Konstanz, Siemens Dematic AG Postal Automation

Automatische Modellierung gebundener Handschrift in einem HMM-basierten Erkennungssystem

Dissertation zur Erlangung des Doktorgrades Dr. rer. nat.

der Fakultät für Informatik der Universität Ulm

von

Marc-Peter Schambach

aus Konstanz

2004

Amtierender Dekan: Prof. Dr. Friedrich W. von Henke

Gutachter: Prof. Dr. Günther Palm

Gutachter: Prof. Dr. Heiko Neumann

Gutachter: Prof. Dr.-Ing. habil. Gerhard Rigoll

Tag der Promotion: 7. Mai 2004

Übersicht

Um Schrift gut erkennen zu können, benötigt man das richtige Modell. Dies wird besonders deutlich bei „fremden“ Schriften, zum Beispiel Arabisch. Zum Aufbau eines Systems zur Erkennung arabischer Schrift benötigt man bisher zumindest grundlegendes Wissen über die Struktur dieser Schrift.

Gängige Systeme zur automatischen Schrifterkennung verwenden Modelle zur Repräsentation der Buchstaben. Das Aussehen der Zeichen wird in den freien Parametern der Modelle repräsentiert. Viele Arbeiten zur Schrifterkennung beschäftigen sich damit, mit welchen Methoden diese Parameter am besten zu bestimmen sind. Die *Grundstruktur der Schrift* – die Anzahl relevanter Schreibvarianten der einzelnen Zeichen, ihre Größe und Komplexität – wird hingegen meist fest vorgegeben. Die Modellierung wird üblicherweise von subjektiven Annahmen bestimmt, zum Beispiel werden Kategorien wie „Handschrift“ und „Maschinenschrift“ gebildet. Häufig wird für alle Buchstaben pauschal dieselbe Modellstruktur vorgegeben. Es ist nicht garantiert, dass die so gewählten Modelle für die Erkennung optimal sind. In dieser Arbeit wird ein neues System vorgestellt, das versucht, für gegebene Schriften das richtige Modell automatisch zu finden.

Grundlage und Ausgangspunkt der Arbeit ist *Powerscript*, ein bewährtes Erkennungssystem für gebundene Handschrift von Siemens Dematic, das erfolgreich in der Postautomatisierung eingesetzt wird. Es repräsentiert Buchstaben durch Hidden-Markov-Modelle (HMMs). Die Schriftstruktur ist in der Topologie der Buchstaben-HMMs angelegt. Verschiedene Schreibweisen der Buchstaben werden durch parallele, lineare Pfade aus Modellzuständen repräsentiert, deren Länge der Komplexität der Zeichen entspricht. *Anzahl* und individuelle *Länge* der Zustandspfade wurden bisher fest vorgegeben und meistens für alle Zeichen gleich gewählt. Gegenstand dieser Arbeit ist es, diese Parameter automatisch zu berechnen: Gebundene Handschrift wird über die Optimierung der HMM-Topologien modelliert.

Es stellt sich die Frage, nach welchen Kriterien die Topologie optimiert werden soll. Das Prinzip zur Wahl des „richtigen“ Modells ist als *Occam's Razor* bekannt: Es muss ein geeigneter Kompromiss zwischen notwendiger Komplexität und möglichst großer Einfachheit gefunden werden. Zur Umsetzung wird ein Bayes'scher Ansatz verwendet, aus dem anhand von Näherungen verschiedene Modellwahlkriterien hergeleitet werden. Solche Kriterien werden in Zusammenhang mit HMMs üblicherweise nur auf einfache Modelle angewendet, in dieser Arbeit werden sie aber an das komplexe System der Buchstaben-HMMs angepasst. Ihre Relevanz wird anhand der Erkennungsleistung untersucht und bestätigt. Neben dieser globalen

Bewertung des vollständigen Schriftmodells werden weitere Maße für die lokale Bewertung und den Vergleich einzelner Buchstaben-HMMs entwickelt. Auch hier lässt sich über den Umweg der Erkennung die praktische Bedeutung dieser Maße zeigen.

Zur Optimierung von HMM-Topologien wurden bereits verschiedene Verfahren vorgeschlagen. Im konkreten Fall liegen durch die Verteilung des Schriftmodells auf einzelne Buchstaben-HMMs und deren vorgegebene Grundstruktur aber besondere Bedingungen vor, die spezielle Verfahren zur Strukturoptimierung notwendig machen. In den hier vorgestellten neuen Verfahren werden einzelne Aspekte der Schriftmodellierung gezielt verbessert. Zwei verschiedene Algorithmen werden spezifiziert, die voneinander unabhängig die jeweilige Länge bzw. die Anzahl der Zustandspfade optimieren. Diese Trennung erlaubt eine separate Analyse des Optimierungspotentials der Methoden. Beide Verfahren arbeiten iterativ und lassen sich durch sequentielle Ausführung einfach zu einem Gesamtsystem kombinieren.

Die neuen Verfahren zur Topologiebestimmung werden durch ein vollständiges Adaptions- und Trainingssystem realisiert, das nur durch die Vorgabe von Alphabet und Trainingsdaten (bestehend aus Wortbildern und Bedeutungen) automatisch eine funktionsfähige Konfiguration des Erkennungssystems generiert. Es baut auf dem Parametertraining des Ausgangssystems *Powerscript* für vorgegebene Modellstrukturen auf, passt bei der Bestimmung der Modellparameter aber zusätzlich – und transparent für den Anwender – die Modelltopologie an. Dazu werden neue Datenstrukturen und Methoden verwendet, mit denen flexible Modellstrukturen generiert, verändert und bewertet werden können. Auf ihnen beruht eine vollständige Neuimplementierung des HMM-Parametertrainings, die die Grundlage für die Realisierung der neuen Verfahren darstellt. Es existieren zahlreiche Tools für die automatische Durchführung der Modellbestimmung, die Visualisierung der generierten Modelle und die Auswertung der erzielten Erkennungsleistung. Die leichte Austauschbarkeit von Algorithmen erlaubt die schnelle Durchführung von Experimenten.

Die entwickelten und implementierten Verfahren sind geeignet, ein zur Erkennung benötigtes gutes Modell zu finden: Die Ergebnisse der Modellierung sind plausibel. Visualisierungen der generierten Schriftmodelle sind intuitiv erklärbar und zeigen die erwarteten gängigen Schreibweisen. Die Leistung des Erkennungssystems verbessert sich durch die Optimierung des Schriftmodells. Im Durchschnitt steigt die Erkennungsrate in den untersuchten Projekten im Vergleich zu *Powerscript* um 8,1 Prozent. Auch bei konstanter Gesamtkomplexität des Schriftmodells – also bei gleichbleibendem Rechenaufwand für die Erkennung – werden deutliche Verbesserungen erzielt. Dadurch lässt sich das System in der Praxis verwenden und wird bereits in ersten Pilotanwendungen in Anlagen der Postautomatisierung eingesetzt.

Insbesondere wird der Entwicklungsaufwand für neue Erkennungssysteme beträchtlich gesenkt. Zum Beispiel kann jetzt auch ohne die Vorgabe von Detailwissen über die Schreibweisen arabischer Schrift ein leistungsfähiges Erkennungssystem automatisch generiert werden.

Inhaltsverzeichnis

Abkürzungen

AIC	Akaike Informations-Kriterium
BCC	Binary Connected Components, Zusammenhangsgebiet
BIC	Bayes'sches Informations-Kriterium
BMBF	Bundesministerium für Bildung und Forschung
CEDAR	Center of Excellence for Document Analysis and Recognition in Buffalo, NY
CS	Cheeseman-Stutz
DTW	Dynamic Time Warping
EM	Expectation-Maximization, iteratives Optimierungsverfahren
HMM	Hidden-Markov-Modell
HMR	Hidden Markov Recognizer, Worterkennungssystem bei Siemens Dematic
KL	Kullback-Leibler
LBG	Linde-Buzo-Gray, Clusteralgorithmus
MAP	Maximum a posteriori
ML	Maximum Likelihood
MMD	Maximum Model Distance
SIMD	Single Instruction Multiple Data
SDTW	Statistical Dynamic Time Warping
ZIP	Zone Improvement Plan; ZIP-Codes sind US-amerikanische „Postleitzahlen“

Verwendete Symbole

α	Parameter der Dirichlet-Verteilung
$\alpha_t(\cdot)$	Vorwärtswahrscheinlichkeit beim Forward-Backward-Algorithmus
$\beta_t(\cdot)$	Rückwärtswahrscheinlichkeit beim Forward-Backward-Algorithmus
$\delta_t(\cdot)$	Wahrscheinlichkeit für besten Teilpfad bei Viterbi-Berechnung
$\Gamma(\cdot)$	Gamma-Funktion
$\gamma_t(i)$	Aufenthaltswahrscheinlichkeit zum Zeitpunkt t in Zustand s_i
λ	HMM-Parameter
$\mathcal{N}(v)$	Normalverteilungsfunktion
μ	Mittelwertvektor
Φ	Transformation des Merkmalsraums
π_i	Eingangswahrscheinlichkeit in Zustand s_i
$\psi_t(\cdot)$	Bester Vorgängerzustand bei Viterbi-Rückverfolgung
θ	Modellparameter
$\varphi_k(O_t)$	Symbolbeobachtungswahrscheinlichkeit bei Fuzzy-Vektorquantisierung
$\xi_t(\cdot,\cdot)$	Sequenzwahrscheinlichkeit für zwei Zustände
A	Hessematrix
a_i	Worterkennungsalternative an Position i
a_{ij}	Übergangswahrscheinlichkeit von Zustand s_i nach s_j
$B(\cdot)$	Beta-Funktion
b_i	Emissionswahrscheinlichkeit aus Zustand s_i
C	Codebuch

c_i	Auftretenshäufigkeit einer Beobachtung oder eines Ereignisses i
d	Anzahl Modellparameter, Dimension von θ
$D(\cdot,\cdot)$	Distanz, Divergenz
$E\{\cdot\}$	Erwartungswert
$G(M_s)$	Gütekriterium für eine Modellstruktur
i, j	Zustandsindizes
k	Symbol- oder Klassenindex
K	Kovarianzmatrix
l_i	Länge eines Zustands s_i (entspricht der durchschnittlichen Anzahl Beobachtungen)
M	Modellierung
M_s	Modellstruktur
n	Anzahl möglicher Beobachtungen oder Ereignisse
N	Anzahl der HMM-Zustände
O	Beobachtungsfolge
$O(\cdot)$	Ordnung, Komplexität eines Algorithmus
$P(\cdot)$	Wahrscheinlichkeit
$p(\cdot)$	logarithmierte Wahrscheinlichkeit, $p(\cdot) = \ln P(\cdot)$
Q	Menge der Zustände $\{s_i\}$ eines HMM
q_t	Postulierter Zustand des HMM
S	Streuungsmatrix
s_i	Zustand eines HMM
T	Länge der Beobachtungsfolge, Anzahl der Trainingsdaten
V	Beobachtungsraum
v_t	Beobachtungs-, Merkmalsvektor
X	Trainingsdaten, Messungen
y_k	diskretes Ausgabesymbol

Kapitel 1

Einleitung

Gebundene Handschrift zeichnet sich durch sehr hohe Variabilität aus. Abbildung 1.1 zeigt anhand verschiedener Schreibweisen desselben Wortes das riesige Spektrum möglicher Schriftvariationen. Auf den Inhalt von Geschriebenem kann häufig nur aus dem Zusammenhang geschlossen werden (im Beispiel aus dem Kontext „kanadische Adresse“). Für den Laien nicht zu entziffernde Arztrezepte können als weiteres typisches Beispiel für unleserliche Handschrift angeführt werden. Die Freiheit beim Schreiben ist so groß, dass es neben der Übermittlung des eigentlichen Informationsinhalts sogar möglich ist, Geschriebenem eine persönliche Ausdrucksnote zu geben. [1]

Man geht zwar davon aus, dass es ein vollständiges Modell der Schrift gibt, das grundlegende, in der Schule gelehrte Regeln für die *Bildung* von Handschrift spezifiziert; diese Regeln werden von Schreibern aber typischerweise so weit ausgelegt, dass eine schriftliche Verständigung *gerade noch* möglich ist. Die Schwierigkeit beim Entziffern unbekannter Schriften scheint nahezulegen, dass einfache Schreibregeln nur ein unzureichendes Modell für das Schreiben und Wiedererkennen von Handschrift darstellen.

1.1 Entwicklung eines Schriftmodells

Es stellt sich die Frage, welche Merkmale für das *Erkennen* von Geschriebenem entscheidend sind. Es geht darum, ein Modell der fertig produzierten Schrift, nicht des Schreibens zu entwickeln. Welches sind die wesentlichen Eigenschaften der verschiedenen Zeichen? Da immer eine eindeutige Zuordnung von Geschriebenem und Inhalt besteht, lässt sich ein Modell anhand existierender Schriftmuster direkt überprüfen. Handschrift stellt dadurch ein schönes und gut zu analysierendes Beispiel für die Modellierung abstrakter Begriffe oder Muster dar.

Systeme zur automatischen Erkennung von Handschrift bilden die natürliche Basis zur Überprüfung von Schriftmodellen. Ihre Entwicklung geht mit der Bildung eines Modells der

[1] Variationen des Schriftstils, etwas Fett- oder Kursivschrift, werden auch in gedruckten Dokumenten zur Übermittlung von Informationen genutzt.

Abbildung 1.1: Verschiedene Schreibweisen desselben Wortes („Toronto“) als Beispiele für schwer lesbare Handschrift. Die erlaubten Variationen der Schrift orientieren sich mehr an der Wahrnehmung als an vorgegebenen Konstruktionsregeln. Zum Teil ist der Inhalt nur aus dem Kontext (hier: „kanadische Adresse“) zu erschließen.

Schrift einher, und über die Erkennungsleistung kann direkt auf die Güte des Modells geschlossen werden. Ein Erkennungssystem ermöglicht also die Suche nach einem sinnvollen, „richtigen“ Modell der Schrift. Die eher theoretische Frage nach den wesentlichen Merkmalen von Handschrift lässt sich also über den Umweg der praktischen Anwendung beantworten: Merkmale, die einen positiven Effekt auf die Erkennung ausüben, sind in diesem Sinne wesentlich, solche, die keinen Effekt haben, sind unwesentlich.

1.2 Optimierung der HMM-Topologie

Diese Arbeit beruht auf einem Handschrifterkennungssystem, das Schrift mit *Hidden-Markov-Modellen* (HMMs) modelliert. HMMs sind ein stochastischer Prozess, durch den Folgen von Beobachtungen – in diesem Fall die Form des Schriftzuges von links nach rechts – leicht beschrieben werden können. Sie werden in Kapitel 3 ausführlich behandelt. Die variable Struktur der HMMs erlaubt eine flexible Modellierung der Schrift. Jeder Buchstabe wird durch ein separates HMM repräsentiert. Die Struktur der Modelle definiert unter anderem die Anzahl verschiedener Schreibvarianten eines Zeichens, sowie deren jeweilige Größe und Komplexität. In Kapitel 5 werden die Buchstabenmodelle detailliert beschrieben.

Die Struktur der HMMs wird auch als Modelltopologie bezeichnet. Über sie fließt explizites Wissen über die Schrift in das System ein. So können zum Beispiel Groß- und Kleinschreibung unterschieden und als mögliche Varianten in einer statischen Modellstruktur festgelegt werden. Die Topologie der HMMs bestimmt damit weitgehend die Modellierung der Schrift. Im Kontext dieser Arbeit entspricht deshalb die Suche nach der besten Modellierung der Optimierung der HMM-Topologie.

Der entscheidende Schritt in dieser Arbeit besteht darin, die Topologie der HMMs bei der Belehrung des Systems als zusätzliche freie Parameter zu gestalten, und Verfahren zu seiner

automatischen Bestimmung zu entwickeln. Aus der Topologie des belehrten Systems können dann neben den erwarteten Leistungsverbesserungen auch Aussagen über Eigenschaften der Schrift gemacht werden.

1.3 Ziele der automatischen Modellierung

Mit der automatischen Schriftmodellierung über die Bestimmung der HMM-Topologie werden mehrere Ziele verfolgt, die sich gegenseitig ergänzen. Da die Hauptanwendung des behandelten Erkennungssystems im Gebiet der Postautomatisierung liegt, werden die Ziele anhand dieser speziellen Anwendung beleuchtet.

1.3.1 Analyse unbekannter Schriften

Der Schwerpunkt der Arbeiten liegt auf der vollautomatischen Bildung von Schriftmodellen. Dies ist besonders wichtig, wenn Erkennungssysteme für Schriften entwickelt werden sollen, für die keine Erfahrung über Schreibvarianten oder Zeichenkomplexität verfügbar ist. Dieser Fall tritt häufig bei der Entwicklung von Systemen für die Postautomatisierung auf, wenn Erkennungssysteme für fremde Schriftarten, zum Beispiel Griechisch, Kyrillisch oder Arabisch, entwickelt werden. In solchen Fällen ist es hilfreich, eine automatische Analyse der Schreibweisen vornehmen zu können, die bei der Bestimmung der Modellstruktur durchgeführt wird. In der Tat erlaubt ein solches System das Lernen von Schriften, deren Eigenschaften der Entwickler im Detail gar nicht kennt.

