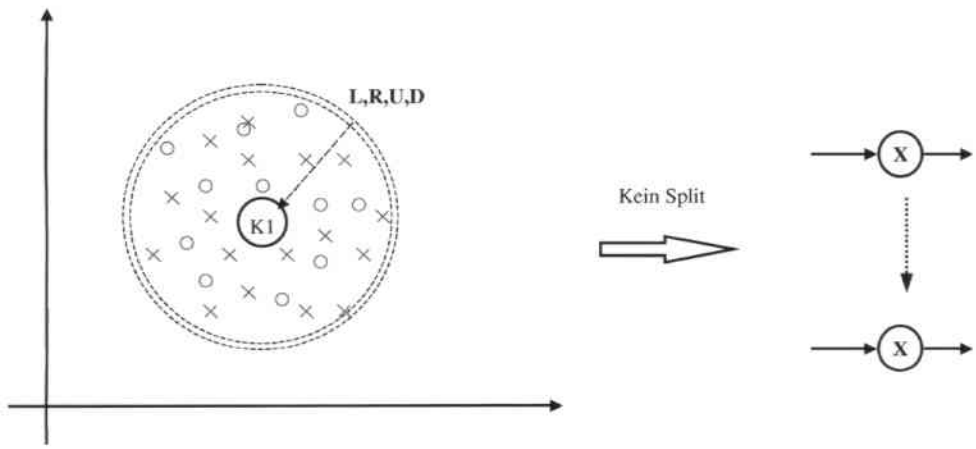

Abbildung 3.5: Keine bedeutende Variation in der Beobachtung. (Gaußverteilungen für L, R, U, D durch gestrichelte Ellipsen angedeutet. Merkmalsraum auf zwei Dimensionen reduziert. Es werden nur zwei Merkmalsfolgen betrachtet (zwei Vorkommen des Wortes). Muster aus der ersten Folge werden in Rot, die aus der zweiten in Grün dargestellt. Dabei repräsentieren Kreuze die in der ersten Hälfte des Sprachsignals auftauchenden Muster; Kreise, die aus der zweiten Hälfte. K1 ist die gemeinsame Klasse, zu der L, R, U und D geclustert werden)

Falls L und R zusammenfallen, U und D aber nicht, war in den Sprachmustern eine Variation vorhanden, die wohl auf unterschiedliche Aussprachearten dieses Wortteils zurückzuführen ist, z.B. wegen der Tonlage des Sprechers, seinem Dialekt, dem Kontext, in dem das Wort vorkommt, usw. An dieser Stelle des neuen Zustandsgraphen werden die Atome U und D parallel eingefügt. (Abb. 3.6)

Falls nur U und D zusammenfallen, aber L und R nicht, ist die Varianz im ursprünglichen Modell eindeutig auf zeitliche Variation im Sprachsignal zurückzuführen. L und R werden in Serie am Platz des alten Atoms in den Zustandsgraphen eingefügt. (Abb. 3.7)

Im letzten Fall, wenn sowohl ein räumlicher als auch ein zeitlicher Split zulässig wäre, muß entschieden werden, welcher der beiden durchgeführt wird. Es soll derjenigen Split gemacht werden, der den größten Informationsgewinn bringt. Deshalb wird wieder ein Entropiemaß benutzt, um die Distanzen distZ zwischen L und R, und distR zwischen U und D zu bestimmen. Falls $distZ > distR$ wird der zeitliche Split vorgenommen, ansonsten der räumliche. (Abb. 3.8)

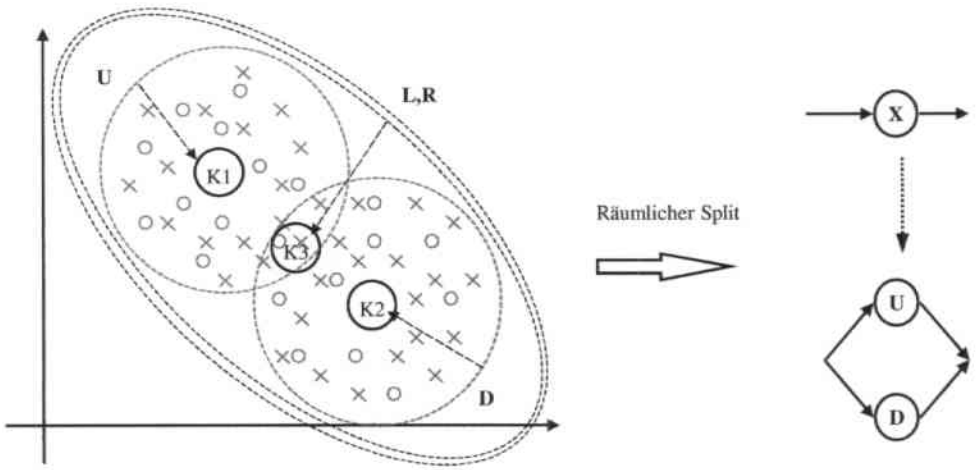

Abbildung 3.6: Räumliche Variation. U und D werden verschiedenen Klassen K1 und K2 zugeteilt; L und R, der gleichen Klasse K3.

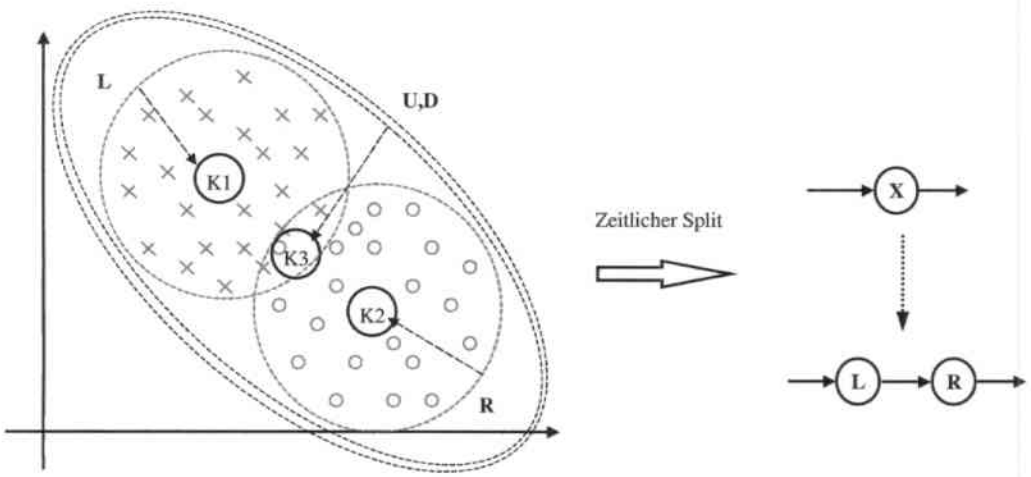

Abbildung 3.7: Zeitliche Variation. U und D werden der gleichen Klasse K3 zugeteilt.

Am Ende entsteht ein neues HM-Netz für jedes Wort, in dem einzelne Zustände zu ihrer Modellierung eine beschränkte Menge neuer Atome untereinander und mit Zuständen anderer Wort-HM-Netze teilen. (Abb. 3.9)

3.4.2 Überlegungen zur Größe der Zustandsgraphen

Wenn genügend Trainingsdaten zur Verfügung stehen und die maximale Anzahl akustischer Atome auf eine vernünftige Zahl beschränkt ist, kann erwartet werden,

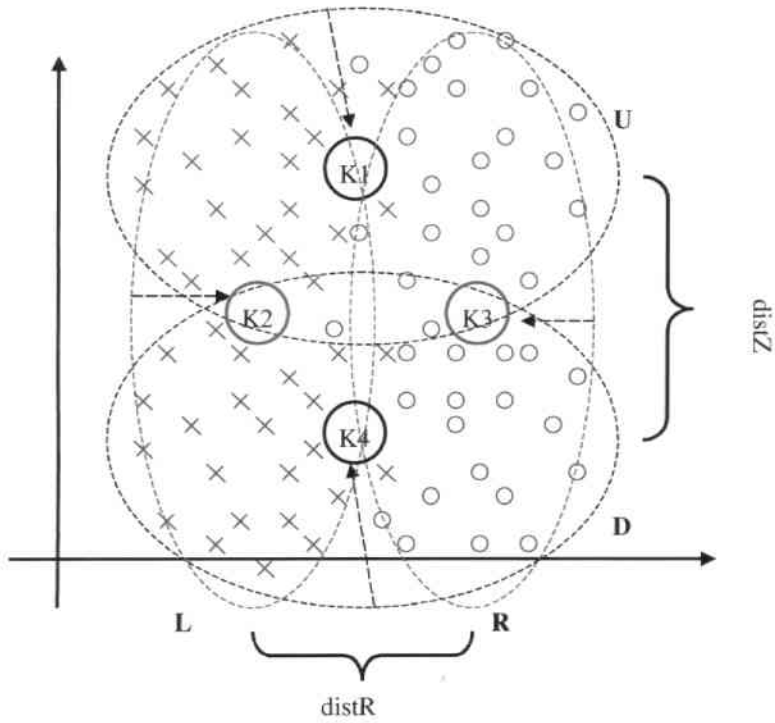

Abbildung 3.8: Räumliche und Zeitliche Variation.