1.3.2 Senkung des Entwicklungsaufwands

Auch bei lateinischer Schrift gibt es große Unterschiede in den Schreibweisen in verschiedenen Ländern. Dies wird in Projekten der Postautomatisierung deutlich, in denen die Erkennungssysteme speziell für einzelne Länder angepasst werden müssen, was mit hohen Entwicklungskosten verbunden ist. Wenn die Systeme so konstruiert sind, dass sie in der Lage sind, sich an die unterschiedlichen Gegebenheiten automatisch zu adaptieren, können projektbezogene Entwicklungsaufwände gesenkt werden. Dies wird durch eine automatische Modellierung erreicht.

1.3.3 Steigerung der Erkennungsleistung

Das Ziel jeder Entwicklung im Bereich von Erkennungssystemen ist eine hohe Erkennungsleistung. Sie kann durch eine gute Modellierung der Schrift erreicht werden. Wird das Kriterium zur Bestimmung einer optimalen Modellstruktur direkt so definiert, dass die Erkennungsleistung des Systems maximiert wird, so erhält man durch die automatische Modellierung in jedem Fall ein mindestens vergleichbares oder besseres Erkennungssystem.

1.4 Aufbau der Arbeit

In ersten Teil dieser Arbeit wird zunächst das verwendete System zur Handschrifterkennung vollständig beschrieben. Kapitel 2 gibt eine kurze allgemeine Einführung in die Methoden der Handschrifterkennung. Kapitel 3 erklärt dann zunächst die mathematischen Grundlagen von Hidden-Markov-Modellen, bevor in den Kapiteln 4 bis 6 die konkreten Schritte der Bildverarbeitung und Merkmalsgenerierung, des Modelltrainings und der Erkennung detailliert dargestellt werden. Ein Überblick in Kapitel 7 über die Arbeiten an diesem System, das die Basis für die weiteren Überlegungen darstellt, schließt den ersten Teil ab.

Im zweiten Teil werden dann Methoden entwickelt, mit denen das in Teil I noch fest vorgegebene Schriftmodell automatisch bestimmt wird. Dazu werden in Kapitel 9 zunächst allgemeine Kriterien für die statistische Modellierung hergeleitet. In Kapitel 10 werden dann bekannte Verfahren zur HMM-Topologieoptimierung vorgestellt und auf die spezielle Situation der Schriftmodellierung mit HMMs eingegangen. Es werden zwei neue, unabhängige Verfahren zur Bestimmung der Topologie der Buchstaben-HMMs entwickelt: Das eine optimiert in Kapitel 11 die Modellierung der einzelnen Zeichen in ihren vorgegebenen Schreibvarianten, das andere bestimmt in Kapitel 13 die beste Anzahl dieser Varianten. Für das zweite Verfahren müssen in Kapitel 12 zunächst noch konkrete Bewertungsmaße für Buchstaben-HMMs entwickelt werden. Die Kombination beider Verfahren bestimmt abschließend das Schriftmodell. Die im vorigen Abschnitt genannten Ziele der automatischen Modellierung können damit erreicht werden. Insbesondere wird eine deutliche Steigerung der Erkennungsleistung im Vergleich zum Basissystem aus Teil I erzielt.

Teil I

Erkennung gebundener Handschrift mit HMMs

Kapitel 2

Handschrifterkennung

Man unterscheidet bei der Handschrifterkennung zwischen *Online*- und *Offline*-Erkennung, je nachdem, ob die Eingabe der Schrift direkt während ihrer Erzeugung über ein Digitalisiertablett erfolgt, oder der fertige Schriftzug später von Papier über einen optischen Scanner erfasst wird. Bei der Online-Erkennung [Tap90, Dol98] hat man im Gegensatz zum Offline-Fall zusätzliche Informationen über die zeitliche Entwicklung des Schriftzuges, was die Erkennungsaufgabe erleichtert. Außerdem kann sich das System bei vielen Anwendungen an den Benutzer und dessen individuelle Schreibweise adaptieren. Dadurch können generell etwas höhere Erkennungsraten erzielt werden als bei der Offline-Erkennung, allerdings müssen Erkennungsraten bei typischen Online-Anwendungen sehr hoch sein, damit ein System von den Benutzern akzeptiert wird. Im Gegensatz dazu können sich Systeme bei typischen Anwendungen der Offline-Erkennung nicht an einzelne Benutzer adaptieren, es werden aber auch mit geringeren Erkennungsraten schon wirtschaftliche Ergebnisse erzielt.

Eine weitere Unterscheidung bei Handschrifterkennungssystemen betrifft Einschränkungen bezüglich der Arten von Schrift, die erkannt werden sollen. Bei westlichen Schriften trifft man typischerweise zwei verschiedene Schreibweisen an: Bei *Handblockschrift* werden zur Bildung ganzer Wörter separate Einzelzeichen aneinandergereiht, von *gebundener* oder *kursiver Handschrift* spricht man, wenn die Zeichen in einem Schriftzug miteinander verbunden sind. Daneben gibt es chinesische und andere ikonographische, nicht alphabetische Schriften, deren Struktur stärker zweidimensional angelegt ist.

In dieser Arbeit wird ausschließlich die Offline-Erkennung alphabetisch orientierter, gebundener Handschrift behandelt. Das System wird als *Scriptworterkenner* bezeichnet. Die Erkennung findet im Kontext ganzer Wörter oder Zeilen statt.

2.1 Methoden

Die gängigen Verfahren zur Erkennung von kursiver Handschrift unterscheiden sich durch die Art der Segmentierung in einzelne Zeichen: Es gibt *segmentierungsbasierte*, *segmentierungsfreie* (implizit segmentierende) und *holistische* Verfahren [Ste99]. Des weiteren kön-

nen die Methoden in *strukturelle* und *statistische* Ansätze eingeteilt werden.

2.1.1 Segmentierung

Klassische segmentierungsbasierte Erkennungsverfahren setzen auf einen zweistufigen Ansatz, bei dem in einem ersten Schritt eine Segmentierung in Einzelzeichen erfolgt, welche in einem zweiten Schritt dann erkannt werden. Bei kursiver Handschrift führt dieses Vorgehen nicht unbedingt zum Erfolg, da sich eine exakte Segmentierung als sehr schwierig erweist. Man steht vor dem Paradox der Mustererkennung, dass für eine korrekte Segmentierung erst der Inhalt erkannt werden muss, die Erkennung aber erst nach der Segmentierung durchgeführt werden kann. Auch Menschen können beim Lesen häufig nur dann eine Trennung der einzelnen Buchstaben vornehmen, wenn sie auch das ganze Wort erkennen. Die Segmentierung muss also auch bei technischen Lösungen implizit durchgeführt werden.

Für solche segmentierungsfreie (oder implizit segmentierende) Methoden gibt es verschiedene Ansätze. Es wird meist eine willkürliche Übersegmentierung mit der Bildung von Teilen, die kleiner oder gleich als ein Zeichen sind, durchgeführt, die in einem nachfolgenden Rekombinationsschritt weiterverarbeitet werden. Das in dieser Arbeit vorgestellte Handschrifterkennungssystem arbeitet mit einer festen Unterteilung in gleich große Sub-Zeichen-Bereiche des Wortbildes, die über die Emissionen der HMMs den verschiedenen Zeichen zugeordnet werden.

Etwas abseits von dieser Kategorisierung der Verfahren stehen holistische Ansätze, die nur bei kleinem Wortschatz Aussicht auf Erfolg haben. Sie versuchen, die menschliche Wahrnehmung nachzubilden, indem die Wörter als ganzes erkannt werden. Bestimmte Merkmale, die über das ganze Wort verteilt sein können, werden herangezogen, um die möglichen Wortalternativen zu bewerten. In gewisser Weise wird die Aufgabe der Worterkennung auf eine übliche Klassifikation abgebildet, die aber das Problem der Klassenbegrenzung hat.

2.1.2 Strukturelle und statistische Ansätze

Die Verfahren können weiter eingeteilt werden, indem man *strukturelle* und *statistische* Ansätze unterscheidet. Bei strukturellen Verfahren wird versucht, ausgehend von vorgegebenem Wissen über den Prozess des Schreibens oder die Struktur der Schrift, die für die entsprechenden Buchstaben charakteristischen Merkmale zu identifizieren. Diese werden explizit formuliert und mit den tatsächlichen Beobachtungen abgeglichen. Statistische Verfahren beruhen dagegen darauf, dass aufgrund einer Trainingsstichprobe die Eigenschaften der zu erkennenden Muster statistisch erfasst werden und daraus eine Klassifikation abgeleitet werden kann. Man kann sagen, der Rechner „lernt“ die Struktur implizit durch viele Beispiele.

Der Vorteil statistischer Verfahren besteht darin, dass allein durch die Hinzunahme weiteren Trainingsmaterials – und durch Zunahme der verfügbaren Parameter – eine größere Variabilität der Muster repräsentiert und gelernt werden kann. Solche Systeme sind „nach

oben offen“, während explizite Strukturbeschreibungen aufgrund der Begrenztheit durch den Menschen eine nicht mehr handhabbare Komplexität erreichen, wenn der Detaillierungsgrad der Beschreibung ein gewisses Maß überschreitet. Für die Ziele in dieser Arbeit sind allein statistische Verfahren geeignet, da – wie eingangs erwähnt – nicht vorgegebene strukturelle Bildungsregeln, sondern die tatsächlichen Erscheinungsformen der Schrift im Interesse stehen

2.2 Anwendungen

Dokumente treten heutzutage fast ausschließlich in gedruckter oder elektronischer Form auf. Texte sind in dieser Form leicht zu lesen, zu handhaben und weiterzuverarbeiten. Das Ziel des Schreibens ist deshalb auch fast immer die Erstellung gedruckter Dokumente. Es stellt sich die Frage, in welchen Bereichen Handschrift überhaupt noch eingesetzt wird, so dass eine automatische Erkennung sinnvoll ist.

Längere Texte werden sicherlich nur noch in Ausnahmefällen handschriftlich verfasst. Das Haupteinsatzgebiet von Handschrift liegt im spontanen Schreiben kurzer Texte, zum Beispiel um Notizen festzuhalten oder wenige Informationen schnell und komfortabel zu übermitteln. Handschrift hat dabei den Vorteil, dass außer einem Stift keine weiteren Hilfsmittel erforderlich sind. Häufig werden in solchen Fällen Formulare eingesetzt, damit die Informationen einfach und eindeutig übermittelt werden können. Gerade bei den dabei auftretenden speziellen Formaten stellt Handschrift eine praktische und komfortable Eingabemöglichkeit dar. Auch Taschencomputer, so genannte *Personal Digital Assistants*, zielen auf einfache Benutzung und sind ungewöhnlich im Format, weshalb Handschrift bei ihnen eine wichtige Option für die Bedienung darstellt.

Es ist davon auszugehen, dass Handschrift auch weiterhin eine wichtige Rolle für die Kommunikation spielen wird. Sie stellt eine mögliche, sehr einfach Schnittstelle für die Übermittlung von Informationen dar, die auch genutzt werden wird. Aus dieser Tatsache beziehen Systeme zur automatischen Erkennung von Handschrift ihre Existenzberechtigung.

2.2.1 Industrieller Einsatz

Zwei Bereiche, in denen Handschrift massenhaft auftritt, und die dadurch wirtschaftlich interessant sind, sind Scheckverarbeitung und Postautomatisierung. Sie stellen die Standardanwendungen für Handschrifterkennungssysteme dar. In beiden Fällen hat man zum einen ein hohes Einsparungspotential, zum anderen wird die Erkennungsaufgabe dadurch praktikabel, dass die Struktur der Wortbilder stark eingeschränkt ist.

Beim Schecklesen [Gui98] hat man ein vorgegebenes Formular, durch das die relevanten Felder leicht und mit großer Sicherheit gefunden werden können. Die Aufgabe des Scriptworterkenners besteht darin, die textuelle Beschreibung des Betrags, den *legal amount*, zu lesen, um damit den Betrag in Ziffern, den *courtesy amount*, zu verifizieren und so die Zuver-

lässigkeit der Erkennung zu steigern. Die Struktur der zu erkennenden Beträge ist bei dieser Anwendung auf eine überschaubare Anzahl Wörter beschränkt und kann durch eine einfache Grammatik beschrieben werden. Es ist jedoch zu beachten, dass bei der Art der Anwendung Fehlerkennungen sehr teuer sind und besonderer Maßnahmen zur Minimierung bedürfen.

In der Postautomatisierung [Sri93, Sri97] lassen sich bereits bei geringen Erkennungsleistungen hohe Einsparungen erzielen, da in jedem Fall manuelle Kodierarbeit eingespart werden kann. Beim Lesen der Adresse wird wieder die Redundanz der Daten ausgenutzt, um die Erkennungsergebnisse zu bestätigen. Die Postleitzahl wird mit dem Städtenamen korreliert, zusätzlich können auch für Anschrift und Straßennamen die Einträge in einer vollständigen Adressdatenbank ausgewertet werden. Auch für allgemeinere Anwendungen des Formularlesens gibt es im postalischen Umfeld einige Möglichkeiten. Beispiele sind die automatische Bearbeitung von Nachsendeaufträgen oder so genannte *business reply mail*, also die Verarbeitung von Warenbestellungen oder Informationsanforderungen auf vorgedruckten Postkarten.

Das hier vorgestellte Handschrifterkennungssystem wird für die Adresserkennung in Postverteilanlagen eingesetzt, ist aber auch in anderen Anwendungen einsetzbar.

2.2.2 Adresserkennung in der Postautomatisierung

Die Aufgabe eines Adresslesesystems besteht darin, ausgehend vom eingescannten Bild einer Postsendung einen Verteilcode zu bestimmen, der den Adressaten der Sendung eindeutig bestimmt. Die dazu notwendigen Berechnungen lassen sich vereinfacht in die folgenden einzelnen Teilschritte zerlegen:

- Suche des Adressblocks,
- Segmentierung des Adressbildes in Zeilen, Wörter und Einzelzeichen,
- Klassifikation der Einzelzeichen und Wörter,
- Zuordnung der Segmente und Klassifikationsergebnisse zu Elementen der Adressstruktur,
- Abgleich der Adresse mit gegebenen Adresslisten und
- Entscheidung über den endgültigen Verteilcode oder die Rückweisung der Erkennungsergebnisse.

Die Schritte werden nicht notwendigerweise in der genannten Reihenfolge abgearbeitet. So erfolgt zum Beispiel erst die vollständige Bestimmung und Erkennung des Postcodes, bevor weitere Adresselemente gesucht und klassifiziert werden. Im nächsten Schritt erfolgt eine Aufstellung der möglichen Zielorte, bevor mit der so generierten, eingeschränkten Liste dann der Worterkenner für die Verifikation des Zielortes aufgerufen werden kann.

2.3 HMM-Erkennungssystem: *Powerscript*

Im Adresslesesystem von Siemens Dematic werden verschiedene Verfahren zur Erkennung gebundener Handschrift eingesetzt. *Fastscript* ist ein struktureller Ansatz, der schnell arbeitet, aber geringere Erkennungsraten erzielt als *Powerscript*, das im weiteren behandelte System. *Powerscript* ist eine Eigenentwicklung der Siemens Dematic Postautomatisierung. Seine Implementierung beruht auf einem System, das am DaimlerChrysler Forschungsinstitut in Ulm für die Spracherkennung entwickelt wurde [Kal91]. Das statistische und segmentierungsfreie System beruht auf Hidden-Markov-Modellen. Es wird in den folgenden Kapiteln ausführlich beschrieben.

Anhand des identischen Ansatzes, der sich bis auf die Ebene der konkreten Implementierung niederschlägt, sieht man die enge Verwandtschaft von Handschrift- und Spracherkennung [Kal93c]. Beide sehen sich ähnlichen Anforderungen gegenübergestellt. In der Spracherkennung muss man sich in ähnlicher Weise mit dem Problem der Segmentierung beschäftigen, zumal es bei Sprache keine eindeutig identifizierbaren Objekte wie einzelne Zeichen gibt. Aus diesem Grund hat sich die Modellierung mit HMMs in der Spracherkennung schon früh durchgesetzt, bevor sie auf das Problem der Erkennung gebundener Handschrift übertragen wurde.

Kapitel 3

Hidden-Markov-Modelle

Die Schwierigkeit bei der Handschrifterkennung und der Spracherkennung liegt darin, dass die Segmentierstellen der beobachteten Daten nicht bekannt sind. Beim Abgleich eines Modells mit einer Folge von Beobachtungen müssen also die Grenzen der einzelnen Teilfolgen der Beobachtung variabel bleiben.

Eine ähnliche Situation besteht, wenn zwei verschiedene diskrete, zeitliche Folgen von Signalen miteinander verglichen werden sollen. Ein Verfahren dazu ist *Dynamic Time Warping* (DTW): Man sucht den besten Pfad durch eine Matrix, die durch die Beobachtungszustände der beiden Signalfolgen aufgespannt wird [Sak78]: Mit dynamischer Programmierung kann er effizient bestimmt werden. Dynamic Time Warping wird dazu verwendet, gegebene Signalfolgen mit einer Menge von Prototypen zu vergleichen und so eine Klassifikation durchzuführen. Eine Klasse wird bei DTW durch eine Menge von prototypischen Signalfolgen repräsentiert.

Im Gegensatz dazu sind *Hidden-Markov-Modelle* [Rab89b, Gha01] ein allgemeines statistisches Modell zur Beschreibung von Beobachtungsfolgen. An die Stelle prototypischer Beobachtungsfolgen tritt eine Menge von Modellzuständen. Die Klassifikation erfolgt – ähnlich wie bei DTW – durch einen Vergleich von Beobachtungs- und Zustandsfolge. Der Vorteil gegenüber DTW besteht darin, dass die statistischen Eigenschaften der Klassen mit HMMs besser erfasst werden können.

3.1 Definition

Ein HMM ist ein Zustandsgraph: ein System, das eine endliche Menge $Q = \{s_1, \ldots, s_N\}$ von Zuständen einnehmen kann, zwischen denen es zeitlich diskret wechselt. Zum Zeitpunkt t gibt der Zustand q_t eine beobachtbare Größe O_t aus (Abbildung 3.1).

Es gelten die Bedingungen für Markov-Ketten erster Ordnung: Die statistische Abhängigkeit eines Zustandes beschränkt sich auf den direkten Vorgänger. Der Name *Hidden-Markov-Modell* rührt daher, dass die Zustände q_t nicht direkt beobachtet werden können, sie sind versteckt (*hidden*). Ein HMM ist durch folgende Angaben charakterisiert:

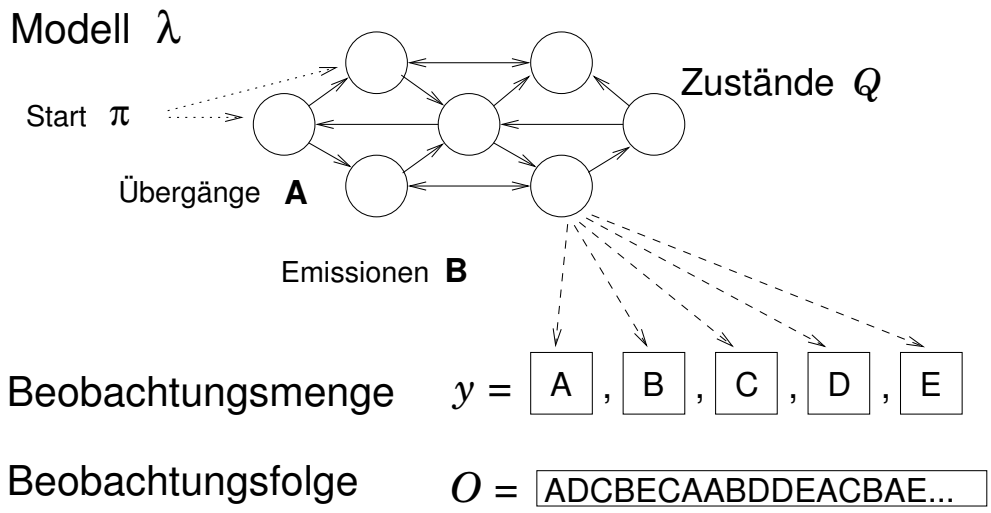

Abbildung 3.1: Hidden-Markov-Modell: Das Modell λ kann eine Menge von Zuständen Q einnehmen. Jeder Zustand emittiert verschiedene Beobachtungen y, dies erzeugt eine Beobachtungssequenz O.

- Die Anzahl N der Modellzustände. Sie sind nicht beobachtbar, haben aber meist eine konkrete Bedeutung und charakterisieren den zugrundeliegenden physikalischen Vorgang. Die richtige Wahl der Zustände ist die elementare Aufgabe bei der Modellierung des zu beschreibenden Prozesses.

- Der Menge der beobachtbaren Symbole $y \in V$: Sie spiegelt die Struktur der Beobachtungen wider und wird durch die Anwendung direkt vorgegeben. Es kann sich dabei um eine diskrete Menge $V = \{y_1, \ldots, y_M\}$ von Symbolen handeln oder um einen kontinuierlichen Merkmalsraum.

- Die Matrix $A = \{a_{ij}\}_{i,j=1,\ldots,N}$ gibt die Wahrscheinlichkeiten an, dass das System vom Zustand i in den Zustand j wechselt: $a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$. Die Übergangswahrscheinlichkeiten a_{ij} bilden bei N Zuständen eine $N \times N$ Matrix. Ist $a_{ij} = 0$, so ist kein Übergang zwischen den Zuständen i und j möglich. Auf diese Weise ist die Struktur oder Topologie des Modells durch die Übergangsmatrix A implizit gegeben. Auch das Verweilen im selben Zustand i ist bei Selbstübergängen mit $a_{ii} > 0$ möglich.

- Mit der Verteilung $B = \{b_j\}_{j=1,\ldots,N}$ werden die Wahrscheinlichkeiten für die Emission einer bestimmten Beobachtung O_t durch den Zustand j angegeben: $b_j(O_t) = P(O_t = y | q_t = s_j)$. Ist die Menge der möglichen Symbole diskret und beträgt ihre Anzahl M, so bilden Emissionswahrscheinlichkeiten B eine $N \times M$-Matrix. Sind die emittierten Symbole aus einem kontinuierlichen Raum, so beschreiben Wahrscheinlichkeitsdichtefunktionen die Emissionen.

- Der Vektor $\pi = \{\pi_i\}_{i=1,\ldots,N}$ beschreibt die Wahrscheinlichkeiten, mit denen sich das System zu Beginn einer Beobachtung in einem bestimmten Zustand befindet: $\pi_i = P(q_1 = s_i)$.

Das Modell und die beschreibenden Wahrscheinlichkeiten werden als zeitlich konstant angenommen. Häufig wird das Tripel $\lambda = (\pi, A, B)$ als kompakte Notation für die Eigenschaften eines HMM verwendet.

Modelle, bei denen die Übergangsmatrix A voll besetzt ist, bei denen also Übergänge zwischen sämtlichen Paaren von Zuständen möglich sind, bezeichnet man als *ergodisch*. Ist in A nur die obere Dreiecksmatrix belegt, spricht man von *links-rechts*-Modellen: Nur diese werden normalerweise für die Beschreibung von Signalen in der Sprach- und Handschrifterkennung verwendet, da sich diese Signale durch eine definierte Abfolge von Zuständen beschreiben lassen. Rücksprünge würden diese Ordnung stören. Selbstübergänge sind bei solchen Modellen dagegen erlaubt, sie modellieren die jeweilige Verweildauer in den Zuständen. Man unterscheidet weiter zwischen linearen Modellen, die neben Selbstübergängen ausschließlich solche in einen nächsten Zustand erlauben, und *Bakis*-Modellen, bei denen es auch Übergänge in den übernächsten Zustand gibt.

Ein Modell λ erzeugt eine Beobachtungsfolge $O = O_1, \ldots, O_T$ auf folgende Weise: Zum Startzeitpunkt $t = 1$ wird gemäß π ein Startzustand q_1 gewählt. Bis zum Erreichen von $t = T$ wiederholt sich dann folgender Prozess: Die Wahrscheinlichkeitsverteilung B bestimmt die Erzeugung der Beobachtung O_t, der nachfolgende Zustand q_{t+1} wird nach den Wahrscheinlichkeiten der Übergangsmatrix A gewählt und der Zeitzähler t um eins weitergezählt.

3.2 Fragestellungen

Im vorigen Kapitel wurde gezeigt, dass eine direkte Verwendung von HMMs darin bestehen kann, mit Hilfe eines Zufallsgenerators typische Beobachtungsfolgen zu *generieren*. Für Anwendungen in der Sprach- oder Schrifterkennung ist es aber sinnvoll, die Perspektive zu ändern: Typischerweise liegen bereits Symbolfolgen vor, und HMMs dienen als Rahmen zu ihrer *Interpretation*. Dabei stellen sich folgende Fragen: Wie kann man HMMs zur Klassifikation gegebener Signalfolgen verwenden? Und wie kann man die Parameter von HMMs bestimmen, um später eine Klassifikation durchführen zu können?

Diese Fragen entsprechen den drei grundlegenden Problemen, die im Zusammenhang mit HMMs zu lösen sind [Rab89b]. Gegeben seien ein Modell λ und eine Beobachtungsfolge O der Länge T:

- Wie berechnet man effizient die Wahrscheinlichkeitsdichte $P(O \mid \lambda)$ für die Generierung einer Beobachtungsfolge $O = O_1, O_2, \ldots, O_T$ durch das Modell λ ?
- Welche Zustandsfolge $q = q_1, q_2, \ldots, q_T$ erklärt die Beobachtung O in optimaler Weise?
- Wie bestimmt man die Parameter des Modells λ, so dass es eine gegebene Zustandsfolge optimal beschreibt, also $P(O \mid \lambda)$ maximal ist?

Die Lösung des ersten Problems kann zur *Erkennung* des gegebenen Signals verwendet werden. Beim Erkennen besteht die Aufgabe darin, aus einer Anzahl von Prototypen, die durch Modelle λ_k beschrieben werden, den wahrscheinlichsten zu finden:

$$k_{\text{best}} = \underset{k}{\text{argmax}}\ P(\lambda_k|O). \tag{3.1}$$

Mit der Bayes-Regel kann die bedingte Wahrscheinlichkeit umformuliert werden zu

$$P(\lambda_k \mid O) = \frac{P(O \mid \lambda_k) \cdot P(\lambda_k)}{P(O)}. \tag{3.2}$$

Die Wahrscheinlichkeit für das Auftreten der Symbolfolge $P(O)$ im Nenner kann dabei vernachlässigt werden, da sie für alle Modelle den gleichen Wert besitzt. Als Konstante hat sie bei der Suche nach dem Maximum (3.1) keinen Einfluss. Die a-priori Modellwahrscheinlichkeiten $P(\lambda_k)$ spiegeln die Auftretenshäufigkeiten der Modelle in der speziellen Anwendung wider. Sie können statistisch berechnet werden, häufig nimmt man aber an, dass alle Modelle gleich wahrscheinlich sind. Auch dieser Term geht deshalb nicht in die Berechnung ein. Mit Lösung des ersten Problems, der Berechnung von $P(O|\lambda_k)$ für alle Modelle k und der Bildung des Maximums kann also eine Erkennung durchgeführt werden.

Die Frage nach der besten Zustandsfolge, die mit der Lösung des zweiten Problems berechnet werden soll, kann als Versuch einer *Erklärung* der Beobachtungen gelten. Die berechnete Folge kann als eine Hypothese für die Vorgänge in dem Teil des Modells, der der Beobachtung nicht zugänglich ist, angesehen werden. Diese Information kann man verwenden, um der Beobachtung bestimmte Bedeutungen, etwa Segmentgrenzen, zuzuordnen. Es wird dadurch möglich, die innere Struktur der Modelle besser zu verstehen und Statistiken über einzelne Zustände zu berechnen. Wenn die Zustände bestimmte Bedeutungsinhalte modellieren, dann dient die Maximierung der Wahrscheinlichkeit – in analoger Weise zum ersten Problem – der Erkennung dieser Inhalte.

Die Bestimmung der HMM-Parameter durch die Anpassung an gegebene Signalfolgen in Problem 3 bezeichnet man als *Training* des Modells. Durch eine große Menge an Beobachtungen, die Trainingsstichprobe, wird das Modell optimal an den vorliegenden Anwendungsfall angepasst. Dies ist der kritische Punkt bei der Entwicklung eines Erkennungssystems: Nur durch die Definition der Trainingsstichprobe erfolgt die Repräsentation der Modelle, und die Güte der Modellschätzung bestimmt die Qualität des Systems. Dazu kommt, dass im Fall von HMMs mit kontinuierlichen Emissionsverteilungen sehr viele Parameter geschätzt werden müssen, was eine sehr große Trainingsstichprobe fordert. Da normalerweise nur Stichproben von begrenzter Größe vorhanden sind, müssen die Verteilungsdichtefunktionen B in geeigneter Weise parametrisiert werden, um die Anzahl der Variablen zu vermindern.

In den folgenden beiden Kapiteln werden algorithmische Lösungen zu den drei Problemen vorgestellt. Wie sich zeigen wird, stehen sie in einer engen Beziehung zueinander, sie bauen gewissermaßen aufeinander auf.

3.3 Erkennung

Die Erkennung erfolgt über die Lösung der ersten beiden im vorigen Abschnitt definierten Probleme. In beiden Fällen steht man dabei zunächst vor der Aufgabe, die Verbundwahrscheinlichkeit für eine Beobachtung $O = O_1, \ldots, O_T$ und einen bestimmten Zustandspfad $q = q_1, \ldots, q_T$ zu berechnen. Sie ist die Kombination der Wahrscheinlichkeiten für den Anfangszustand, alle Übergänge und Emissionen:

$$P(O, q \mid \lambda) = \pi_{q_1} b_{q_1}(O_1) \cdot \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(O_t). \tag{3.3}$$

Für die Lösung des ersten Problems soll die Wahrscheinlichkeit einer bestimmten Beobachtung berechnet werden. Sie ergibt sich durch Summation über sämtliche Folgen von Zuständen, die möglich sind:

$$P(O \mid \lambda) = \sum_{q \in Q^T} P(O, q \mid \lambda). \tag{3.4}$$

Das zweite Problem erfordert die Suche der maximalen Wahrscheinlichkeit aus allen diesen Folgen:

$$q_{\text{best}} = \operatorname*{argmax}_{q \in Q^T} P(O, q \mid \lambda). \tag{3.5}$$

In beiden Fällen wächst die Menge der möglichen Zustandsfolgen exponentiell mit der Länge der Beobachtungssequenz. In allen praktisch vorkommenden Situationen ist deshalb eine Berechnung sämtlicher Terme unmöglich. Es müssen deshalb alternative Verfahren gefunden werden. Durch Einsatz von dynamischer Programmierung [Cor90] ist eine Berechnung mit linearem Aufwand möglich. Erst dadurch werden HMMs für die praktische Anwendung interessant.

3.3.1 Forward-Backward-Algorithmus

Mit dem Forward-Backward-Algorithmus [Bau67] wird die Wahrscheinlichkeit berechnet, mit der das HMM eine gegebene Merkmalsfolge erzeugt hat, ohne dass dazu die Wahrscheinlichkeiten aller Folgen verdeckter Zustände berechnet werden müssen. Er beruht auf dem Prinzip dynamischer Programmierung, das darin besteht, die Berechnung in geeignete Teilprobleme zu zerlegen, die nur einmal gelöst werden müssen. Dies geschieht in diesem Fall dadurch, dass als Hilfsvariable die *Vorwärtswahrscheinlichkeiten*

$$\alpha_t(i) \equiv P(O_1, \ldots, O_t, q_t = s_i \mid \lambda) \tag{3.6}$$

eingeführt werden. Sie geben die Wahrscheinlichkeit an, dass sich das Modell zum Zeitpunkt t im Zustand i befindet und die Beobachtungssequenz $O_1, \ldots, O_t$ erzeugt hat. Die Vor-

wärtswahrscheinlichkeit lässt sich für alle Zeiten $t = 1, \ldots, T$ induktiv berechnen:

$$\alpha_1(i) = \pi_i b_i(O_1) \qquad 1 \leq i \leq N \tag{3.7}$$

$$\alpha_t(i) = \left(\sum_{j=1}^{N} \alpha_{t-1}(j) \cdot a_{ji} \right) \cdot b_i(O_t) \qquad 1 \leq i \leq N,\ t = 2, \ldots, T. \tag{3.8}$$

Die gesuchte Wahrscheinlichkeit für alle Zustandsfolgen berechnet der Forward-Algorithmus dann durch Summieren über alle Endzustände:

$$P = P(O \mid \lambda) = \sum_{j=1}^{N} \alpha_T(j). \tag{3.9}$$

In analoger Weise kann man die Berechnung vom Ende der Beobachtung her durchführen. Dies wird als Backward-Algorithmus bezeichnet. Man definiert die *Rückwärtswahrscheinlichkeiten*

$$\beta_t(i) \equiv P(O_{t+1}, \ldots, O_T; q_t = s_i \mid \lambda), \tag{3.10}$$

und führt die Rekursion in entgegengesetzter Richtung durch, beginnend zum Zeitpunkt T. Die Berechnungsvorschriften lauten jetzt

$$\beta_T(i) = 1 \qquad 1 \leq i \leq N \tag{3.11}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_i(O_{t+1}) \cdot \beta_{t+1}(j) \qquad 1 \leq i \leq N,\ t = T-1, \ldots, 1, \tag{3.12}$$

und man erhält durch

$$P = P(O \mid \lambda) = \sum_{j=1}^{N} \pi_j \cdot b_j(O_1) \cdot \beta_1(j) \tag{3.13}$$

das identische Ergebnis wie mit dem Forward-Algorithmus. Für die Lösung der Aufgabe wird der Backward-Algorithmus an dieser Stelle also nicht zusätzlich zum Forward-Algorithmus benötigt. Er wird hier aber dennoch erwähnt, da er in der Diskussion der beiden anderen zu lösenden Probleme Verwendung finden wird.