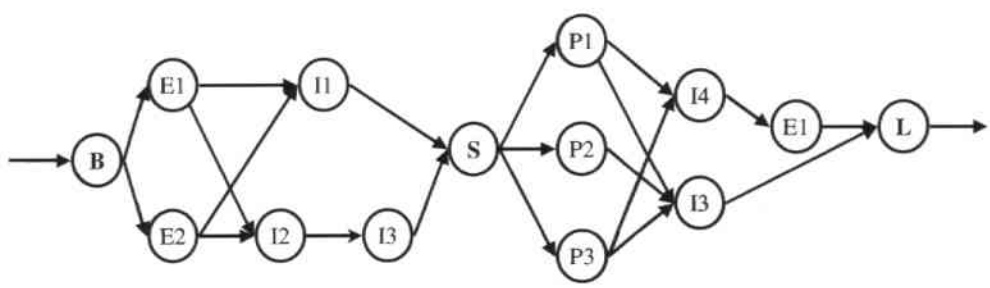

Abbildung 3.9: Beispiel für resultierenden Zustandsgraphen.

eine für spontane, kontinuierliche Sprache repräsentative Menge von Atomen zu erhalten. Dabei bleibt die Länge (oder besser Tiefe) der mit ihnen aufgebauten Zustandsgraphen automatisch beschränkt:

Betrachten wir zum Beispiel das Wort "Amerika". Gäbe es keine Grenze für die Anzahl neuer Atome, so könnte der Anfangsteil des Wortes, mit einer ziemlich langen Kette nur leicht verschiedener Atome für den Laut "A" modelliert werden. Die Länge der Kette wäre nur durch die Anzahl der Trainingsmuster beschränkt, die für diesen Wortteil

existieren.
Wenn aber nur wenige repräsentative Atome zur Verfügung stehen, ist zu erwarten, daß nur eine kleine Anzahl davon einen Bereich im Merkmalsraum abdeckt, der den Laut "A" repräsentiert. Nur diese Atome können zur Modellierung des Wortanfangs benutzt werden. Sollte versucht werden, in einer nächsten Iteration eines dieser Atome weiter aufzusplitten, würden die entstehenden Atome nach dem Clustern wahrscheinlich zusammenfallen, da nur eine begrenzte Anzahl Zentren zugelassen ist. Dies gilt natürlich auch für alle anderen Zustände von "Amerika". Aus den selben Gründen kann ein Zustandsgraph auch nicht zu stark in die Breite wachsen. Er hört auf, sich zu erweitern, sobald er mit den zur Verfügung stehenden akustischen Atomen nicht mehr effizienter gestaltet werden kann.

Trotzdem kann es vorkommen, daß wegen ungeschickter Wahl der Splits beim Aufbau der Zustandsgraphen überflüssige Zustände eingebaut werden. Angenommen, ein Wortteil werde einmal "AO", einmal "AU" ausgesprochen. Wenn unglücklicherweise die Aufteilung im Raum vor der in der Zeit durchgeführt wird, entsteht ein unnötig verdoppelter "A"-Zustand. Dies ist bei der Wahl der Splits vielleicht nicht vermeidbar, weil die räumliche Aufteilung zunächst tatsächlich mehr Informationsgewinn bringt. Um dieses Problem der verdoppelten Zustände zu beheben ist es möglich, in einer letzten Phase den Zustandsgraphen noch zu bereinigen. (Abb. 3.10)

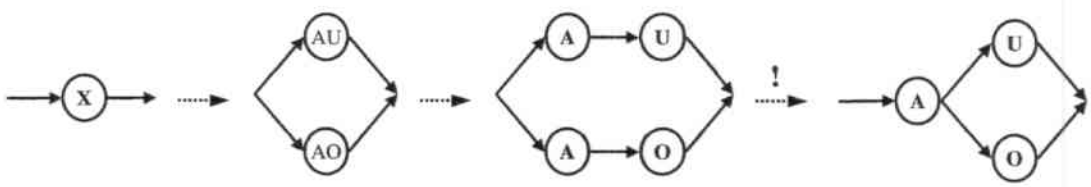

Abbildung 3.10: Reduktion der Breite des Zustandsgraphen.

3.5 Zusammenfassung des Algorithmus:

Zusammengefaßt stellt sich der Algorithmus wie folgt dar:

1. Starte mit trainierten Anfangsatomen und linearen Zustandsgraphen für jedes Wort.

2. Splitte für jedes Wort die Zustände seines Graphen in Raum und in Zeit. Es entsteht für jedes Wort eine Menge neuer Kandidat-Atome.

3. Trainiere diese Kandidaten mittels Viterbi, indem der Zustandsgraph eines Wortes einmal mit räumlichen, einmal mit zeitlichen Splits als HMM für dieses Wort benutzt wird.

4. Clustern der Kandidaten: Weise jedem Kandidaten eine von k Klassen zu. Die Zentren dieser Klassen werden als neue Atome benutzt.

5. Baue die neuen Zustandsgraphen auf. Wähle für jeden Zustand die beste Vorgehensweise (räumlicher Split, zeitlicher Split, kein Split) und füge die entsprechenden Atome in den Graphen ein. Hier ist es möglich, noch einen letzten Reduktionsschritt vorzunehmen, in dem unnötig verdoppelte Zustände zusammengefaßt werden.

6. Gehe wieder zu Punkt 2, bis sich an den Zustandsgraphen nichts mehr ändert, oder eine gewählte Zahl Iterationen überschritten wurde.

3.6 Probleme bei der Initialisierung und dem Training zeitlicher Splits

Das zeitliche Aufsplitten eines Zustands wirft einige nicht ganz triviale Probleme auf, auf die es sich lohnt, näher einzugehen.

Beim temporalen Splitten eines Zustands, der ursprünglich mit dem Atom O modelliert wurde, werden an seiner Stelle zwei neue Zustände in Serie eingefügt, die jeweils mit eigenen Atomen L und R ausgestattet sind. Wenn wir, genau wie beim räumlichen Split, die Atome L und R einfach mit den Werten von O initialisieren, sie um einen kleinen Wert $\pm e$ entlang der Richtung der größten Varianz verschieben, und dann mittels Viterbi trainieren, erhalten wir keinesfalls immer eine Aufteilung, die eine Zeitliche Variation im Sprachsignal wiederspiegelt. Um dies zu verdeutlichen, analysieren wir die folgenden zwei stark vereinfachten Fälle:

3.6.1 1. Fall: Nur räumliche Variation

In dem betrachteten Wortteil ändert sich das Sprachsignal (und damit die Trainingsmuster) nicht mit der Zeit. Dafür ist es von Sprecher zu Sprecher, oder je nach Kontext sehr unterschiedlich. Bei der Berechnung des Viterbi Pfades, der jedem Muster eindeutig ein Atom zum Training zuordnet, entsteht folgendes Szenario (Abb. 3.11):

Der Viterbi Algorithmus, der versucht, die Wahrscheinlichkeit der Beobachtung A1 A2 A3 A4 zu maximieren, wird A1, A2 und A3 dem Modell zuordnen, das ihnen im Merkmalsraum am nächsten ist, also L. Das letzte Muster A4 wird nur R zugeordnet, weil der Algorithmus den Pfad durch das HMM L-R vervollständigen muß. Für die Aussprechvariante B1 B2 B3 B4 hingegen werden B2, B3 und B4 dem Atom R zugeordnet. Für das erste Muster B1 wäre zwar von der Wahrscheinlichkeit her R vorzuziehen, aber es muß L zugeordnet werden, wieder um den Pfad zu komplettieren.