Das Produkt aus Vorwärts- und Rückwärtswahrscheinlichkeit gibt gerade die Verbundwahrscheinlichkeit an, dass das System die Beobachtung O emittiert und sich zum entsprechenden Zeitpunkt t außerdem gerade im Zustand i befindet:

$$P(O, q_t = s_i \mid \lambda) = \alpha_t(i) \cdot \beta_t(i). \tag{3.14}$$

Dadurch erhält man eine weitere Darstellung für

$$P = P(O \mid \lambda) = \sum_{j=1}^{N} \alpha_t(j) \cdot \beta_t(j) \qquad \text{für alle } t, \tag{3.15}$$

da zum Zeitpunkt t irgendein Zustand eingenommen werden muss. Mit

$$\gamma_t(i) \equiv P(q_t = s_i \mid O, \lambda) = \frac{P(O, q_t = s_i \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \tag{3.16}$$

definiert man die Wahrscheinlichkeit, dass sich das Modell bei einer vorliegenden Beobachtungssequenz O zur Zeit t im Zustand i befindet.

3.3.2 Viterbi-Algorithmus

In der Sprach- und Schrifterkennung wird häufig von der Annahme ausgegangen, dass sich die Signaldaten recht eindeutig bestimmten Zuständen zuordnen lassen. Deshalb ist es für die Klassifikation ausreichend, sich bei der Berechnung der Wahrscheinlichkeit der Beobachtung auf die spezielle Zustandssequenz zu beschränken, welche die *beste* Bewertung erzielt:

$$P(O, q^* \mid \lambda) = \max_{q \in Q^T} P(O, q \mid \lambda). \tag{3.17}$$

Stellt die Sequenz q^* tatsächlich eine sehr gute Zuordnung zu den Signalen dar, so ist das Maximum stark ausgeprägt und kann als Näherung für die Signalwahrscheinlichkeit $P(O \mid \lambda) \approx P(O, q^* \mid \lambda)$ dienen.

Das Maximum kann effizient mit dem Viterbi-Algorithmus [Vit67] berechnet werden. Das Prinzip der Berechnung ist zum *Forward-Backward*-Algorithmus identisch. Die Zerlegung des Problems in Einzelschritte erfolgt diesmal durch die Definition der Hilfsvariable

$$\delta_t(i) \equiv \max_{q_1,\ldots,q_{t-1}} P(O_1, \ldots, O_t, q_1, \ldots, q_{t-1}, q_t = s_i \mid \lambda), \tag{3.18}$$

welche jetzt die höchste Wahrscheinlichkeit aller Pfade, die zum Zeitpunkt t im Zustand i enden, bezeichnet. Die Berechnung erfolgt erneut induktiv und wird mit den folgenden Anweisungen durchgeführt:

$$\delta_1(i) = \pi_i b_i(O_1) \qquad 1 \le i \le N \tag{3.19}$$

$$\delta_t(i) = \max_{j=1,\ldots,N} \left(\delta_{t-1}(j) \cdot a_{ij}\right) \cdot b_i(O_t) \qquad 1 \le i \le N,\ t = 2, \ldots, T. \tag{3.20}$$

Die gesuchte Wahrscheinlichkeit ergibt sich durch Maximieren:

$$P^* = P(O, q^* \mid \lambda) = \max_{j=1,\ldots,N} \delta_T(j). \tag{3.21}$$

Der Viterbi-Algorithmus unterscheidet sich vom Forward-Algorithmus nur dadurch, dass die Bildung der Summe durch eine Maximumsuche ersetzt wird. Dadurch kann aber deutlich Rechenzeit eingespart werden: Bei der Realisierung des Algorithmus arbeitet man generell mit logarithmierten Wahrscheinlichkeiten, um zu verhindern, dass der zulässige Zahlenbereich unterschritten wird. Dies hat einen zusätzlichen Einfluss auf die relativen Kosten der einzelnen Operationen: Die Multiplikation vereinfacht sich zugunsten der Addition, während die Kosten für die Maximumsuche konstant bleiben. Diese Voraussetzungen machen die Viterbi-Berechnung für eine Implementierung bestens geeignet.

Die direkte Bedeutung des Viterbi-Algorithmus, neben den Optimierungsmöglichkeiten, besteht aber in der Lösung des zweiten Problems, der Bestimmung der besten Zustandsfolge. Zur konkreten Angabe der Folge ist allerdings ein weiterer Rechenschritt notwendig.

3.3.3 Rückverfolgung

Das zweite zu lösende Problem für HMMs besteht in der Frage, welche Folge von Zuständen q^* eine Beobachtung am besten *erklärt*. Dazu ist zunächst zu klären, welches Kriterium

die „beste" Zustandsfolge definiert. Die bisherigen Betrachtungen ergeben zwei mögliche Definitionen. Zum einen können diejenigen Zustände gewählt werden, die individuell ihre Auftretenswahrscheinlichkeit $P(q_t = s_i \mid O, \lambda)$ maximieren:

$$q_t^* = \operatorname*{argmax}_j P(q_t = s_i \mid O, \lambda) = \operatorname*{argmax}_j \gamma_t(j). \tag{3.22}$$

Man erhält eine Folge, von der eine Mehrzahl der Zustände die Beobachtung optimal beschreibt. Allerdings hat die Methode einen entscheidenden Nachteil: Wenn zum Beispiel nicht sämtliche Übergänge erlaubt sind ($\exists i, j : a_{ij} = 0$), dann können Abfolgen von Zuständen ausgewählt werden, die durch das HMM gar nicht erzeugt werden können. Für die Erkennung oder die Analyse der Vorgänge bei der Erkennung ist dieses Kriterium deshalb nicht geeignet.

Man möchte stattdessen die vollständige Sequenz bestimmen, die die Beobachtung mit der höchsten Wahrscheinlichkeit erzeugt. Der Wert dieser Wahrscheinlichkeit wurde mit dem Viterbi-Algorithmus bereits bestimmt. Um aber die konkrete Folge von verborgenen Zuständen zu bestimmen, ist noch ein zweiter Verarbeitungsschritt bei der Viterbi-Berechnung notwendig. Dieser arbeitet die Zustände in umgekehrter Reihenfolge ab und bestimmt jeweils den besten Vorgängerzustand. Man führt eine weitere Variable ψ ein, die den Vorgängerzustand bezeichnet. Sie wird mit

$$\psi_1(i) = 0 \tag{3.23}$$

$$\psi_t(i) = \operatorname*{argmax}_{j=1,\ldots,N} (\delta_{t-1}(j) \cdot a_{ij}) \tag{3.24}$$

bestimmt, parallel zur Berechnung von δ durch (3.19) und (3.20). Die Variablenfelder δ und ψ bezeichnet man als *Trellis* (Gitter). Im Gegensatz zum einfachen Viterbi-Algorithmus, der immer nur auf die letzte Spalte δ_{t-1} zugreift, muss für die zweite Berechnungsstufe der vollständige Trellis zugreifbar sein.

Die beste Sequenz q^* bestimmt man, indem man den wahrscheinlichsten Zustand zum Zeitpunkt T auswählt und im Zustandstrellis ψ bis zum Startpunkt zurückfolgt.

$$q_T^* = \operatorname*{argmax}_{j=1,\ldots,N} (\delta_t(j) \tag{3.25}$$

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t = T-1, \ldots, 1. \tag{3.26}$$

3.4 Training der Modellparameter

Die Bestimmung der Modellparameter ist der entscheidende Schritt in der Entwicklung eines Handschrifterkennungssystems. Durch sie soll gesichert werden, dass die Charakteristik einer prototypischen Beobachtung O durch das Modell optimal beschrieben wird. Die untersuchte Sequenz O ist in diesem Fall eine im allgemeinen sehr lange Folge, da sie das gesamte Spektrum aller möglichen Beobachtungen abdecken muss. Sie wird auch als *Trainingsstichprobe* bezeichnet. Als Kriterium für die Wahl der besten Modellparameter λ wird meist die

Wahrscheinlichkeit für die Erzeugung der Beobachtung, $P(O \mid \lambda)$, gewählt. Man spricht dann von *Maximum-Likelihood*-Training. Die Berechnung der Parameter erfolgt durch die Lösung des dritten der Grundprobleme für HMMs.

Es ist keine analytische Methode bekannt, die die Modellparameter an eine endliche Folge von Beobachtungen anpasst. Verwendet wird deshalb im allgemeinen das Baum-Welsh-Training [Bau70], ein iteratives Verfahren, mit dem lokale Maxima der Wahrscheinlichkeitsverteilung berechnet werden können. Zur exakten Lösung würde das Verfahren eine unendliche Menge an Trainingsdaten benötigen; die Größe der Stichprobe ist deshalb für die Qualität des Trainings essenziell. Die Iteration verwendet die *Expectation-Maximization*-(EM)-Methode [Dem77] . In jedem Schritt wird aus einem Vorgängermodell λ ein neues Modell $\overline{\lambda}$ erzeugt. Die neuen Modellwahrscheinlichkeiten werden jeweils aus der erwarteten Verteilung der Zustände, die mit dem alten Modell berechnet wird, bestimmt. Anschaulich bedeutet das, dass in einem ersten Schritt die Beobachtungssequenz segmentiert wird (wobei die Segmentierung „variabel“ erfolgen kann, die Zuordnung der Zustände zu den Beobachtungen also mit Wahrscheinlichkeiten beschrieben wird). In einem zweiten Schritt können mit der so markierten Stichprobe die Verteilungen neu geschätzt werden. Es kann bewiesen werden, dass ein mit dieser EM-Methode geschätztes Modell die Daten besser beschreibt als das Vorgängermodell, und dass somit ein lokales Optimum gefunden wird (vgl. Anhang A).

3.4.1 Fuzzy-Vektorquantisierung

Die Trainingsmethode ist zunächst auf diskrete Symbolfolgen eingeschränkt. Für kontinuierliche Beobachtungen, wie sie meistens auftreten, können die Signale über ein Codebuch in Symbole übersetzt werden. Man spricht von *semi-kontinuierlichen* HMM: Kontinuierliche Verteilungen werden modelliert, indem Symbole durch Symbolwahrscheinlichkeiten ersetzt werden. Diese Wahrscheinlichkeiten erhält man durch eine *Fuzzy-Vektorquantisierung*. Die Vorgehensweise wird durch Einführung einer neuen Variablen

$$\varphi_k(O_t) = P(O_t = y_k). \tag{3.27}$$

zur Beschreibung der Beobachtungssequenz realisiert. Sie gibt die Wahrscheinlichkeit an, dass es sich bei der Beobachtung zum Zeitpunkt t um das Symbol y_k handelt. Die Wahrscheinlichkeiten sind normiert, $\sum_k \varphi_k = 1$, und der Fall direkter Beobachtung diskreter Symbole kann mit $\varphi_k(O_t) = 1$ für die Beobachtung von $O_t = y_k$ und $\varphi_k(O_t) = 0$ für alle anderen Symbole beschrieben werden.

Durch genügend viele Symbole kann jede beliebige Verteilung kontinuierlicher Beobachtungen beschrieben werden. Jeder beliebige Klassifikator kann zur Quantisierung verwendet werden, sowohl Normalverteilungs- und Polynomklassifikatoren [Sch77], als auch Neuronale Netze [Rot00]. Die Wahl des Klassifikatortyps entscheidet über die Qualität der Modellierung der tatsächlichen Verteilung und die Anzahl der zu schätzenden Parameter. Auf die Auswahl und die Belehrung des Klassifikators selbst wird in diesem Abschnitt aber nicht

weiter eingegangen. Die Bereitstellung der Lernmenge für den Klassifikator erfolgt über die Erwartungswerte der Zustände, die mit der im folgenden beschriebenen Methode berechnet werden.

3.4.2 Forward-Backward-Training

Das Forward-Backward-Training implementiert die EM-Methode. Sie besteht darin, dass aus den Erwartungswerten der Zustände die neuen Eingangs-, Übergangs- und Emissionswahrscheinlichkeiten berechnet werden. Zur Bestimmung des neuen Parametersatzes $\overline{\lambda}$ verwendet man folgende Vorschrift:

$$\overline{\pi}_i = \text{erwartete Aufenthalte in Zustand } i \text{ bei Start der Beobachtung}, \tag{3.28}$$

$$\overline{a}_{ij} = \frac{\text{erwartete Anzahl Übergänge von Zustand } i \text{ nach } j}{\text{erwartete Anzahl Übergänge aus Zustand } i}, \tag{3.29}$$

$$\overline{b}_i(k) = \frac{\text{erwartete Anzahl Beobachtungen von } k \text{ aus Zustand } i}{\text{erwartete Aufenthalte in Zustand } i}. \tag{3.30}$$

Die erwartete Anzahl Übergänge aus einem Zustand ist gleich der erwarteten Aufenthalte in demselben. Zur Berechnung der Werte können die Zwischenergebnisse des Forward-Backward-Algorithmus mit dem aktuellen Parametersatz λ verwendet werden. So werden durch Gleichung (3.16) bereits die Aufenthaltswahrscheinlichkeiten

$$\gamma_t(i) \equiv P(q_t = s_i \mid O, \lambda) \tag{3.31}$$

bestimmt, die für die weitere Verarbeitung in einer Tabelle abgelegt werden. In ähnlicher Weise werden die Sequenzwahrscheinlichkeiten $\xi_t(i,j)$ definiert. Sie geben die Wahrscheinlichkeiten an, dass sich das System zum Zeitpunkt t im Zustand i, im nächsten Schritt aber im Zustand j befindet:

$$\xi_t(i,j) \equiv P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda). \tag{3.32}$$

Auch ξ lässt sich mit den in (3.6) und (3.10) definierten Hilfsvariablen α und β der Forward-Backward-Berechnung berechnen,

$$\xi_t(i,j) = \frac{P(q_t = s_i, q_{t+1} = s_j, O \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j) \cdot \beta_t(j)}, \tag{3.33}$$

und kann somit im Rahmen derselben Berechnung wie γ ermittelt werden. Es sei hier darauf hingewiesen, dass zwischen ξ und γ der natürliche Zusammenhang

$$\gamma_t(i) = \sum_j \xi_t(i,j) \tag{3.34}$$

besteht. Nachdem also die Wahrscheinlichkeiten berechnet werden können, erfolgt die Bildung der Erwartungswerte durch eine Mittelung über alle Zeitpunkte.

3.4.3 Neuschätzung mit Erwartungswerten

Ausgehend von den Auftretenswahrscheinlichkeiten werden die neuen Parameter $\overline{\lambda}$ geschätzt. Die neuen Eingangswahrscheinlichkeiten können mit

$$\overline{\pi}_i = \gamma_0(i) \tag{3.35}$$

direkt angegeben werden. Für die Übergangswahrscheinlichkeiten müssen die Erwartungswerte für die Aufenthaltswahrscheinlichkeiten γ und ξ bestimmt werden. Dazu mittelt man über die entsprechende Beobachtungsdauer:

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \tag{3.36}$$

Bei der Berechnung der neuen Emissionswahrscheinlichkeiten werden nun auch die Wahrscheinlichkeiten $\varphi_k(O_t)$ für die Beobachtungen der einzelnen Symbole verwendet. Diese erlauben neben diskreten Symbolfolgen auch die Repräsentation von kontinuierlichen Beobachtungen. In diesem Fall ist zu beachten, dass nicht die vollständige Verteilungsfunktion neu gelernt wird, sondern nur die Parametrisierung, die durch die Wahl der Quantisierung vorgegeben ist, und die den Emissionswahrscheinlichkeiten

$$\overline{b}_i(k) = \frac{\sum_{t=1}^{T} \gamma_t(i) \cdot \varphi_k(O_t)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{3.37}$$

entspricht.