Beim Anpassen der Modellparameter (z.B. mit Baum-Welch) wird L also mit allen Mustern aus der Variante A trainiert (außer dem letzten Muster), und R mit denen

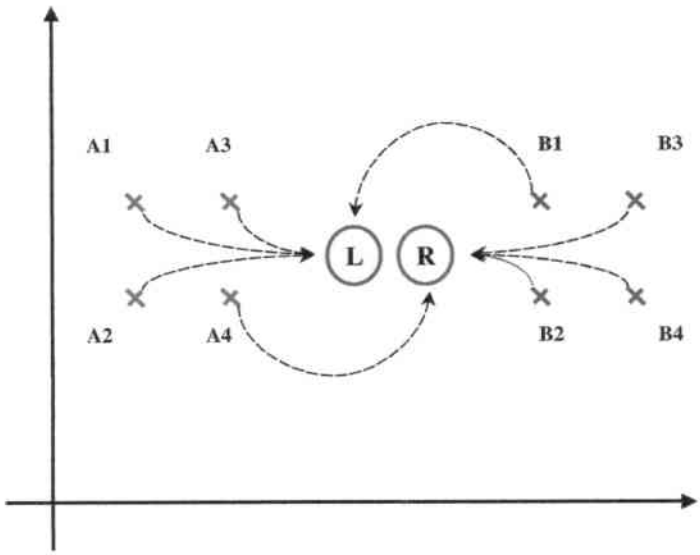

Abbildung 3.11: Viterbi-Zuordnung von Mustern zu Atomen (aus zeitlichem Split) bei räumlicher Variation. (Merkmalsraum auf zwei Dimensionen vereinfacht. A1 A2 A3 A4: Muster für erste Aussprechvariante in der Reihenfolge des Auftretens. B1 B2 B3 B4: Muster für zweite. Atome L, R entlang der Richtung größter Varianz initialisiert.)

aus der Variante B (außer dem ersten.) Die so trainierten Atome L und R passen sich also der räumlichen Variation an. Die Distanz zwischen ihnen wird sehr groß, obwohl es überhaupt keine zeitliche Variation im Signal gibt.

Auch wenn ein Weg gefunden wird, L und R entlang der größten zeitlichen Varianz zu initialisieren, ist das Problem nicht gelöst. Da es in dem vorliegenden Fall gar keine zeitliche Varianz gibt, ist diese Richtung undefiniert. In diesem Beispiel nehmen wir deshalb an, sie wäre genau senkrecht zur Richtung der größten räumlichen Varianz gewählt (ansonsten ist dieser Fall auf den vorherigen reduzierbar.) Dennoch erhalten wir keine klare Zuordnung (Abb. 3.12).

Die einzige sichere Aussage, die gemacht werden kann, ist daß A1 und B1 immer L, A4 und B4 immer R zugeordnet werden (aus dem selben Grund wie oben.) Für A2, A3, B2 und B3 ist keine sichere Aussage machbar. Falls alle Muster aus A identisch sind, ist es Implementationssache, wem A2 und A3 zugeordnet werden. Falls nicht, reicht die kleinste Abweichung aus, um den berechneten Viterbi Pfad stark zu verändern.

Das Verhalten der so trainierten Atome L und R ist demnach nicht eindeutig vorhersagbar. Sie werden sich wohl mehr oder weniger in Richtung jeweils eines der räumlichen Ballungszentren bewegen. Nur in extrem idealisierten Fällen bleiben sie beide im Mittelpunkt der Ballungszentren. Die Distanz zwischen L und R bleibt also auch hier nicht unbedingt wie gewollt klein.

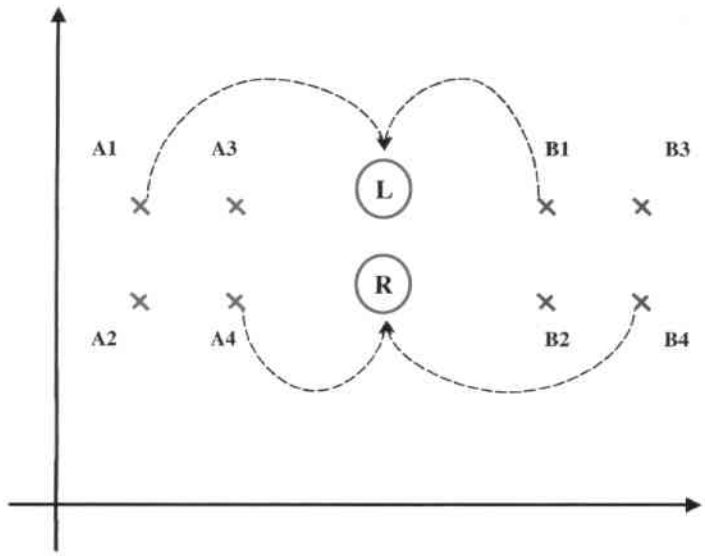

Abbildung 3.12: Zuordnung bei Initialisierung senkrecht zur Richtung der größten Varianz.

3.6.2 2. Fall: Nur zeitliche Variation

Der betrachtete Wortteil wird von jedem Sprecher in jeder Situation fast gleich ausgesprochen. Das Sprachsignal weist fast nur zeitliche Variation auf, wie z.B. bei den lauten "AU" oder "EI". Je nachdem, wie L und R auf der Achse der größten räumlichen Varianz angeordnet sind, passiert bei der Viterbi Pfad Berechnung folgendes (Abb. 3.13, 3.14):

Falls L zufällig in Richtung der Muster initialisiert wurde, die am Anfang des Lauts vorkommen, läuft alles wie erhofft. A1 bis A4 werden dem Atom L zugeordnet, A5 bis A8 dem Atom R. Nach dem Training sollte sich L den Mustern aus dem ersten Lautteil angepaßt haben, R denen aus dem zweiten.

Im gegenteiligen Fall aber ist nur sicher, daß A1 L und A8 R zugeordnet wird. Für die restlichen Muster gibt es zwei Möglichkeiten. Die schlimmste Wahl für den Viterbi-Algorithmus wäre, A1 bis A4 L und A5 bis A8 R zuzuordnen. Dann wäre jedes Muster dem schlechtmöglichsten Modell zugeteilt worden. Es wird also nur A1 an L gehen und alle restlichen Muster an R. So wird zumindest die Wahrscheinlichkeit, A2 bis A8 mit R zu beobachten, im Mittel nicht so gering. Die zweite (symmetrische) Lösung ist, nur A8 Atom R und die restlichen Muster Atom L zuzuteilen. Beide Lösungen bewirken, daß eines der Atome sich beim Training dem Mittelpunkt der Ballungszentren nähert, während das andere nur mit den allerersten (oder allerletzten) Mustern trainiert wird.

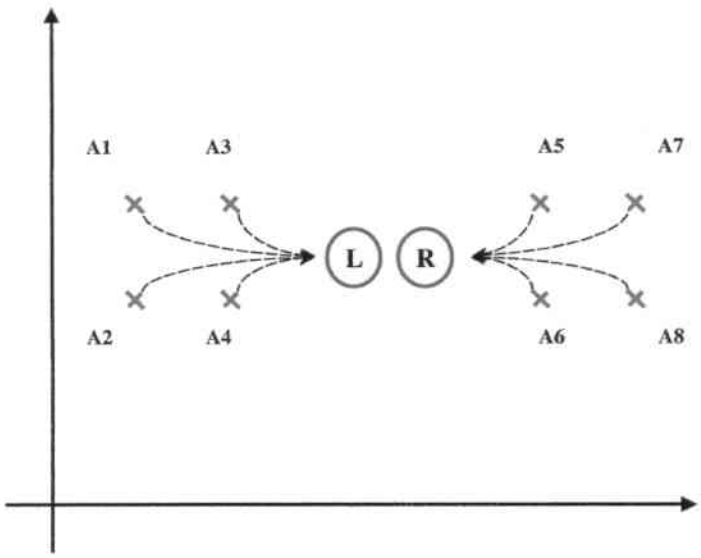

Abbildung 3.13: Zuordnung bei richtiger Initialisierung. (A1-A4: Muster für ersten Teil des Lauts, A5-A8: Muster für zweiten Teil. Bei dem Laut AU z.B. wären A1-A4 im Teil des Merkmalsraums, der Laute ähnlich wie "A" repräsentiert, A5-A8 im Teil für "U".)

Es wäre also nötig, um die Initialisierung korrekt vorzunehmen, vorher die erwartete Lage der ersten und der letzen Muster im Merkmalsraum zu kennen. Dies wäre mit einer Heuristik durchaus machbar. Alternativ kann man den Wortteil zweimal trainieren. Nach dem ersten Training dürften L und R in einer günstigeren Lage sein, so daß sie sich beim zweiten Durchlauf wie erwartet an die Muster anpassen.

3.6.3 Zusammengefaßt:

Das Trainieren der Atome für einen zeitlichen Split mit einem Viterbi Pfad ist also nur mittels sorgfältiger, aufwendiger Initialisierung möglich. Darüber hinaus liefert es unerwünschte Ergebnisse, wenn in dem trainierten Wortteil keine zeitliche Variation vorhanden ist. Aus diesem Grund wurde entschieden, das Problem mit einem schnellen, einfachen Verfahren zu lösen:

Beim Trainieren eines Wortteils wird grundsätzlich die zeitlich erste Hälfte der Muster dem Atom L, und die zweite dem Atom R zugeordnet. (Abb. 3.15)

Dieses Verfahren liefert nicht immer den besten Split. Es garantiert aber zumindest, daß die Atome L und R bei Fehlen von zeitlicher Variation in den Mustern nach dem Training praktisch zusammenfallen. (Abb. 3.16)

Die starre Einteilung der Muster hat zur Folge, daß eins der Atome L und R auch

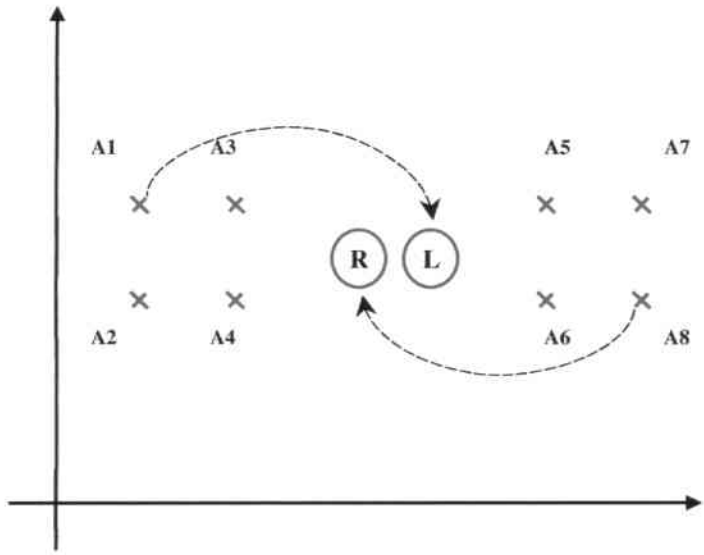

Abbildung 3.14: Zuordnung bei falscher Initialisierung.

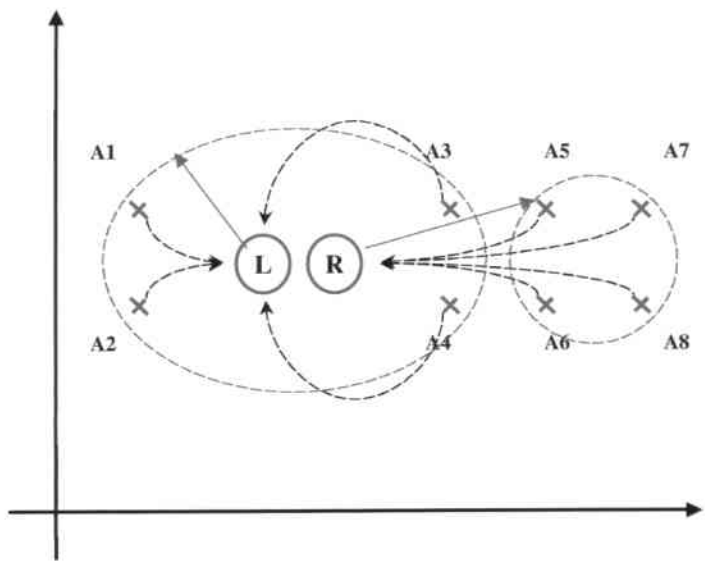

Abbildung 3.15: Starre Zuordnung von Mustern zu Atomen bei zeitlicher Variation.

falsche Muster mittrainiert, wenn nicht genau die erste Hälfte der Muster den ersten Lautteil repräsentiert. Dies wird aber später durch weiteres Aufsplitten der entstandenen Atome relativiert. Allerdings kann ein solches mehrmaliges Splitten wiederum ein unnötiges Verdoppeln von Zuständen zur Folge haben. Das Problem

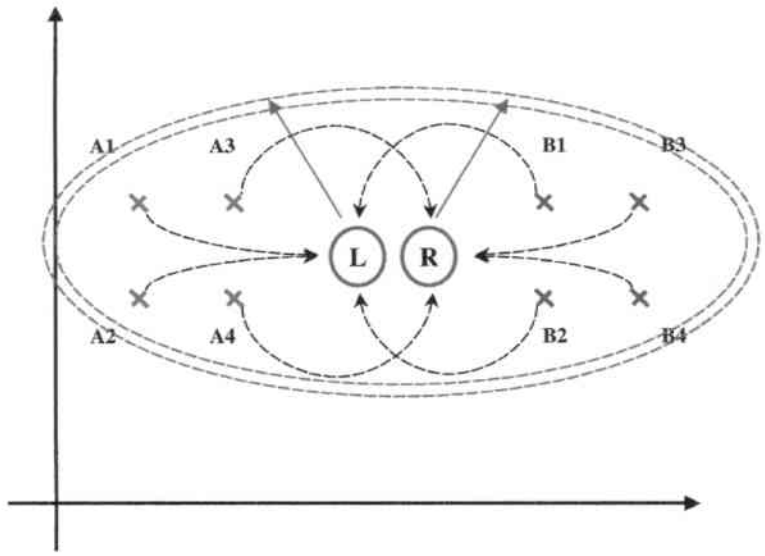

Abbildung 3.16: Starre Zuordnung bei räumlicher Variation. Die Muster, die L und R zugeteilt werden, sind gleichermaßen stark über den Merkmalsraum verstreut. Ihre Verteilungen fallen somit praktisch zusammen.

kann behoben werden, indem in einem letzten Reduktionsschritt, genau wie bei paralleler Verdoppelung, die betroffenen Zustände fusioniert werden. (Abb. 3.17)

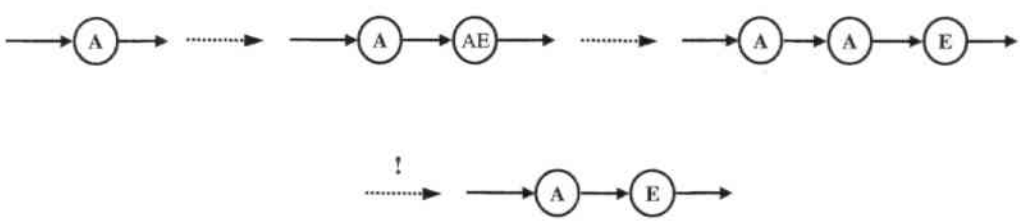

Abbildung 3.17: Reduktion der Länge des Zustandsgraphen.

3.7 Aufbauen des Aussprachelexikons

Wie an dem Algorithmus unschwer zu erkennen ist, wird das Aussprachelexikon Schritt für Schritt in jeder Iteration aufgebaut, und zwar in Form von Zustandsgraphen. Für jedes Wort wird nämlich ein Zustandsgraph aufgebaut, der nicht nur angibt, aus welchen akustischen Einheiten es aufgebaut ist, sondern der auch alle (oder die

wichtigsten) im Trainingsmaterial vorkommenden Ausspracheformen enthält. Der Unterschied zu herkömmlichen Lexika ist, daß diese Information nun nicht in Form von Listen vorliegt, sondern in Form eines mehr oder weniger stark verzweigten Graphen. Außerdem ist an dem neuen Lexikon für Menschen nicht mehr einfach erkennbar, wie ein Wort denn ausgesprochen wird. Es wird also schwieriger zu erkennen, wo das System evtl. Fehler gemacht hat.

Es ist schwierig vorherzusehen, wie lange der Algorithmus brauchen wird, bis sich die Zustandsgraphen stabilisiert haben, wenn mit einem einzigen Atom pro Wort angefangen wird. Es ist aber möglich, den Prozeß zu beschleunigen, durch eine geeignete Initialisierung der Länge der Zustandsgraphen. Ohne den Anspruch fallen zu lassen, das Lexikon automatisch, ohne menschlichen Eingriff aufzubauen, ist es möglich, diese Länge wie folgt zu ermitteln:
Ein einfaches Programm zur Sprachsynthese liefert für jedes eingegebene Wort eine geeignete Phonemfolge. Diese kann nun zum Aufbau eines Zustandsgraphen benutzt werden. Er besteht aus einer einfachen Kette von Zuständen, deren Länge der Anzahl ausgegebener Phoneme entspricht. (Abb. 3.18)

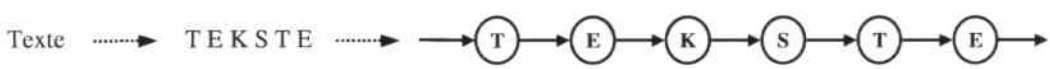

Abbildung 3.18: Erzeugung einer Phonemfolge aus einem Text und anschließender Aufbau des Zustandsgraphen.