Häufig besteht die Trainingsstichprobe aus verschiedenen, unabhängigen Sequenzen. Bei der Handschrifterkennung werden zum Beispiel einzelne, gekennzeichnete Wörter als Trainingsdaten verwendet. Für jede dieser N verschiedenen Beobachtungssequenzen φ^n ermittelt man dann die Aufenthaltswahrscheinlichkeiten γ^n und ξ^n separat. Bei der Bildung der Erwartungswerte wird dann zusätzlich über sämtliche Einzelsequenzen gemittelt. Die Eingangswahrscheinlichkeiten berechnen sich so zu

$$\overline{\pi}_i = \frac{1}{N} \sum_n \gamma_0^n(i). \tag{3.38}$$

Bei den Übergangs- und Emissionswahrscheinlichkeiten ist das Vorgehen analog. Die Berechnungsvorschriften lauten

$$\overline{a}_{ij} = \frac{\sum_n \sum_{t=1}^{T^n-1} \xi_t^n(i,j)}{\sum_n \sum_{t=1}^{T^n-1} \gamma_t^n(i)} \tag{3.39}$$

und

$$\overline{b}_i(k) = \frac{\sum_n \sum_{t=1}^{T^n} \gamma_t^n(i) \cdot \varphi_k^n(O_t)}{\sum_n \sum_{t=1}^{T^n} \gamma_t^n(i)}. \tag{3.40}$$

Die Neuschätzung erfolgt in mehreren iterativen Schritten, wobei die neu bestimmten Parameter jeweils zur erneuten Schätzung der Aufenthaltswahrscheinlichkeiten herangezogen

werden. Diese Iteration wird so lange ausgeführt, bis eine Abbruchbedingung erfüllt wird. Ziel der Parameterschätzung ist es, die *Likelihood* der Trainingsdaten zu optimieren. Dass sie tatsächlich konvergiert, wird in Anhang A nachgewiesen. Als Konvergenzkriterium dient die relative Verbesserung der Beobachtungswahrscheinlichkeit. Unterschreitet sie einen bestimmten Wert, so können die Parameter nicht mehr substantiell verbessert werden und das Training wird beendet.

Da beim Baum-Welsh-Training nur ein *lokales* Optimum erreicht wird, ist eine gute Initialisierung der Modellparameter wichtig. Zudem können gute Startwerte die Anzahl der notwendigen Iterationen deutlich reduzieren, was die Zeit, die für die Durchführung des Trainings notwendig ist, verringert. Die Übergangswahrscheinlichkeiten werden typischerweise mit überall identischen Werten initialisiert. Bei den Emissionswahrscheinlichkeiten kann man auf dieselbe Weise vorgehen; es werden so keine Symbolklassen bevorzugt, und man erhält in der ersten Trainingsiteration bei linearen Modellen eine Zuordnung, die einer gleichmäßigen, linearen Verteilung der Beobachtungen auf die Zustände entspricht. Dies ist häufig eine plausible Annahme. Bestehen jedoch Zusammenhänge zwischen den HMM-Zuständen und der Symbolmenge, so ist es ratsam, diese Information bei der Initialisierung des Trainings zu berücksichtigen.

3.4.4 Viterbi-Training

Neben dem Forward-Backward-Training wird manchmal auch das so genannte Viterbi-Training verwendet. Die Vorgehensweise ist prinzipiell zu der beim Forward-Backward-Training identisch, nur rücken an die Stelle der Wahrscheinlichkeiten direkte Beobachtungen. Diese werden aus dem besten Zustandspfad postuliert, der beim Viterbi-Algorithmus berechnet wird. Sei q^* die beste Zustandfolge, dann berechnen sich

$$\gamma_t(i) = \begin{cases} 1 & \text{wenn } q_t^* = s_i \\ 0 & \text{sonst,} \end{cases} \tag{3.41}$$

und

$$\xi_t(i,j) = \begin{cases} 1 & \text{wenn } q_t^* = s_i \text{ und } q_{t+1}^* = s_j \\ 0 & \text{sonst.} \end{cases} \tag{3.42}$$

Beide Werte behalten, wie beim Forward-Backward-Training, ihre Bedeutung als Wahrscheinlichkeiten und können für die weitere Behandlung – die Bildung von Erwartungswerten – identisch verwendet werden. Das Viterbi Training hat bei einer häufig vergleichbar guten Anpassung der Parameter an die Daten den Vorteil eines geringeren Rechenaufwands. Gute Ergebnisse können mit Viterbi-Training dann erzielt werden, wenn die Trainingsdaten eine eindeutige Zuordnung der Zustände erlauben.

Kapitel 4

Bildrepräsentation

Hidden-Markov-Modelle können sinnvoll für Erkennungsaufgaben eingesetzt werden, bei denen es darum geht, eine eindimensionale Folge von Merkmalen oder Symbolen zu klassifizieren. Die klassische Anwendung ist deshalb die Spracherkennung, bei der eine zeitliche Folge von akustischen Signalen vorliegt und ausgewertet werden soll. Bei der Handschrifterkennung ist das ursprüngliche Signal jedoch zweidimensional, weshalb die Bildinformation zunächst in eine Folge von Merkmalsvektoren transformiert werden muss. Dabei kann der Umstand ausgenutzt werden, dass eine eindeutige Schreibrichtung existiert – bei lateinischer Schrift von links nach rechts – durch die der zeitliche Vorgang des Schreibens räumlich abgebildet wird.

Es findet also eine Transformation der Bildinformation in eine kompakte Darstellung statt, durch die der Erkennungsprozess erleichtert wird. An diese Transformation wird dabei die Bedingung gestellt, dass sie ähnlichkeitserhaltend sein soll, so dass kleine Änderungen des Schriftzuges nur leichte Unterschiede bei den Merkmalsvektoren nach sich ziehen. Vor allem aber geht es auch darum, nur die für die Erkennung relevanten Information zu extrahieren. Störungen, etwa durch Unterstreichungen, ungenaues Digitalisieren oder fehlerhaftes Binarisieren, sollten so weit wie möglich eliminiert werden.

Kursive Handschrift zeichnet sich durch eine extrem hohe Variabilität aus, sowohl durch die Eigenheiten des Schreibstils verschiedener Schreiber, als auch durch die verwendeten Schreibmittel. Diese Variabilität spiegelt sich unter anderem in Variationen der Strichdicke, der Schriftgröße, der Drehlage, der Scherung und der relativen Länge von Ober- und Unterlängen wider. Solche für die Erkennung unwichtigen Informationen sollten durch Normierungen entfernt werden.

Das hier vorgestellte Verfahren [Cae93] zur Bildvorverarbeitung und Merkmalsgewinnung wurde am DaimlerChrysler Forschungsinstitut in Ulm entwickelt und bei Siemens Dematic an spezielle Anforderungen angepasst [Ueb97]. Das Verfahren kann Binärbilder verarbeiten, die eine Zeile mit einem oder mehreren handschriftlichen Wörtern darstellen. Die Generierung dieser Bilder geschieht meist bereits durch einen komplexen Prozess, in dem die Bilder gescannt und binarisiert werden, bevor eine Segmentierung einzelne Zeilen oder

Abbildung 4.1: Originalbild: Der Worterkenner erhält Wortbilder als Eingabe, die in einem vorhergehenden Segmentierungschritt aus Adressbildern ausgeschnitten wurden.

Wörter aus der Vorlage extrahiert (Abbildung 4.1).

4.1 Vorverarbeitung und Normierung

Die gesamte Vorverarbeitung besteht aus einer sequenziellen Abarbeitung einzelner Schritte, die im folgenden genauer beschrieben werden. Dabei werden verschiedene Aufgaben bearbeitet: Einige Methoden transformieren nur die interne Repräsentation des Bildes, damit die weiteren Schritte effizient ausgeführt werden können. Andere dienen der Schätzung von Bildparametern, etwa des Neigungswinkels, andere wiederum transformieren das Bild selber, zum Beispiel um die Neigung zu korrigieren. In manchen Fällen werden mehrere dieser Aufgaben in einem Schritt abgearbeitet, zum Beispiel bei der Skelettierung, bei der alle drei Berechnungsarten auf einmal durchgeführt werden.

4.1.1 Ikonische Operationen

Im ersten Schritt werden Störungen, die beim Scannen und Binarisieren des Bildes entstanden sind, behoben. Es wird ein so genannter „Zebrafilter" auf das Bild angewendet, der kleine horizontale und vertikale Lücken schließt. Je nach Qualität der Eingabedaten kann auf diesen Filter verzichtet werden.

4.1.2 Konturbildung

Durch eine Analyse der Zusammenhangsgebiete (*Binary Connected Components*, BCC) wird das Rasterbild in eine Repräsentation umgewandelt, die aus hierarchisch geschachtelten Konturen besteht, also sich überdeckenden, abwechselnd schwarzen und weißen Zusammenhangsgebieten (Abbildung 4.2). Dadurch wird eine erhebliche Kompression der Daten erreicht, ohne dass Bildinformation verloren geht, so dass die nachfolgenden Algorithmen sehr effizient arbeiten können.

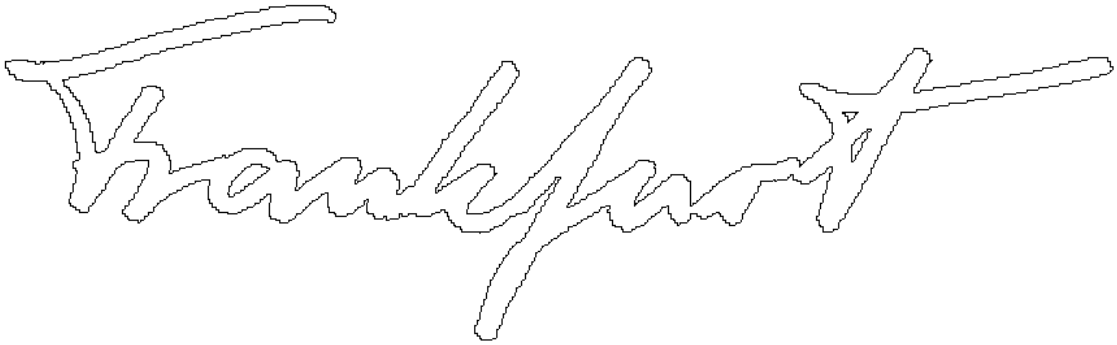

Abbildung 4.2: Konturbildung: Aus dem Rasterbild werden die Konturlinien extrahiert. Es entsteht eine hierarchische Struktur von ineinander geschachtelten Zusammenhangsgebieten.

Die BCC-Analyse [Man90] erfolgt zeilenweise und baut die neue Struktur in einem Durchgang auf. Ein BCC-Objekt besteht aus einem Umriss-Polygon, einer Farbe (schwarz oder weiß), dem umschreibenden Rechteck und einem Verweis auf die inneren Objekte, die vollständig von der Kontur des übergeordneten Objekts eingeschlossen werden. Durch die hierarchische Repräsentation werden Einschlüsse, die wichtige Merkmale für die Klassifikation der Zeichen sind, explizit dargestellt. Algorithmen, die nur die Kontur auf bestimmten Ebenen benötigen, können nachrangige BCC-Objekte ignorieren.

4.1.3 *Salt & pepper* - Entfernung

Ein Filter auf der BCC-Basis beseitigt kleine Objekte, die als „*salt & pepper*"-Effekte bekannt sind. Sie sind oft auf Störungen bei der Binarisierung zurückzuführen und somit unerwünscht. Das BCC-Sieb benutzt jeweils den Umfang des umschreibenden Rechtecks als Kriterium, ob ein BCC-Objekt entfernt werden soll.

4.1.4 Konturglättung

Die durch die BCC-Analyse generierte Kontur führt exakt entlang der rechtwinkligen Pixelgrenzen des Rasterbildes und besteht somit aus waagrechten und senkrechten Segmenten. Eine Glättung der Kanten dieses Polygons dient dazu, die allgemeine Strichrichtung besser darzustellen (Abbildung 4.3). Dies geschieht durch die sogenannte *Wall*-Approximation [Wal84].

Die Polygonpunkte werden der Reihe nach abgetastet. Eine Folge von Segmenten wird dann durch eine einzige Linie ersetzt, wenn die Fläche zwischen den beiden Polygonen unterhalb einer gesetzten Schwelle bleibt. Um eine möglichst glatte Kontur zu erhalten, die aus wenigen Polygonpunkten besteht, wird dieser Algorithmus iterativ ausgeführt. Dabei dient das im vorigen Lauf erzeugte Polygon als Eingabe für den nächsten Lauf.

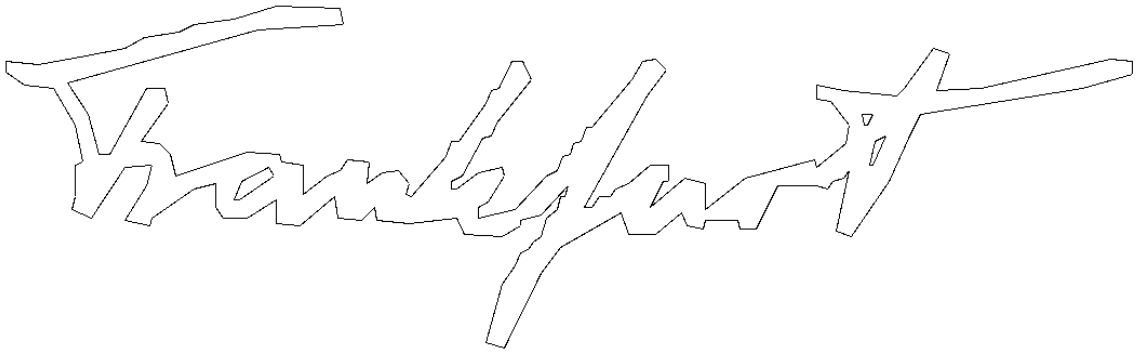

Abbildung 4.3: Konturglättung: Die kantigen Stufen der Kontur des Rasterbildes werden mit der *Wall*-Approximation geglättet. Dadurch wird die zu bearbeitende Datenmenge beträchtlich reduziert und die Schreibrichtung besser repräsentiert.

4.1.5 Neigungswinkel

Die Neigung bezeichnet den Winkel, den die Buchstaben zur Senkrechten einnehmen. Sie ist ein typisches Beispiel für die individuelle Ausprägung der Handschrift. Durch die Berechnung des Neigungswinkel und die anschließende Scherung des Bildes wird ein bedeutender Aspekt der Schreiberabhängigkeit beseitigt. In der Praxis hat sich gezeigt, dass die Neigung über ein gesamtes Wort als konstant angenommen werden kann.

Aus der geglätteten Kontur wird ein Histogramm über die Richtungswinkel der Liniensegmente aufgebaut. Die Segmente werden mit ihrer Länge gewichtet und es werden nur Segmente berücksichtigt, deren Ausrichtung in einem Intervall $[-50°, +50°]$ zur Senkrechten liegt. Das Histogramm wird sortiert und eine gewisser Prozentsatz der Einträge (15 %) als Ausreißer gewertet und nicht weiter berücksichtigt. Als Scherwinkel wird das gewichtete Mittel der verbliebenen Histogrammwerte berechnet.

4.1.6 Schreiblinienschätzung

Ein sehr wichtiges Merkmal für die nachfolgende Merkmalsextraktion ist die Schätzung der Schreiblinien. Die meisten Schreiber halten sich beim Schreiben auch auf leerem, unliniertem Papier an imaginäre Schreiblinien. Dadurch ist eine Normierung der Schreiberabhängigkeiten über die Bestimmung dieser Linien möglich. Sie können verwendet werden für:

- Normierung der Schräglage,
- Detektion von Ober- und Unterlängen,
- Höhennormierung der einzelnen Zonen des Schriftzuges.

Die korrekte Bestimmung der Schreiblinien ist ein sehr wesentlicher Punkt für die Güte des ganzen Systems. Fehler, die an dieser Stelle gemacht werden, sorgen für große Diskontinuitäten (kleine Änderungen zeigen große Wirkung) und können von den restlichen Stufen der Erkennungsprozedur meist nicht korrigiert werden.

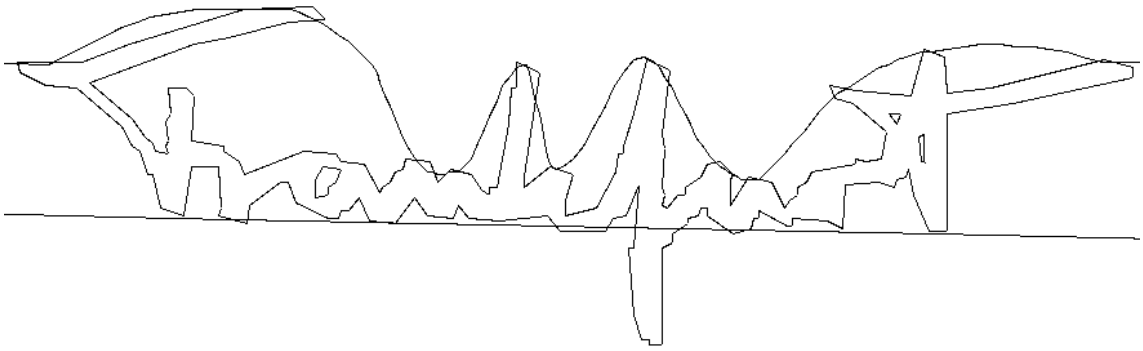

Abbildung 4.4: Schätzung der Schreiblinien: Als Basislinie wurde eine Gerade berechnet, die obere Linie approximiert mit *Splines* die Einhüllende. Das Bild wurde vor der Berechnung der Schreiblinien bereits geschert.

Es gibt die beiden Möglichkeiten, die Schreiblinien als Geraden oder als gebogene Linien zu modellieren. Im Falle von Geraden sind die zu bestimmenden Schreiblinien die Grundlinie und die Linie an der Oberkannte der Kleinbuchstaben ohne Oberlängen. Ihr Vorteil besteht darin, dass nur wenige Parameter zu schätzen sind und dies deshalb sehr genau getan werden kann, was insbesondere bei kurzen Wörtern einen Vorteil verspricht. Man hat ein starkes Modell für die zu erwartenden Eigenschaften des Schriftzuges, womit ein Mangel an verwertbaren Daten ausgeglichen werden kann.