Diese primitiven Zustandsgraphen reichen natürlich bei weitem nicht aus, um gesprochene Sprache zu erkennen. Sie dienen nur als Starthilfe für das System, das nun schon leichter Wortgrenzen im Sprachsignal finden und grobe Zuordnungen von Trainingsmustern zu Zuständen im Wort-Zustandsgraph machen kann. Jetzt muß nur noch die Menge der Atome vergrößert und die Zustandsgraphen erweitert, angepaßt und verfeinert werden.

Selbst wenn das Phonemsyntheseprogramm nicht für die betrachtete Sprache konzipiert war, und die initiale Konfiguration sehr schlecht ist, sollte der Algorithmus viel schneller zu einem guten Ergebnis kommen, als wenn er die Zustandsgraphen von Grund auf aufbauen muß. Anstatt eines Syntheseprogramms kann man genauso ein mit Regeln automatisch erstelltes Lexikon benutzen, um eine anfängliche, auf Phonemen basierte Aussprache zu erhalten. Die durch Anwendung von Regeln gemachten Fehler, z.B. bei nicht berücksichtigten Ausnahmefällen, sollen dann während der Verfeinerung vom System automatisch beseitigt werden.

Mit derselben Methode sollte es auch möglich sein, schon existierende, z.B. auf kontextabhängigen Phonemen aufbauende Systeme zu verbessern. In ihrem Aussprachelexikon, in denen die Wörter meist noch durch kontextunabhängige Phoneme

beschrieben werden, müßten diese lediglich durch ihre kontextabhängigen Entsprechungen ersetzt werden. Dann können sogar die trainierten, kontextabhängigen Phoneme benutzt werden um die Zustände des initialen Graphen zu modellieren. (Abb. 3.19)

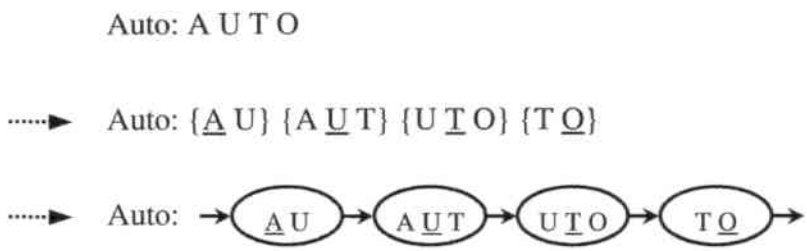

Abbildung 3.19: Initialisierung mit kontextabhängigen Phonemen.

Nach demselben Muster ist es auch möglich, neue Wörter in jeder beliebigen Iteration des Algorithmus ins Lexikon einzufügen. Es muß für sie nur vor der nächsten Split-Phase ein Zustandsgraph erstellt werden, der entweder noch im Grundzustand ist (nur ein Atom) oder beliebig komplex initialisiert ist. Wir können auch entscheiden, ob diese Wörter aktiv beim Clustern mitwirken oder ob die von ihnen, für Splits vorgeschlagenen Atome lediglich Klassen zugeordnet werden, ohne auf die Berechnung des neuen Zentrums Einfluß zu nehmen. So kann das System erst auf einer kleineren Menge häufig auftretender Wörter arbeiten, bis die erzeugten Atome und Zustandsgraphen diese zufriedenstellend genau modellieren. Dies sollte um einiges schneller gehen, als mit der gesamten Wortmenge. Dann kann die Menge von Atomen festgesetzt, und keine weiteren Änderungen an ihr zugelassen werden. Die bei Splits entstehenden Atome U, D, L, R werden nur noch existierenden Atom(klass)en zugeteilt, ohne daß jemals neue Zentren berechnet werden. An den früher trainierten Zustandsgraphen sollte sich nichts mehr ändern, während die neuen Zustandsgraphen immer noch in der Lage sind, ihre Zustände unter Verwendung existierender Atome aufzusplitten.

Kapitel 4

Experimente und Ergebnisse

Das Verfahren wurde auf dem Spracherkennungssystem JANUS implementiert, das in einer Zusammenarbeit der Universität Karlsruhe und der Carnegie Mellon University (Pittsburgh) entwickelt wird. Das JANUS System bietet die nötige Flexibilität, um auf niedrigster Ebene Anpassungen an der Akustik vorzunehmen, und gleichzeitig komplexe Mechanismen in der höheren Programmiersprache TCL zu entwerfen. Darüber hinaus ermöglicht dies einen direkten Vergleich mit anderen Systemen, die schon am ILKD (Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe) entwickelt wurden. Das Training und die Tests fanden in einer Unix-Umgebung statt.

Für die Sprecherdaten wurde die VERBMOBIL Datenbank benutzt. Sie enthält Aufzeichnungen von spontanen Dialogen zwischen zwei Personen in deutscher Sprache. Der Diskursbereich ist auf die telefonische Terminvereinbarung beschränkt. Für das Training wurden die Daten von 2641 Sprechern benutzt. Diese enthalten 13468 verschiedene Wörter. Die Testmenge umfaßt 30 Sprecher und 7215 verschiedene Wörter.

4.1 Das Referenzsystem

Das Referenzsystem (im folgenden S0), an dem die Leistung des neuen Verfahrens gemessen wird, ist ein am ILKD entwickelter kontextabhängiger Erkenner, auf Basis von 50 Phonemen. Neben 42 Phonemen für die Wiedergabe von Wörtern

@ A AEH AH AI AU B CH X D E E2 EH ER2 EU F G H I IE J K L M N NG O OE OEH OH P R S SCH T TS U UE UEH UH V Z

sind auch eins zur Modellierung von Stille (SIL) und 7 "Phoneme" zur Modellierung von nichtsprachlichen Geräuschen enthalten.

+QK +hGH +hEH +hEM +hHM +nGN +nKL

Die 7 speziellen Phoneme repräsentieren jeweils: Ein falsch ausgesprochenes Wort (z.B. bei Stottern oder Neuanfang), ein generelles menschliches Geräusch, ein menschliches

"äh", ein "ähm", ein "hm", ein generelles nichtmenschliches Geräusch und ein Klicken. Hier werden diese Geräusche also jeweils durch ein akustisches Modell beschrieben.

Das Referenzsystem S0 verfügt noch über einzelne Modelle für jedes kontextunabhängige Phonem. Die Zahl der kontextabhängigen Modelle hingegen wurde durch Clustern mit einen gewichteten Entropiedistanz-Maß, auf 2647 beschränkt. Es ist also ein HMM-basiertes, vollkontinuierliches System mit 2647 Codebüchern zu jeweils 12 32-dimensionalen Gaußverteilungen. Letztere benutzen diagonale Kovarianzmatrizen.

Die Merkmalsvektoren werden von dem JANUS System in einer Reihe von Signalverarbeitungsschritten aus den Sprecherdaten erzeugt. Als Merkmale wurden die Cepstralkoeffizienten gewählt. In jedem Zeitframe (das sind 10 ms) werden 13 Koeffizienten ermittelt, mit ihren ersten und zweiten Ableitungen, berechnet über einem diskreten Zeitintervall von ± 3 Frames. Diese 39 Merkmale werden dann durch eine LDA-Transformation auf 32 reduziert.

Beim System S0 wurde keine Vokaltraktlängennormierung vorgenommen. Stattdessen wurde eine Mittelwert-Subtraktion (Cepstral Means Substraction) vorgenommen: Über alle Merkmalsvektoren eines Sprechers wurde zunächst ein Mittelwert gebildet und die Cepstralkoeffizienten danach anhand dieses Mittelwertes normiert.

4.2 Das System zur Automatischen Generierung von Atomen

Bei der Implementierung des neuen Split-Algorithmus wurde darauf verzichtet, die Parameter von Gaußverteilungen anzupassen. Nicht nur verlangsamt es das Training, es verlangt auch noch aufwendige Distanzberechnungen beim Clustern, was zu unvertretbar langen Rechenzeiten führt. Stattdessen werden nur die Mixturgewichte der Gaußverteilungen angepaßt. Es wurde ein semikontinuierliches System (im folgenden S1) gebaut, mit einem einzigen Codebuch zu 256 Gaußverteilungen. Diese benutzen immer noch diagonale Kovarianzmatrizen und dieselben Merkmalsvektoren wie S0. Um die Referenzvektoren des Codebuchs zu initialisieren wurde ein k-Means Algorithmus auf der gesamten Trainingsmenge angewendet. Diese Parameter werden in Zukunft nicht mehr verändert.
Jede Mixturgewichtverteilung besteht also aus 256 Gewichten für die Gaußverteilungen des Codebuchs. Im folgenden verstehen wir unter akustischem Atom nur eine solche Mixturgewichtverteilung. Nur sie werden beim Training und beim anschließenden Clustern angepaßt.

Es wurde entschieden, wie schon in Abschnitt 3 angedeutet, eine Initialisierung der Zustandsgraphen auf Basis von Phonemen durchzuführen. Um eine initiale Kette von Phonemen zu bekommen, wird der Text-zu-Phonem Konvertierer verwendet, der im

Rahmen des Sprachsynthesprogramms HADIFIX [1], an der Universität Bonn entwickelt wurde. Er liefert zu jeder eingegebenen Textfolge eine entsprechende Lautschrift, basierend auf dem SAMPA Phonemsatz [2]. Als initiale Atome für die Zustandsgraphen werden die kontextunabhängigen Modelle des Systems S0 verwendet.

4.3 Versuche

Versuch 1

Ein erster Versuch wurde gestartet, bei dem die Menge der zu splittenden Wörter vorerst auf 100 beschränkt wurde (die 100 häufigsten.) Dies macht das System (im folgenden S1) kleiner und schneller, und ermöglicht eine relativ einfache Analyse seines Verhaltens beim Splitten und beim Aufbauen von Zustandsgraphen.
Es wurden 3 Iterationsschritte durchgeführt. Nur für die 100 häufigsten Wörter wurden Splits vorgenommen, Atome geclustert und neue Zustandsgraphen aufgebaut. Den restlichen Trainingswörtern wurde nur ein gemeinsamer Zustandsgraph zur Verfügung gestellt. Dieser "Generelles Wort"-Graph enthält anfangs nur einen Zustand, der aber mit der Zeit auch erweitert wird. So werden zwar für weniger gebrauchte Wörter vorerst keine individuellen Graphen aufgebaut, ihre Trainingsmuster haben aber dennoch einen Einfluß, beim Clustern, auf die Bestimmung neuer Atome. Die Berechnung der Mittelwerte der Cluster geschieht nämlich gewichtet, wobei das Gewicht eines Atoms die Anzahl der Trainingsmuster ist, die ihm vom Viterbi- Algorithmus zugeordnet wurden. Die Atome, die beim Split von "Generelles Wort"-Zuständen entstehen haben also meistens große Gewichte, obwohl die Wörter, deren Teile sie modellieren, einzeln genommen nur selten vorkommen.

Im 4. Schritt erhielten auch die restlichen 13368 Wörter des Trainingsvokabulars eigene Zustandsgraphen. Diese wurden zu Beginn wie bei den 100 Startwörtern mit den vom Text-zu-Phonem Konvertierer gelieferten Phonemen in linearer Form initialisiert. Für alle Wörter wurden Splits durchgeführt und die resultierenden Atome trainiert. Beim Clustern mußte allerdings eine Änderung des Algorithmus vorgenommen werden.

Bei dem Versuch, ungefähr 400000 Atome zu Clustern (13468 Wörter, durchschnittlich 8 Zustände pro Wort, 4 Atome pro Zustand) mußte leider festgestellt werden, daß dies zu für Entwicklungszwecke unangenehm langen Rechenzeiten führt. Und das trotz Implementierung zeitkritischer Routinen in C und Neukompilierung des JANUS Systems. Denn selbst bei der kleinen Anzahl von 100 Klassenzentren sind 40000000 Distanzberechnungen zwischen 256-wertigen Mixturgewichtverteilungen pro Clusteriteration nötig. Es mußte also ein Weg gefunden werden, diesen Schritt zu beschleunigen. Da viele Wörter nur selten in den Trainingsdaten vorkommen, haben die meisten Atome, die geclustert werden sollen, sehr kleine Gewichte und beeinflussen

[1] Siehe www.ikp.uni-bonn.de

[2] Sampa ist ein maschinenlesbarer Phonemsatz der 1987-89 im Rahmen des ESPRIT Projektes von einer internationalen Gruppe von Phonetikern erstellt wurde. Siehe auch www.phon.ucl.ac.uk/home/sampa

kaum die Lage der Zentren. Aus diesem Grund wurde der folgende Lösungsansatz getestet:
In den ersten Iterationen des Cluster-Algorithmus werden nur die Atome der n häufigsten Wörter bei Distanzberechnungen und Mittelwertbildung berücksichtigt (wir nennen diese Atome und die entsprechenden Wörter "aktiv", die restlichen "passiv".) Nach ein paar Schritten sollten die Klassenzentren dann schon eine günstige Lage erreicht haben, bevor in der letzten Clusteriteration wieder alle Atome berücksichtigt werden. Die passiven Atome sind zwar nicht stark gewichtet, wegen ihrer großen Zahl haben sie aber trotzdem die Chance, in der letzten Iteration die Zentren noch etwas zu verschieben. Wenn die Anzahl aktiver Atome nicht zu klein gewählt wird, sind die Klassen schon so eingeteilt, daß es zu keinen chaotischen Verschiebungen mehr kommt.
Als weitere Vereinfachung wurde die euklidische Distanz als Abstandsmaß verwendet, um die ständige Logarithmierung von Fließkommazahlen, wie sie zuhauf bei der Entropie-Distanz vorkommt, zu vermeiden. Diese Vereinfachungen bringen gewiß einen Verlust an Genauigkeit, haben dafür aber zu einer deutlichen Beschleunigung des Verfahrens geführt

In der 4. Iteration des Algorithmus wurden also nur die 1000 häufigsten Wörter als aktive Wörter für das Clustern ausgewählt.
Bei jeder Iteration wurde die Anzahl neu zugelassener Atome auf das doppelte des alten Wertes gesetzt, so daß bei einer Ursprünglichen Menge von 50 in diesem letzten Schritt 800 neue erzeugt wurden.

Eine Reduktion der entstehenden Zustandsgraphen wurde vorerst nur für sehr einfache, durch zeitliche Splits bedingte Verdoppelungen implementiert: Falls ein Zustand im Graphen nur einen Nachfolger besitzt, und beide durch das selbe Atom modelliert werden, werden beide Zustände fusioniert. Natürlich sind an dieser Stelle auch andere, beliebig komplexe Reduktionsalgorithmen anwendbar, auch um räumlich bedingte Verdoppelungen zu eliminieren. Ein Beispiel, das einen nicht ganz trivialen Fall räumlicher Verdoppelung und seine gewünschte Behebung zeigt, wird hier graphisch dargestellt. (Abb. 4.1)

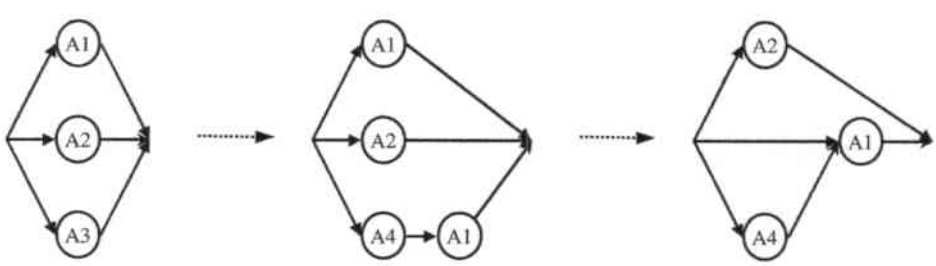

Abbildung 4.1: Zustandsverdoppelung und Graphenreduktion.

Bei größeren, komplexeren Graphen ist die Lösung aber nicht mehr so leicht zu bestimmen. Leider würde die Entwicklung eines kompletten Reduktionsverfahrens, das auch die meisten Fälle zufriedenstellend löst den Rahmen dieser Studienarbeit sprengen. Die hier erzeugten Zustandsgraphen sind deshalb häufig um einiges größer als nötig.

Für den Test der Erkennungsleistung wurde zunächst mit dem System S0 für jede Testäußerung ein Worthypothesegraph erzeugt. Auf diesem wurde dann eine Viterbi-Suche durchgeführt und der wahrscheinlichste Pfad als Hypothese ausgegeben. Um besser die Leistung des Algorithmus zu messen, der hauptsächlich auf die Verfeinerung der Akustik zielt, wurde auf den Einsatz eines Sprachmodells verzichtet. Die Verwendung eines Sprachmodells verbessert zwar stark die Erkennungsrate, macht es aber schwieriger, die Leistung der Akustik einzuschätzen.