Allerdings stellt sich gerade bei kurzen Wörtern heraus, dass bei der Bestimmung der Geraden häufig Fehler auftreten. Die Fehlinterpretation weniger Punkte des Wortbildes kann hier zu gänzlich falschen Linien führen. Bei längeren Wörtern kann man zudem feststellen, dass der Schreiber keine genaue gerade Linie einhalten konnte, wodurch ebenfalls Fehler entstehen müssen. In diesen Fällen ist die Berechnung von gebogenen Linien vorzuziehen. Jede der Methoden hat jedoch ihre Vor- und Nachteile; welche gewählt wird, ist von der speziellen Anwendung abhängig. Häufig entscheidet man sich unterhalb des Schriftzuges für eine gerade Basislinie, oberhalb davon für eine gebogene Linie (Abbildung 4.4).

Die Bestimmung der Geraden erfolgt in zwei Schritten. Zur Berechnung der *Basislinie* werden nur die Punkte der Projektion auf den unteren Rand herangezogen. Aus dieser Punktemenge werden die lokalen Minima berechnet, die die zu berechnende Gerade bestimmen werden. Dazu erhält jedes Minimum ein Gewicht, das sich aus der Breite der Umgebung bei vorgegebener Höhe berechnet. Auf diese Weise erhalten flache Minima ein höheres Gewicht als spitze. Dies entspricht der Beobachtung, dass beim Schreiben lange, horizontale Striche stabiler auf der Schreiblinie liegen als kurze, vertikale. Überschneiden sich die Bereiche um zwei Minima, so wird nur der tiefer liegende Punkt beachtet. Da der erste Buchstabe häufig vom restlichen Wort abgetrennt und vertikal versetzt steht, werden die beiden am weitesten links stehenden Punkte für die Schreiblinienberechnung verworfen. In mehreren Iterationen wird durch lineare Regression die Gerade durch die Menge gewichteter Punkte berechnet und jeweils der Punkt mit der größten euklidischen Distanz verworfen, bis die restliche Varianz eine gegebene Schwelle unterschreitet.

Die Bestimmung der *Oberkante der Kleinbuchstaben* ist schwieriger, da viele Oberlängen störend einwirken, geschieht aber prinzipiell auf die gleiche Weise. Es wird auf die Projektion auf die obere Kante verzichtet, da sonst wichtige Maxima durch breite Großbuchstaben, zum Beispiel weite T-Striche, verdeckt werden. Die Gewichtung der Punkte erfolgt wie bei der Bestimmung der Basislinie, und es wird auch eine bestimmte Menge der Maxima am Wortanfang verworfen. In der ersten Iteration wird das Wissen über die Lage und Basislinie eingebracht und die erste Regressionsgerade parallel zu ihr gelegt. Für die weiteren Iterationen gilt diese Bedingung jedoch nicht mehr.

Wird der Schriftverlauf durch gebogene Linien dargestellt, so verzichtet man auf die klassische Definition von Schreiblinien. Stattdessen werden als Begrenzungen die obere und untere Einhüllende des Wortes herangezogen [Wol97]. Zwei Voraussetzungen sollen die Linien dabei erfüllen, um den Charakter der Schrift nachzubilden: Sie sollten möglichst wenig oszillieren, und sie dürfen sich nicht berühren oder sogar schneiden. Beide Anforderungen bedingen sich gegenseitig.

Die Linien werden durch *Splines* approximiert. Die Stützstellen zur Definition der Splines werden auf die gleiche Weise berechnet wie zur Bestimmung der Geraden. Polynome dritten Grades definieren den Verlauf der Linien in den einzelnen Teilintervallen zwischen den Stützstellen:

$$P_i(x_i) = A_i + B_i(x - x_i) + C_i(x - x_i)^2 + D_i(x - x_i)^3. \tag{4.1}$$

Die Bedingung gilt, dass die Funktionswerte an benachbarten Intervallgrenzen x_i durch denselben Punkt (x_i, y_i) mit identischer Steigung t_i verläuft, damit sich eine glatte Linie ergibt. Die Steigung berechnet sich mit Hilfe der zwei benachbarten Stützstellen jeweils links und rechts vom Punkt i. Durch sie wird eine mittlere Steigung berechnet, wobei der Einfluss der beiden Seiten durch ein Maß für die Konstanz der Steigung gewichtet wird. Rechts und links vom Wertebereich müssen dazu zwei Punkte extrapoliert werden, deren y-Werte durch ein Polynom zweiten Grades bestimmt werden. Eine Interpolation nach Akima [Aki72] berechnet die Polynomkoeffizienten der Segmente:

$$\begin{aligned} A_i &= y_i, \\ B_i &= t_i, \\ C_i &= \frac{\frac{y_{i+1}-y_i}{x_{i+1}-x_i} - 2t_i - t_{i+1}}{x_{i+1} - x_i}, \\ D_i &= \frac{t_i + t_{i+1} - 2\frac{y_{i+1}-y_i}{x_{i+1}-x_i}}{(x_{i+1} - x_i)^2}. \end{aligned} \tag{4.2}$$

Um eine möglichst glatte Einhüllende zu erhalten, die den Prozess des Schreibens widerspiegelt, muss auch hier die Menge der Stützstellen eingeschränkt werden [Wol97]. Es werden Stützstellen entfernt,

- die jenseits oder nahe an der gegenüberliegenden Linie liegen,

- die starke Schwankungen der lokalen Steigung bewirken und
- deren x-Werte sehr nahe beieinander liegen.

4.1.7 Skelettierung

Ein zentrales Konzept bei der Verarbeitung von handschriftlichen Eingaben ist der *Strich*. Für die Erkennung des Inhalts ist nur die Trajektorie des Stifts von Belang, die durch den Mittelpunkt des Strichs gegeben ist (Abbildung 4.5). Durch eine Normierung der Strichdicken kann von Eigenschaften des Stifts abstrahiert werden.

Es ist einfach, aus einer Linie ein Rasterbild zu erzeugen, das umgekehrte Vorgehen – die *Skelettierung* – ist schwieriger. An einen Skelettierungsalgorithmus werden verschiedene Ansprüche gestellt: Das erzeugte Skelett soll glatt sein; störende Artefakte, zum Beispiel abstehende Haarlinien, sind unerwünscht. Die Anzahl der Verzweigungspunkte sollte minimal sein und die Endpunkte von Linien nicht gekürzt werden. Möglichst sollten Strichdickeninformationen am Skelett vorhanden sein, nicht notwendig ist aber eine fehlerfreie Rekonstruierbarkeit des Originalbildes.

Es wird ein konturbasierter Skelettierungsalgorithmus von Kwok [Kwo88] verwendet. Dieser geht von einer speziellen Konturdarstellung des Ursprungsbildes aus, das als binäres Pixelbild vorliegt. Jedes Objekt dieser Struktur repräsentiert ein Randpixel und kann bestimmte Zustände einnehmen. Diese zeigen beispielsweise an, ob ein Objekt schon abgearbeitet wurde, wie weit sein Abstand zum äußeren Rand ist, oder ob es schon zum endgültigen Skelett gehören wird. Iterativ werden neue, innenliegende Konturen aufgebaut, wobei die Konturobjekte seriell abgearbeitet werden und in jeder Iteration ihre Zustände ändern können. Die Iteration endet, sobald alle übrig gebliebenen Pixel ihren Endzustand eingenommen haben. Das entstandende Skelett wird anschließend mit der oben beschriebenen *Wall*-Approximation weiter geglättet.

Die verwendeten Schriftproben besitzen meist eine Strichdicke von nur wenigen Pixeln. Der Algorithmus arbeitet mit diesen Daten recht effektiv und erzielt gute Ergebnisse. Dennoch wird auf die Berechnung des Skeletts häufig verzichtet. Der Grund dafür ist, dass die Berechnung der Merkmalsvektoren (im folgenden Abschnitt 4.2) mit beliebigen Segmentfolgen für die Repräsentation des Schriftzugs durchgeführt werden kann. Es hat sich herausgestellt, dass das Skelett nur geringe Vorteile gegenüber der Konturdarstellung besitzt, da hauptsächlich die Richtung und Länge der Segmente in die Berechnungen eingehen. Deshalb schlägt sich der erhöhte Rechenaufwand meist nicht in einer höheren Erkennungsleistung nieder.

4.1.8 Normierungstransformationen

Die Dreh- und Schieflage des Wortbildes kann über die Lage der Basislinie bestimmt werden. Aus diesem Grund wird die Basislinie in jedem Fall berechnet, auch wenn für die Be-

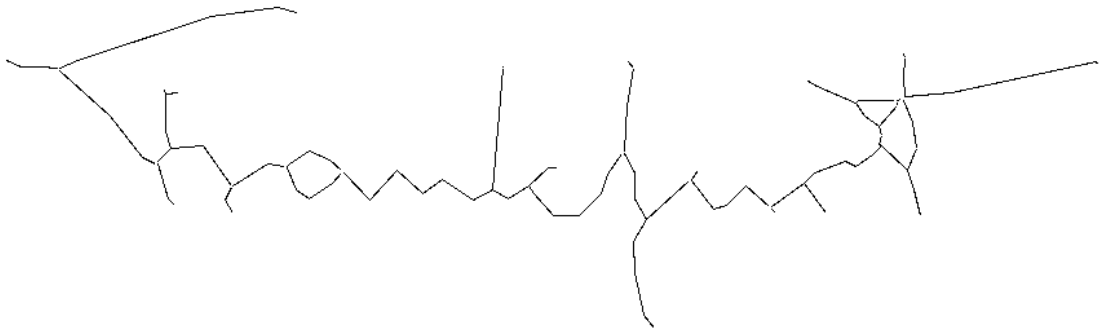

Abbildung 4.5: Skelettierung: Aus der Kontur wird die Trajektorie des Strichs nachgebildet.

rechnung der Merkmale die Einhüllende herangezogen wird. Durch eine Rotation des Bildes kann von dieser Variablen abstrahiert werden. Die Neigung der Schrift wird durch eine Scherung kompensiert.

Es ist nicht notwendig, die Transformation am Rasterbild durchzuführen, stattdessen werden nur die Punkte der Polygonrepräsentation transformiert. Die Transformationsmatrix wird dazu im Vorhinein berechnet. Man erhält auf diese Weise vernachlässigbare Rechenzeiten.

Eine Größennormierung des Bildes findet an dieser Stelle nicht statt. Sie wird implizit bei der Merkmalsgewinnung durchgeführt.

4.1.9 Ablaufsteuerung

Es gibt keine eindeutig festgelegte Ablaufsteuerung, die die Reihenfolge der einzelnen vorgestellten Operationen vorgibt. Dies wurde in den vorigen Abschnitten deutlich, in denen häufig nur von der *Möglichkeit* einer Berechnung die Rede war. Zu den Variablen im Programmablauf gehören zum Beispiel die Fragen, ob das Bild vor oder erst nach der Berechnung der Basislinien geschert werden soll, welche Art der Basislinien verwendet wird, und ob eine Skelettierung überhaupt durchgeführt werden soll. Über diese Freiheitsgrade hinaus gibt es in der Vorverarbeitung eine große Anzahl von Parametern, die angegeben werden müssen. Deutlich sieht man dies bei der Berechnung der Schreiblinien. Zahlreiche Vorgaben legen fest, wieviele und welche der Stützstellen herangezogen werden sollen.

Gerade die Schreiblinien haben einen wesentlichen Einfluss auf die Güte der Vorverarbeitung. Die Wahl der freien Parameter ist deshalb stark von der zugrunde gelegten Aufgabenstellung abhängig. Die Schreibstile unterscheiden sich stark: Ist beispielsweise der Anteil von Großbuchstaben-Blockschrift sehr hoch, dann ist die Definition einer Schreiblinie als Oberkante der Kleinbuchstaben meist nicht geeignet. Werden sehr dicke Stifte als Schreibmittel gebraucht, so kann unter Umständen nicht auf eine Skelettierung verzichtet werden.

Betrachtet man unterschiedliche Schriftarten, so wird die Notwendigkeit einer frei para-

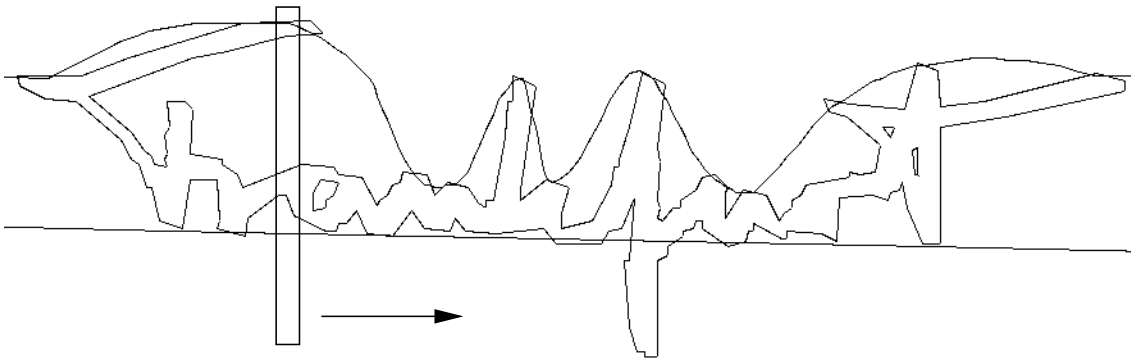

Abbildung 4.6: Merkmalsberechnung: Ein Rahmen wird über das Bild geschoben. Auf diese Weise wird aus dem zweidimensionalen Bild eine Merkmalsfolge.

metrisierbaren Vorverarbeitung noch offensichtlicher. Stellt die untere Basislinie bei lateinischer Schrift einen starken Anhaltspunkt dar, so ist dies bei arabischer Schrift häufig bereits nicht mehr gegeben. Letztendlich hilft nur die visuelle Kontrolle der mit unterschiedlichen Parametern durchgeführten Vorverarbeitungen. Dabei bereitet bereits häufig das Erfassen der *Ground-truth*-Daten Schwierigkeiten. Die Definition, von welchen Merkmalen der Eingangsdaten abstrahiert werden kann und muss, ist dabei bereits der wesentliche Punkt. Die Anpassung der Bildverarbeitung an die gewählte Aufgabenstellung bleibt somit eine Tätigkeit, die nicht vollständig automatisiert werden kann.

4.2 Merkmalsgewinnung

Im Anschluss an die Vorbereitung und Normierung des Wortbildes erfolgt der entscheidende Schritt, dass die zweidimensionale Bildinformation in eine Folge von Merkmalsvektoren überführt wird, die die Beobachtungssequenz für dass HMM darstellt. Der Bildinhalt muss durch diese Folge in geeigneter Weise repräsentiert werden.

4.2.1 Geometrische Merkmale

Zu diesem Zweck wird ein schmales, vertikal ausgerichtetes Fenster, das als Schlitzsonde betrachtet werden kann, in diskreten Schritten in Schreibrichtung über das Bild geschoben (Abbildung 4.6). Jeder Fensterposition ist ein Vektor der Folge zugeordnet. Aus den Liniensegmenten innerhalb des Fensters werden Merkmale identifiziert und quantifiziert, die die Elemente des entsprechenden Vektors darstellen.

In den vorbereitenden Schritten fand bisher keine Größennormierung des Bildes statt. Sie wird bei der Berechnung der Merkmalsvektoren implizit durchgeführt, indem die Größe des Fensters dynamisch an die Bildgröße angepasst wird. Dazu erfolgt eine grobe Schätzung der Schriftgröße durch eine Auswertung der Größe des umschließenden Rechtecks und des Abstands der Schreiblinien. Die Schriftgröße bestimmt die Breite des Fensters und die

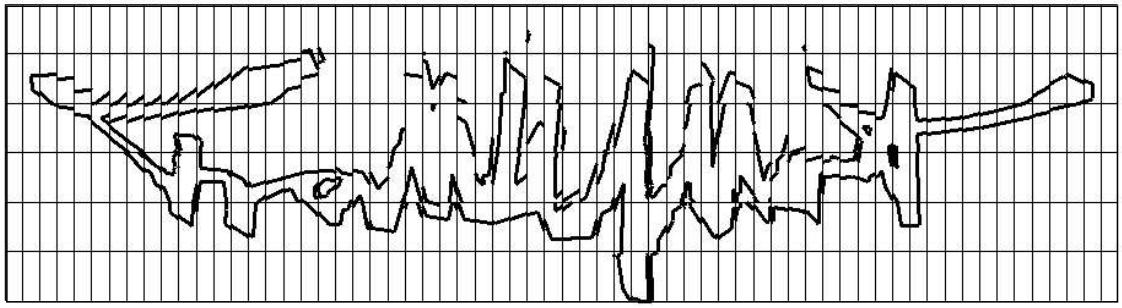

Abbildung 4.7: Merkmalsbereiche: Das normierte Bild wird in verschiedene Bereiche eingeteilt, in denen aus den Liniensegmenten Merkmale berechnet werden. Die einzelnen Zonen überlappen sich dabei, damit die Merkmale stabil gegen kleine Variationen sind.