Bei diesem Test erzielte das System S0 eine Worterkennungsrate von 42%, während S1 weniger als 1% erreichte. Offensichtlich wurde die Anzahl zu splittender Wörter zu niedrig gewählt, so daß kein gutes Training der Modelle möglich war. Darüber hinaus wurde festgestellt, daß die Zustandsgraphen ohne ein effektives Reduktionsverfahren bei manchen Wörtern zu stark anwachsen. Pro Iteration wurden durchschnittlich etwa doppelt so viele zeitliche wie räumliche Splits und so gut wie keine Reduktionen vorgenommen. In der dritten Iteration waren es 316 zeitliche, 98 räumliche Splits und 5 Reduktionen. In der letzten Iteration, 397 zeitliche und 299 räumliche Splits, während die Anzahl Reduktionen sprunghaft auf 53 anstieg. Dies deutet darauf hin, daß die Menge verfügbarer Atome die Anzahl der möglichen Splits für die neu hinzugekommenen Wörter beschränkt hielt. Obwohl die mittlere Anzahl Zustände pro Wort somit relativ klein blieb (12 in der 4. und 7 in der 5. Iteration), erhielten manche Wörter aber auffällig große Zustandsgraphen. So z.B. das Wort "Donnerstag", das letztendlich mit 75 Zuständen modelliert wurde. Um solch Große Graphen zu vermeiden muß eine zusätzliche Beschränkung der Zahl der Zustände pro Wort eingeführt werden.

Versuch 2

Für den zweiten Versuch (System S2) wurde die Zahl der Zustände eines Graphen auf 32 beschränkt. Sehr lange, selten vorkommende Wörter werden somit weniger genau modelliert, dafür können sich Graphen von kleinen, häufig auftretenden Wörtern sehr stark verfeinern. Die Menge der Wörter, für die Splits erzeugt werden, wurde auf 1000 erhöht, und die Anzahl neuer Atome schon ab der ersten Iteration auf 1000 festgesetzt. Die deutlich größere Zahl Modelle sollte eine Verbesserung der Erkennungsleistung ermöglichen. So wurden 4 Iterationsschritte durchgeführt. In der 5. und letzten Iteration wurden wieder alle 13486 Wörter des Trainingsvokabulars bei Splits berücksichtigt. Beim Clustern aber blieben weiterhin nur 1000 Wörter aktiv. Abbildung 4.2 zeigt die so erzielten Ergebnisse.

Die Erkennungsleistung steigt zu Beginn rasch an und erreicht in der 3. Iteration für

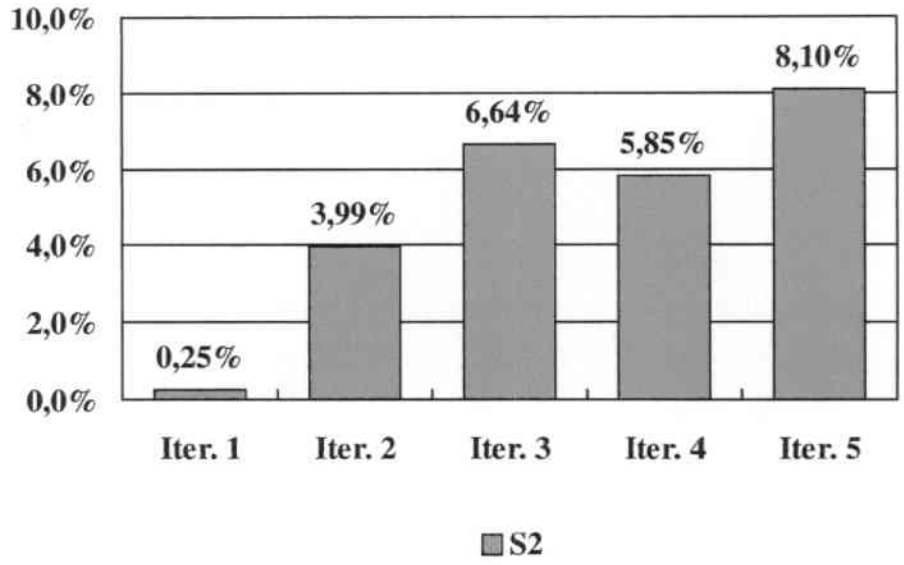

Abbildung 4.2: S2, Iterationen 1 bis 5

1000 Wörter schon ein Maximum. Der Leistungseinbruch in der 4. Iteration deutet darauf hin, daß die zur Verfügung stehende Atommenge nicht ausreicht, um die Zustandsgraphen noch signifikant zu verbessern. Dennoch bringt die Einführung eigener Zustandsgraphen für die restlichen 12468 Wörter im letzten Schritt wie erwartet noch eine leichte Steigerung.
Um die Erkennungsrate weiterhin zu verbessern wird aber eine weitere Erhöhung der Zahl der Atome nötig sein.

Versuch 3

Es wurde also ein weiteres System (S3) trainiert. Diesmal wurden die 3000 häufigsten Wörter bei Splits berücksichtigt und die Anzahl neuer Atome ebenfalls auf 3000 festgesetzt. Diese Vergrößerung brachte eine deutliche Verbesserung des Systems, das schon bei der ersten Iteration 6,06% und bei der zweiten 14,67% Wortakkuratheit erzielte. Allerdings sank die Erkennungsleistung bei der dritten Iteration wieder auf 8,91%. Ähnlich wie in Versuch 2 schien ein Maximum schon erreicht zu sein. Allerdings war noch nicht klar, ob diese Grenze durch die Menge an Trainingsdaten, durch die Anzahl Modelle des Systems oder durch andere Faktoren bedingt war. Deshalb wurde noch ein zusätzlicher 4. Iterationsschritt durchgeführt, immer noch mit 3000 zu splittenden Wörtern, aber mit 6000 neuen Atomen. Wie Abbildung 4.3 zeigt, war wieder eine Verbesserung der Leistung auf 11,59% möglich.

Allerdings wurde trotz der doppelten Anzahl Atome die Erkennungsleistung, die in der 2. Iteration erzielt wurde, nicht mehr erreicht. Dies zeigt, daß das System allein durch Erhöhung der Zahl der Atome nur bis zu einem gewissen Grad verbessert werden kann. Im Endeffekt sind die hier trainierten und geclusterten akustischen Atome nur

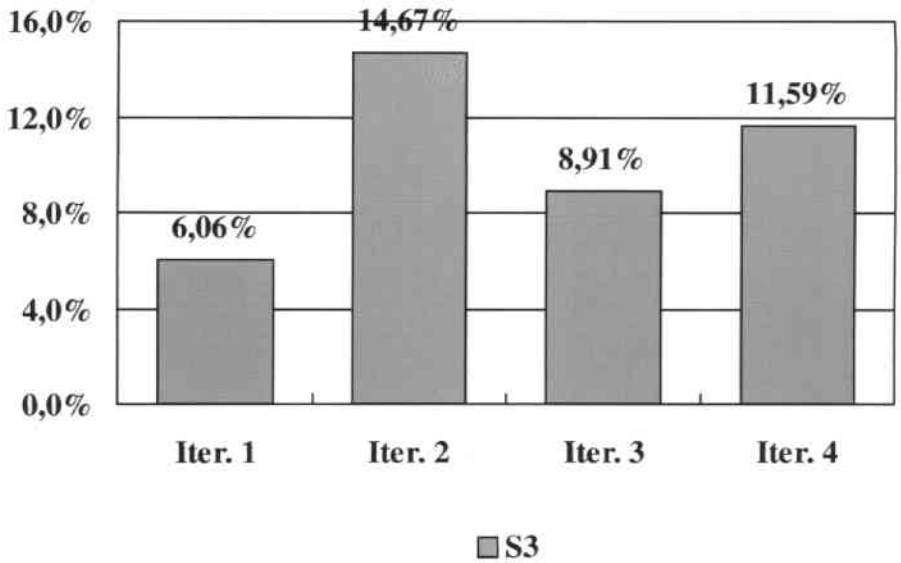

Abbildung 4.3: S3, Iterationen 1 bis 4

Mixturgewichte für die 256 Verteilungen eines einzigen Codebuchs. Bevor die Zahl der Mixturgewichte noch weiter erhöht wird, müßte auch das Codebuch vergrößert werden.

4.4 Zusammenfassung

Der größte Anteil von Fehlern in den erzeugten Hypothesen ist auf überflüssiges Einfügen von Wörtern zurückzuführen. Im Schnitt erzeugten die bisher vorgestellten Systeme Hypothesen, die bis zu 40% größer als die Transkriptionen waren. Dies vermindert sehr stark die Worterkennungsrate. Um diesen Effekt zu reduzieren, wurde bei Erkennungsläufen ein einfacher Strafterm für die Anzahl Wörter in einem Satz eingeführt. Abbildung 4.4 zeigt noch mal die so erzielte Leistung für S0, S2 und S3. Obwohl die Erkennungsleistung der Systeme S2 und S3 deutlich unter der des kontextabhängigen Systems S0 blieb, zeigen die Ergebnisse der einzelnen Iterationen, daß der Aufbau eines Spracherkennungssystems durch automatisches Splitten, Clustern und Erzeugen von Zustandsgraphen prinzipiell möglich ist. Um eine bessere Erkennung zu erreichen, müßten aber noch einige Grundlegende Schritte unternommen werden:

- Das Codebuch müßte vergrößert werden. Die Genauigkeit der akustischen Modelle hängt in großem Maße von den Gaußverteilungen des Codebuchs ab. Wenn zu wenige Referenzvektoren im Codebuch enthalten sind wird eine Erhöhung der Mixturgewichte keine nennenswerte Verbesserung bringen. Hier wurde eine kleine Zahl Gaußverteilungen verwendet um Entwicklungszeiten klein zu halten.