Schrittweite: Die Breite wird so gewählt, dass ein Rahmen etwa ein Drittel bis ein Fünftel der mittleren Buchstabenbreite einnimmt.[1]

Die Höhe des Fensters erstreckt sich immer über das gesamte Bild, und der Wert der berechneten Merkmale versteht sich immer relativ zur absoluten Höhe des Fensters. Auf diese Weise werden Schreibweisen, bei denen die Buchstaben sehr schmal oder sehr breit sind, einander angeglichen. Daneben gibt es allerdings Schreibstile, bei denen Ober- oder Unterlängen stark betont werden oder aber nicht so stark ausgeprägt sind. Um diese Variabilität zu erfassen, wird das Fenster in verschiedene Zonen unterteilt, die durch die berechneten Schreiblinien bestimmt werden (Abbildung 4.7). Je nach Wahl der Schreiblinien (vgl. Abschnitt 4.1.9) werden unterschiedliche Einteilungen der Zonen vorgenommen. Man betrachtet meist sechs Bereiche: Unterhalb der Basislinie (*Bottom*), im Bereich der Basislinie (*Base*), zwischen den Schreiblinien (zwei Bereiche, *Center* und *Upper Center*), im Bereich der Oberkante der Kleinbuchstaben (*XEnd*) und darüber (*Top*).

Die Sondenzonen überlappen sich in vertikaler Richtung in ähnlicher Weise wie die Fenster horizontal. Der Grad der Überlappung bemisst sich relativ zur Ausdehnung der Zonen und beträgt in etwa 10–20%. Die Stabilität der berechneten Merkmale wird so erhöht und es wird verhindert, dass Merkmale aufgrund ungünstiger Zoneneinteilungen übersehen werden. Folgende Merkmale werden berechnet:

Linien $|-/\backslash$ **:** Es werden Linien detektiert, die vertikal, horizontal oder diagonal durch die Zone verlaufen. Die Merkmale geben an, wie weit die Zone von den Liniensegmenten in der entsprechenden Richtung durchmessen wird.

Punkte $\cdot$ **:** Dieses binäre Merkmal gibt an, ob einzelne isolierte Punkte innerhalb der Zone vorhanden sind. Das wird dadurch definiert, dass es Liniensegmente gibt, die nicht mit den benachbarten Zonen verbunden sind.

[1] Die Struktur der HMMs ist eng mit der Breite des Merkmalfensters verbunden. Die hier beschriebene Fensterbreite passt optimal zu links-rechts Modellen der Länge drei. Der Inhalt dieser Arbeit hat unter anderem zum Ziel, dieser Abhängigkeit automatisch Rechenschaft zu tragen.

	Geraden Oberkante Kleinbuchstaben	Splines Oberkante ist Einhüllende
Top	$\cdot \mid -$	$\mid$
XEnd	$\mid -/\backslash$	$\mid -/\backslash$
Upper Center		$\mid -/\backslash$
Center	$( ) \mid -/\backslash$	$\mid -/\backslash$
Base	$\vee \mid -/\backslash$	$\vee \mid -/\backslash$
Bottom	$\mid$	$\mid$

Tabelle 4.1: Merkmalsgewinnung: Auswahl der in den einzelnen Zonen berechneten Merkmale für zwei typische Vorverarbeitungskonfigurationen mit verschiedenen Schreiblinien.

Rundungen $( )$ **:** Bögen, die nach rechts oder links offen sind und sich über die gesamte Zone erstrecken. Ihr Vorhandensein wird daran gemessen, ob es verbundene Segmente gibt, von denen sich je eines in der oberen und der unteren Hälfte der Zone befindet, die nahe der Mitte der Zone verbunden sind und an dieser Verbindung einen entsprechenden Winkel bilden.

Spitzen $\vee$ **:** Mit Spitzen sind nach unten weisende Zacken gemeint: Sie bestehen aus zwei verbundenen Segmenten, die den oberen Rand berühren. Im Idealfall stehen sie möglichst senkrecht und bilden einen spitzen Winkel.

Für die Berechnung der Merkmale ist die Repräsentation des Bildes durch Liniensegmente Voraussetzung. Da bei der Berechnung der Merkmale hauptsächlich die Strichrichtung und -länge ausgewertet wird, können Kontur- und Skelettdarstellung in identischer Weise behandelt werden.

In den verschiedenen Zonen werden jeweils unterschiedliche Merkmale berechnet. Die Wahl richtet sich dabei nach den typischen Merkmalen, die die Schrift in dem entsprechenden Bereich auszeichnen. Oberhalb der Oberkante der Kleinbuchstaben charakterisieren zum Beispiel Aufstriche, T-Striche und i-Punkte die Schrift. Entsprechend werden waagrechte und senkrechte Striche sowie Punkte detektiert. Im Bereich der Kleinbuchstaben dominieren geschwungene Auf- und Abstriche, also wird unter anderem das Vorhandensein von Bögen bewertet. Wie die Wahl der Schreiblinien die Zoneneinteilung bestimmt, so beeinflusst sie auch die Wahl der Merkmale in den Zonen. Tabelle 4.1 gibt einen Überblick über die Merkmale für die beiden Fälle, dass die Oberkante der Kleinbuchstaben oder die obere Einhüllende als obere Schreiblinie berechnet wird. Bei der unteren Schreiblinie spielt die Wahl der Darstellung als Gerade oder gebogene Linie keine Rolle.

Ist die obere Schreiblinie die Einhüllende, so wird die Dimension der Merkmalsvektoren um eins erweitert und als zusätzliches Merkmal der Abstand zwischen oberer und unterer Schreiblinie dazugenommen. Diese wichtige Information über den Schriftzug würde sonst durch die Normierung der Schrifthöhe ignoriert werden.

4.2.2 Dynamische Merkmale

Um der dynamischen Struktur der Schrift besser gerecht zu werden, kann es sinnvoll sein, bei der Beschreibung des Ausschnitts einen größeren Bereich des Schriftzuges zu berücksichtigen als den, der durch das eigentliche Fenster vorgegeben wird. In dieser Arbeit werden deshalb Merkmale verwendet, die jeweils zwei zusätzliche Fenster links und rechts vom aktuellen Merkmal betrachten. Dabei werden zwei unterschiedliche Strategien verwendet.

Die eine Methode besteht darin, einfach mehrere Vektoren, die direkt aus den Segmentmerkmalen gebildet wurden, in einen Vektor zusammenzufassen, der dann als eigentliche Beobachtung gewertet wird. Im speziellen Fall wird ein Vektor der fünffachen Länge erzeugt. Die Besonderheit der Nachbarschaftsbeziehung muss dann nicht weiter behandelt werden. Bei der anderen Methode wird die Dynamik explizit durch die Bildung so genannter Δ-Vektoren dargestellt. Ein Δ-Vektor wird durch die gemittelte Differenz zu den zwei benachbarten Vektoren gebildet:

$$\Delta v_t = \frac{1}{2}[(v_{t+1} - v_t) + (v_t - v_{t-1})] = \frac{1}{2}(v_{t+1} - v_{t-1}). \tag{4.3}$$

Auf dieselbe Weise erzeugt die Differenz von Δ-Vektoren einen ΔΔ-Vektor. Jeder Fensterbereich wird durch drei Vektoren dargestellt. Man macht die Annahme, dass die drei Vektoren voneinander statistisch unabhängig sind, und quantisiert jeden Vektor mit einem separaten Codebuch.

In jedem der beiden Fälle ist es notwendig, an Anfang und Ende der Beobachtungssequenz zwei zusätzliche Nullvektoren zu stellen. Im Gesamtsystem, das Training und Erkennung umfasst, finden beide der hier vorgestellten Methoden zur Berücksichtigung von Merkmalsdynamik Anwendung. Das System wird in den folgenden beiden Kapiteln beschrieben.

Kapitel 5

Schriftmodell

Die Vorgehensweise bei der Handschrifterkennung mit HMMs besteht darin, die Bildinformation eines Schriftzuges zunächst in eine handhabbare, eindimensionale Vektorfolge zu transformieren. Anschließend wird das Problem der Segmentierung dieser Folge mit dem Formalismus der Hidden-Markov-Modelle behandelt. Für die Konstruktion eines konkreten Systems stellt sich die Frage, wie die Struktur der HMMs aufgebaut sein soll, damit sie den Anforderungen der speziellen Anwendung entspricht. Es geht darum, mit dem gegebenen Formalismus der HMMs ein Modell zu entwickeln, das Handschrift in geeigneter Weise repräsentiert.

Lesen findet immer in einem bestimmten Kontext statt. Insbesondere bei unleserlicher Schrift ist es auch für Menschen wichtig, das Anwendungsgebiet zu kennen, in das eine schriftliche Äußerung eingebettet ist. Ein klassisches Beispiel stellen handgeschriebene Arztrezepte dar, die für Nicht-Mediziner meist nicht zu entziffern sind. Ganz generell hilft linguistischer Kontext, Geschriebenes mit größerer Sicherheit zu erkennen. Im Zusammenhang mit automatischer Schrifterkennung wird solches Vorwissen typischerweise in *Wörterbüchern* mitgegeben, deren Einträge die Menge der Wörter, die im aktuellen Kontext auftreten können, darstellen. Ein Schriftmodell muss in der Lage sein, mit beliebigen, auch dynamisch generierten Wörterbüchern umgehen zu können. Deshalb werden bei dem hier betrachteten System einzelne Buchstabenmodelle, aus denen beliebige Wörter erzeugt werden können, als Grundlage für die Schriftmodellierung verwendet.

Viele Buchstaben besitzen so genannte *Allographen*, also unterschiedliche Schreibweisen derselben Bedeutung. Ein einleuchtendes Beispiel dafür sind Groß- und Kleinschreibung. Die Struktur der Modelle sollte diese Eigenschaft widerspiegeln. Darüber hinaus stellen sich weitere Fragen, wie die Topologie der Buchstaben-HMMs insgesamt aussehen sollte, um die Vielfalt der Schriftdaten möglichst gut repräsentieren zu können. Es geht unter anderem darum, festzulegen, wieviele Zustände gebraucht werden, wie diese miteinander verbunden sind, und wie die Emissionen der Beobachtungen modelliert werden. Erst wenn diese Fragen geklärt sind, kann ein System konstruiert werden, das sich – indem es lediglich eine größtmögliche, für die Aufgabenstellung repräsentative Stichprobe in einem Lernalgo-

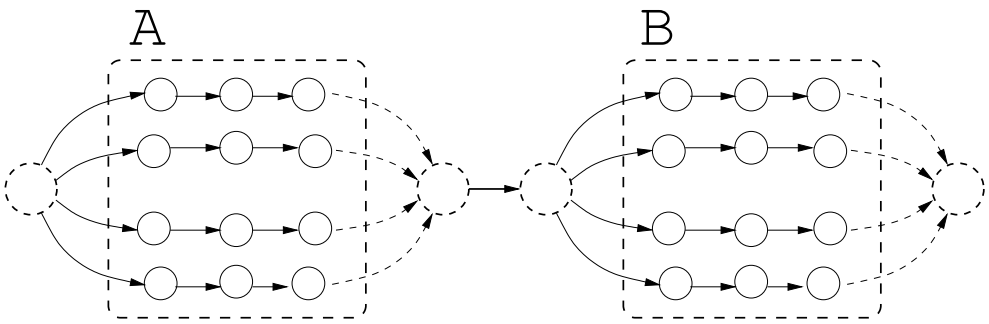

Abbildung 5.1: Buchstabenmodelle: Jeder Buchstabe ist durch ein HMM beschrieben. Zur Erkennung eines ganzen Wortes werden Buchstaben-HMMs zu einem einzigen Wort-HMM verkettet. Dazu besitzt jedes Buchstabenmodell am Anfang und am Ende insgesamt zwei virtuelle Zustände, also solche, die keine Beobachtungen emittieren. Die realen Übergänge zwischen End- und Startzuständen werden durch Produkte aus drei Wahrscheinlichkeiten beschrieben.

rithmus verarbeitet – auf die speziellen Schreibweisen adaptiert.

5.1 Buchstaben-HMMs

Für die Erkennung von Wortbildern muss man zunächst die entsprechenden *Wortmodelle* bilden, um damit – für Training und Erkennung – den Forward-Backward- bzw. Viterbi-Algorithmus ablaufen zu lassen. Die Grundbausteine dafür sind durch die einzelnen Buchstaben-HMMs gegeben. Sie müssen eine gewisse Struktur aufweisen, damit aus ihnen sinnvolle Wort-HMMs gebildet werden können.

Es wird postuliert, dass durch die Art der Merkmalsgewinnung sichergestellt ist, dass die einzelnen Modellzustände eine definierte Abfolge von Beobachtungen beschreiben. Die Schriftmodellierung kann so durch die in Abschnitt 3.1 definierten links-rechts-HMMs erfolgen. Diese sind dadurch eingeschränkt, dass es nur einen einzigen möglichen Startpunkt in der Zustandsfolge gibt, der zwingend den Anfang der Beobachtung beschreibt. Dementsprechend wird auch die Struktur der einzelnen Buchstaben auf links-rechts-Modelle mit definiertem Anfangszustand festgelegt.

Um die Modelle möglichst einfach verbinden zu können, definiert man *virtuelle Zustände*, die den Beginn und das Ende jedes Buchstabens repräsentieren. Der virtuelle Startzustand stellt den definierten Einstiegspunkt in ein Buchstabenmodell dar. Wortmodelle werden durch Erzeugen von Übergängen zwischen Start- und Endzuständen aufeinanderfolgender Buchstaben gebildet (Abbildung 5.1). Die beiden Zustände bezeichnet man als *virtuell*, weil sie keine Emission erzeugen; bildlich gesprochen „verbrauchen“ sie keine Beobachtung. Die Übergänge zwischen Zuständen verschiedener Buchstabenmodelle erfolgen über mehrere virtuelle Zwischenzustände. Sie entsprechen einem einzigen Übergang zwischen den Zuständen, dessen Wahrscheinlichkeit sich über das Produkt aller beteiligten Übergangs-

wahrscheinlichkeiten ergibt.

In dem Fall, dass ein einziges Wortmodell als Folge von Buchstaben-HMMs erzeugt wird, beträgt die Übergangswahrscheinlichkeit zwischen virtuellen Zuständen jeweils eins. Es ist aber auch möglich, beim Aufbau des eigentlichen HMM an jeder Stelle mehr als nur jeweils zwei Buchstabenmodelle miteinander zu verbinden. Für die Erkennung ist zum Beispiel die Realisierung von Baumstrukturen sinnvoll, in denen alle möglichen Wörter enthalten sind. Es stellt sich die Frage, welche Wahrscheinlichkeiten in diesem Fall den Übergängen zwischen den virtuellen Zuständen zugeordnet werden sollen. Über die Angabe von Wahrscheinlichkeitswerten kann an dieser Stelle Vorwissen über die entsprechende Anwendung eingebracht werden. Das kann zum Beispiel darin bestehen, dass während der dynamischen Generierung von Wörterbüchern den Wörtern unterschiedliche a-priori-Wahrscheinlichkeiten zugeordnet werden. Es können aber auch statistische Informationen über die Häufigkeit bestimmter Buchstabenkombinationen verwertet werden. Meist ist es aber nicht sinnvoll, solches Vorwissen bereits bei der Worterkennung mit einzubeziehen. Insbesondere bei der Erkennung unter Verwendung von Wörterbüchern ist es für die abschließende Bewertung der Erkennungsergebnisse einfacher, isolierte und unabhängige Wahrscheinlichkeiten für die Wortgüten zu erhalten.

Für die Implementierung ist es ausreichend, mit einem einzigen virtuellen Zustand zwischen den Buchstabenmodellen zu arbeiten. Der virtuelle Startzustand besitzt keine direkte Repräsentation, die Übergänge zu den realen Zuständen werden durch die Eingangswahrscheinlichkeiten π des Buchstaben-HMM dargestellt. Den Endzustand bezeichnet man als Absorptionszustand. Er wird bei der Berechnung der Zwischenergebnisse von Viterbi- und Forward-Algorithmus direkt miteinbezogen. Nach Abschluss jeder Iteration, die eine einzelne Beobachtung beschreibt, werden die entsprechenden Hilfsvariablen $\alpha_t(\cdot)$ oder $\delta_t(\cdot)$ auch für den Absorptionszustand berechnet und dienen dann als Ausgangspunkt für die nächste Iteration. Diese Vorgehensweise ist sinnvoll, um die Anzahl der ausgeführten Berechnungen klein zu halten. Bezeichnet man die Anzahl der Endzustände mit E und die der Anfangszustände mit A, so sind statt $A \cdot E$ Übergängen nur noch $A + E$ Übergänge zu berechnen. Die Symmetrie von Anfangs- und Endzustand wird bei der konkreten Repräsentation dadurch aufgehoben, dass die Berechnung in einer bestimmten Richtung durchgeführt wird. Für den Backward-Algorithmus stellt der virtuelle Eingangszustand den Absorptionszustand der Buchstaben dar.