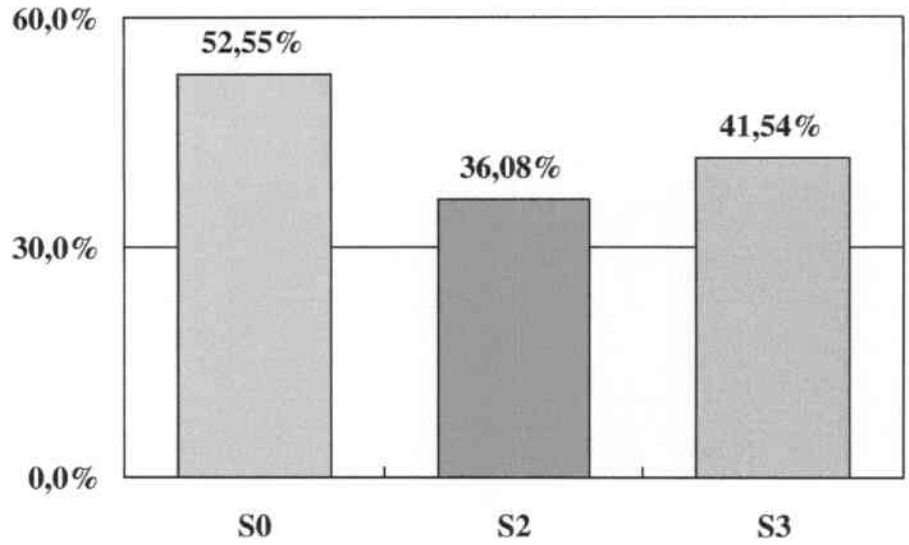

Abbildung 4.4: S0, S2, S3, mit Strafterm

Auch wurde aus demselben Grund darauf verzichtet, das Codebuch mitzutrainieren. Eine Erhöhung der Zahl der Codebücher oder gar die Verwendung der Codebücher zur Repräsentation akustischer Atome würde ohne Zweifel die Erkennungsleistung stark erhöhen.

- Ein effizienter Algorithmus zur Reduktion von Zustandsgraphen wäre nötig, um die gelegentlich erzeugte Redundanz in den Wortgraphen zu eliminieren. Das hier verwendete Verfahren zur Fusionierung verdoppelter Zustände kann nur sehr spezielle Fälle von Verdoppelungen beseitigen. So werden die Graphen häufig größer als nötig. Ein komplexes Verfahren könnte sogar in der Lage sein, Gruppierungen innerhalb verschiedener Wortgraphen zu erkennen, und somit gemeinsame Wortteile als selbstständige Graphen zu behandeln.

- Es müßte eine größere Anzahl Iterationen durchgeführt werden, um herauszufinden, wann die Zustandsgraphen und die Atommenge sich definitiv stabilisiert haben. Durch das ständige Clustern der Atome und reorganisieren der Graphen können diese sich noch lange Zeit optimieren, selbst wenn die maximale Anzahl Atome oder die maximale Graphengröße erreicht wurde. Interessant wäre auch zu sehen, wie flexibel ein solches System bei der Sprecheradaptation wäre, wenn also keine feste Zahl Iterationen vorgegeben ist, sondern das System kontinuierlich seine Parameter an neue Sprecherdaten anpassen darf.

Kapitel 5

Ausblick

In der hier vorgestellten Arbeit konnte nur auf einen Teil der Möglichkeiten eingegangen werden, die sich bei der automatischen Erzeugung von akustischen Einheiten ergeben.

Zum einen wäre es interessant zu untersuchen, wie stark sich das automatisch auf akustischer Information erzeugte Lexikon von den linguistisch inspirierten in seiner Grobstruktur unterscheidet. Um die generierten Atome wieder lesbar zu machen könnte ihre Distanz zu Standard-Phonemen ermittelt und jedes Atom durch das Phonem umschrieben werden, das ihm am ähnlichsten ist. Dann könnten aus den Zustandsgraphen Lexikoneinträge erzeugt werden, indem jeder Pfad durch den Graphen als Aussprechvariante genommen wird. Das Ergebnis wäre wieder nach ein paar elementaren Vereinfachungen in menschlich lesbarer Form.

Eine Abwandlung des Verfahrens könnte womöglich auch für die Erkennung periodischer Geräusche, wie z.B. Rotorlärm, Lachen oder Brummen eingesetzt werden. Zusätzlich zu den vorgestellten zeitlichen und räumlichen Splits könnte eine dritte Form, eine Variante des zeitlichen Splits, zugelassen werden, bei der die ursprüngliche reflexive Verbindung erhalten bleibt, in Form einer Rückverbindungen vom letzten zum ersten Zustand. Auf Dauer entstünde ein zyklischer Zustandsgraph, der besser geeignet wäre, Repetitionen, wie sie z.B. in lautem Lachen vorkommen, zu modellieren, oder gar um Stottern oder Neuanfänge zu erkennen. Außerdem wäre ein Zustandsgraph mit Zyklen wahrscheinlich gut geeignet, für die Analyse von Musikstücken, in denen typischerweise eine ganze Menge Periodizität vorhanden ist. Da der Algorithmus selbstständig Splits mit oder ohne Zyklen auswählt, könnten selbst für unregelmäßige Rhythmen passende Zustandsgraphen erzeugt werden. Für die unterschiedlichsten Bässe z.B. könnten so passende Modellierungen gefunden werden.

Ein weiterer Einsatzbereich wäre das Training ohne Transkriptionen. Der Erkenner soll in der Lage sein, selbständig auf einem Strom von Sprachdaten zu trainieren und so gewissermaßen unüberwacht neue Wörter lernen. Um dies zu erreichen wird ein Konfidenzmaß benutzt, um dem System zu signalisieren, an welcher Stelle wahrscheinlich ein neues Wort aufgetaucht ist. Mit einer Variante des hier vorgestellten

Algorithmus könnte das System für dieses einen neuen Zustandsgraphen generieren: Ausgehend von einem einzigen Zustand werden ein paar schnelle Iterationen durchgeführt, bei denen auf den eben erhaltenen Mustern trainiert wird, bis weitere Splits keine nennenswerte Veränderung mehr bringen. Zur Modellierung der Zustände können bereits existierende Atome benutzt werden, oder es können leichte Änderungen an der Atommenge zugelassen sein. Wenn das Wort noch einmal vorkommt, mit ungefähr derselben Aussprache, besteht die Chance, das es von dem neuen Graphen mit genügend hohem Konfidenzgrad als solches erkannt wird. Dieser kann dann mit den neuen Daten weiter trainiert werden. Um herauszufinden, um welches Wort es sich nun letztendlich handelt, z.B. bevor eine Konversion in Textform für einen menschlichen Leser erzeugt wird, ist nur noch ein letzter Schritt nötig. Wenn die Zustandsgraphen wie oben angedeutet in lesbare Form konvertiert werden, könnte diese benutzt werden, z.B. um in einem Wörterbuch nach den ähnlichsten Wörtern zu suchen, und die passendste Hypothese auszugeben.

Literaturverzeichnis

[Lee90] Kai-Fu Lee
Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition;
in Readings in Speech Recognition, Edited by Alex Waibel and Kai-Fu Lee
Morgan Kaufmann, San Mateo, California, 1990, pp 374 ff.

[Schultz94] Tanja Schultz
Akustische Modellierung Sprachlicher und Nichtsprachlicher Geräusche, Studienarbeit.
Institut für Logik, Komplexität und Deduktionssysteme; Universität Karlsruhe, 1994.

[Takami-Sagay] Jun-ichi Takami and Shigeki Sagayama
A Successive State Splitting Algorithm for Efficient Allophone Modelling
ATR Interpreting Telephony Research Laboratories, Kyoto, Japan.

[Singer-Ost] Harald Singer, Mari Ostendorf
Maximum Likelihood Successive State Splitting.
ATR Interpreting Telephony Research Laboratories, Kyoto, Japan.