Der Zusammenbau des Wort-HMM erfolgt durch Verweise auf die jeweiligen Buchstabenmodelle (Abbildung 5.2). Das reduziert nicht nur den notwendigen Speicherplatz und die Rechenzeit beim Aufbau der Modelle, sondern beschleunigt auch die Berechnung der Algorithmen. Da die Parameter der Buchstaben-HMM-Zustände mehrfach vorkommen können, wird vermieden, dass identische Emissionswahrscheinlichkeiten bei der Berechnung doppelt berechnet werden müssen.

Eine Besonderheit des hier betrachteten Systems bei der Wahl der Buchstabenmodellierung stellen die so genannten *Joker*-Modelle dar. Es sind vom Aufbau her normale Buchsta-

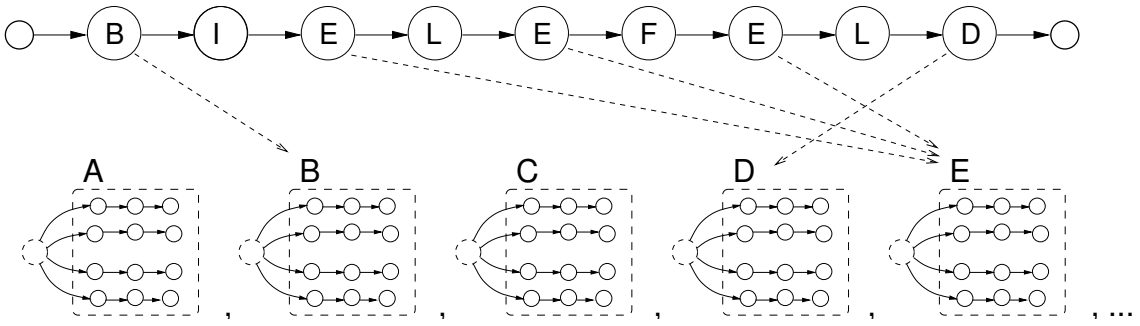

Abbildung 5.2: Generierung der Wortmodelle: Die Wort-HMMs werden nur durch Verweise auf ein Alphabet aus Buchstabenmodellen aufgebaut. Das hat mehrere Vorteile: Der Speicherverbrauch ist gerade bei großen Wörterbüchern sehr gering, bei der Erkennung kann durch Redundanz Rechenzeit eingespart werden, und beim Training besteht ein direkter Zugriff auf die zu modifizierenden Buchstabenmodelle.

benmodelle, die aber keine konkreten Buchstaben, sondern ein Platzhalter für alle Zeichen repräsentieren, indem sie den Querschnitt aller auftretenden Zeichen modellieren. Ihre Aufgabe ist es, bei der Erkennung nicht vorgesehene Gestaltklassen zu repräsentieren und so die Robustheit des Systems zu steigern. Mit ihnen kann die Qualität von Schriftproben bestimmt werden, indem auch Wörter, die nur aus Joker-Modellen bestehen, klassifiziert werden. Man hat so ein Vergleichsmaß, das zur Bestimmung der Güte der Erkennung herangezogen werden kann.

5.2 Modelltopologie

Die Buchstabenmodelle müssen in der Lage sein, die Vielfalt und Komplexität der Schrift vollständig zu beschreiben. Der Entwurf der Modelltopologie, also der Verbindungsstruktur der einzelnen Modellzustände, hat darauf großen Einfluss. Zum einen muss die Topologie die verschiedenen Gestaltklassen der Buchstaben repräsentieren können. Für denselben Buchstaben gibt es viele verschiedene Schreibweisen; Groß- und Kleinschreibung, Blockschrift und Handschrift unterscheiden sich in ihrer Erscheinung zum Teil fundamental. Zum andern muss das Modell die Komplexität der Zeichen erfassen können und die notwendigen Zustände zur Beschreibung zur Verfügung stellen.

Die hier verwendete Struktur [Kal93a, Kal93b] hält für jede Gestaltklasse einen eigenen HMM-Zustands-Pfad bereit, Die Pfade liegen parallel, Übergänge *zwischen* den einzelnen Pfaden sind nicht vorgesehen (Abbildung 5.3). Es wird davon ausgegangen, dass sich Allographen so weit unterscheiden, dass es keinen Sinn macht, die Zustände untereinander zu verbinden. Für die Wahl der Eingangswahrscheinlichkeiten in die einzelnen Pfade gibt es mehrere Möglichkeiten. Verwendet man für die Erkennung den Viterbi-Algorithmus, so kann es Sinn machen, alle Wahrscheinlichkeiten als eins anzunehmen. Die verschiedenen Pfade werden in diesem Fall als eigene (Sub-)Modelle mit gleicher Bedeutung angesehen.

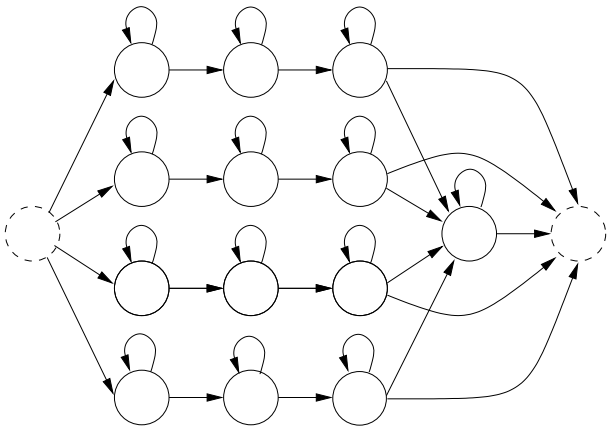

Abbildung 5.3: HMM-Buchstabenmodell: Jeder Buchstabe kann in verschiedenen Gestaltklassen auftreten, die durch verschiedene Zustandspfade modelliert sind. Die Zustandspfade sind lineare links-rechts-Modelle. Zwischen den einzelnen Buchstaben werden optionale Pausenmodelle für Leerstellen eingeführt, die Bestandteil des Buchstabenmodells sind.

Auf diese Weise wird bei der Suche nach dem besten Pfad ihre Auswahl nicht durch apriori-Annahmen über die Gestalt beeinflusst. Man kann die Eingangswahrscheinlichkeiten aber auch mit dem normalen Training des Systems bestimmen. Besonders bei ähnlichen Gestaltpfaden kann der Forward-Algorithmus dann Mischformen besser erkennen.

Zwischen zwei Buchstaben kann jederzeit ein Leerraum auftreten. Dies wird durch Einfügen eines optionalen Pausezustands (der also übersprungen werden kann) am Ende jedes Buchstabenmodells ermöglicht. Die Pausezustände aller Modelle werden identisch modelliert und teilen sich ihre Parameter.

Die einzelnen Zustandspfade sind lineare links-rechts-HMMs. Übergänge sind nur zum nächsten Zustand möglich, dadurch ist festgelegt, dass jede Beobachtung durch eine definierte Abfolge von Zuständen beschrieben wird. Die Selbstübergänge der Zustände modellieren die Buchstabenbreite. Eine typische Wahl für die Länge der Zustandspfade sind drei Zustände, mit denen Anfang, Mitte und Ende der Buchstaben repräsentiert werden. Diese Wahl spiegelt die Generierung der beobachteten Merkmale wider: Zum Beispiel beeinflusst die Breite des Merkmalsfensters die optimale Pfadlänge.

Für den Aufbau des Erkennungssystems wurde bisher generell der angegebene Wert drei für die Länge der Pfade verwendet. Die Anzahl der Pfade wurde ebenfalls entsprechend der Aufgabenstellung fest vorgegeben und betrug aufgrund der typischen Unterteilung in große und kleine Block- und Handschrift meist vier. Es ist Thema und Inhalt dieser Arbeit, die Kriterien für die optimale Auswahl dieser Parameter genauer zu spezifizieren; das wird in Teil II genau beschrieben werden. Die Grundstruktur der hier vorgestellten Topologie bleibt allerdings bei allen weiteren Schritten erhalten.

5.3 Emissionen

Die vom HMM zu repräsentierenden Beobachtungen sind durch die Merkmalsvektoren v_t gegeben, die aus den Bildinformationen extrahiert werden, wie in Abschnitt 4.2 beschrieben wurde. Prinzipiell liegt damit der Fall von kontinuierlichen HMMs vor. Es muss nun eine geeignete Beschreibung der Emission dieser Beobachtungen durch die einzelnen Zustände s_n gefunden werden. Gesucht ist eine geeignetes Modell für die Wahrscheinlichkeitsverteilung $P(v \mid s_n)$ im Merkmalsraum. Sie sollte mit Hilfe eines durch die Größe der Stichprobe beschränkten Parametersatzes in der Lage sein, die reale Verteilung der Daten wiederzugeben. Man spricht bei einer solchen Modellierung von semi-kontinuierlichen HMMs. Eine hier geeignete Modellierung ist die Überlagerung von Gaußverteilungen:

$$P(v \mid s_n) = \sum_k g_{nk} \cdot \mathcal{N}(v, \mu_{nk}, K_{nk}), \tag{5.1}$$

$$\mathcal{N}(v, \mu, K) = \frac{1}{\sqrt{(2\pi)^N \det K}} \cdot \exp\left[-\tfrac{1}{2}(v-\mu)^T K^{-1}(v-\mu)\right]. \tag{5.2}$$

Für jeden Zustand s_n geben die Faktoren g_{nk} die Gewichtung der speziellen Gaußfunktionen an, die durch die Mittelwerte und Kovarianzen $C_n = (\mu_{nk}, K_{nk})$ charakterisiert sind. Die verschiedenen Gaußverteilungen aller Zustände werden in einer einzigen großen Menge $C = (\mu_k, K_k)$ zusammengefasst, in der die Basisverteilungen aller Zustände vertreten sind. Die einzelnen Zustände unterscheiden sich nur durch die Gewichtungsfaktoren g_{nk}. Ein Beschreibung durch eine einzige Menge C ist möglich, auch wenn sich die Verteilungen der verschiedenen Zustände gegenseitig nicht überschneiden. Dann wird einfach jeder Zustand durch spezielle Gaußfunktionen aus der Gesamtmenge beschrieben und die anderen Verteilungen durch Gewichtungsfaktoren Null ausgeblendet.

Die Merkmale von verschiedenen Buchstaben sind teilweise jedoch so ähnlich, dass man von einer großen Überlappung der Verteilungen ausgehen kann. Der Vorteil dieser Beschreibung besteht dann darin, dass die verschiedenen Zustände sich die Parameter der Verteilungen teilen können. Bei der begrenzten Menge an verfügbaren Daten können die Gaußverteilungen beim Training des Systems so besser geschätzt werden. Da die Zustände über die Mischung der Verteilungen miteinander verbunden sind, bezeichnet man diesen Aufbau der HMMs als *tied-mixture*-Modell.

Die Berechnung der Emissionen erfolgt in zwei Stufen, die es erlauben, die HMMs trotz der kontinuierlichen Merkmalsvektoren fast wie diskrete Modelle zu behandeln. Man spricht bei einer solchen Modellierung deshalb auch von *semi-kontinuierlichen* HMMs [Hua89]. In der ersten Stufe wird die Menge der einzelnen Gaußverteilungen als Codebuch verwendet, mit dem eine Fuzzy-Vektorquantisierung durchgeführt wird. Den Codebuchklassen y_k werden Wahrscheinlichkeitsverteilungen

$$\varphi_K(v) \equiv P(v \mid y_k) = \mathcal{N}(v, \mu_k, K_k) \tag{5.3}$$

zugeordnet, die schon in der Beschreibung des Trainings in Abschnitt 3.4 verwendet wurden. Die Symbolmenge V, aus deren Elementen die eigentlichen – für das HMM sichtbaren –

Beobachtungen O_t bestehen, wird also durch die Codebuchklassen y_k definiert:

$$O_t \in V = \{y_1, \ldots, y_{k_{\max}}\}. \tag{5.4}$$

Auf dieser Basis wird in der zweiten Stufe die Dichtefunktion der Zustandsemissionen,

$$P(v \mid s_n) = \sum_k P(v \mid y_k) \cdot P(y_k \mid s_n), \tag{5.5}$$

berechnet. Durch einen Vergleich mit (5.1) sieht man, dass die diskreten Emissionswahrscheinlichkeiten $b_n(\cdot)$ den Gewichten der Gaußverteilungen entsprechen:

$$b_n(O_t) = P(O_t = y_k \mid s_n) \equiv g_{nk}. \tag{5.6}$$

Diese Gewichte können im Rahmen des in Abschnitt 3.4 beschriebenen Trainings adaptiert werden. Die Bestimmung der Parameter der Gaußverteilungen (μ_k, K_k) selber geschieht in einem separaten Schritt.

In dieser Arbeit erfolgt die Beschreibung der Emissionen ausschließlich über einen Normalverteilungsklassifikator. Es können beliebige andere Klassifikatoren zur Quantisierung verwendet werden, zum Beispiel Polynomklassifikatoren oder Neuronale Netze, da durch sie ebenfalls Wahrscheinlichkeitsdichtefunktionen beschrieben werden [Sch96]. Viele Arbeiten verwenden Neuronale Netze und interpretieren die Aktivität der Ausgabeneuronen als Wahrscheinlichkeiten [Bou96, Rob96, Wil97b, Rot00]. Solche Systeme werden häufig als *hybride Systeme* bezeichnet. Sie werden in dieser Arbeit jedoch nicht weiter betrachtet.

Als Spezialfall tritt die Situation auf, dass mehrere Codebücher vorhanden sind, die als statistisch unabhängig angenommen werden. Dies ist der Fall, wenn in der Vorverarbeitung die voneinander unabhängigen Vektoren v, Δv und $\Delta\Delta v$ zur Beschreibung der dynamischen Bildinformation eines Fensters geliefert werden (Abschnitt 4.2.2). Für jeden dieser drei Vektoren wird eine eigene Vektorquantisierung durchgeführt. Die Emissionen der HMM-Zustände n werden dann durch mehrere Emissionswahrscheinlichkeiten $b_{n,c}(\cdot), c = 1, \ldots, 3$ beschrieben, die miteinander kombiniert werden. Besonders in der Spracherkennung werden häufig mehrere Datenströme, zum Beispiel verschiedene Frequenzbänder, auf diese Weise verarbeitet. Man spricht von *multiple streams* [Lee90].

5.4 Training der Parameter

Das grobe Design des Erkennungssystems ist an dieser Stelle festgelegt: Die Modellierung erfolgt mit Buchstaben-HMMs, deren Topologie und Emissionsmodellierung beschrieben wurde. Für eine konkrete Repräsentation der Schrift müssen nun die Variablen des Systems bestimmt werden. Dies sind zum einen die Parameter der Buchstaben-HMMs, also der Satz $\lambda = (\pi, A, B)$ der Eingangs-, Übergangs- und Emissionswahrscheinlichkeiten, dessen prinzipielle Methode zur Schätzung bereits in Abschnitt 3.4 beschrieben wurde. Zum anderen müssen geeignete Codebücher gefunden werden, im vorliegenden Fall also die durch

Mittelwerte und Kovarianzen $C = (\mu, K)$ definierten Verteilungsdichten der Klassen, die die beobachteten Daten beschreiben. Den gesamten Vorgang bezeichnet man als *Adaption* oder *Training* des Systems.

Zur Konfiguration wird nur vorgegeben, welche Buchstaben zu unterscheiden sind, und zunächst auch, wieviele Schreibvarianten es für jeden dieser Buchstaben gibt. Im übrigen stützt sich die Adaption ausschließlich auf statistische Daten. Die einzige Information, die berücksichtigt wird, besteht aus einer Menge von Trainingsdaten. Dies sind Wortbilder, die mit der Beschreibung des jeweiligen Wortes gekennzeichnet sind. Bei dieser Kennzeichnung, *Label* genannt, kann es sich um eine bloße Textdarstellung des Wortbildes handeln, es kann aber auch die Gestaltinformation der einzelnen Buchstaben als zusätzliche Information enthalten sein. Man spricht dann von einem *Finelabel*. Je detaillierter die Kennzeichnung ist, umso besser kann das System adaptiert werden, allerdings steigen auch die Kosten für die Erstellung der Trainingsdaten, die in der Regel manuell erzeugt werden. Als Kompromiss wird dem vollständigen Wort häufig eine einzige Gestaltinformation zugeordnet, die dann für alle Buchstaben gilt.

Die Trainingsdaten enthalten keinerlei Information über die Segmentierung des Wortbildes in Einzelzeichen, es werden für die Adaption auch keine Annahmen über das Aussehen der verschiedenen Buchstaben gemacht. Durch diese Entscheidung wird verhindert, dass die wahre Gestalt der Schrift, wie sie sich in den realen Trainingsdaten findet, durch irgendwelche Vorbedingungen verfälscht wird.

Die Schwierigkeit besteht nun darin, dass zu Beginn des Trainings keine Information über die Struktur der Beobachtungsvektoren vorliegt. Man steht vor der paradoxen Aufgabe, dass man zur Belehrung der HMM-Parameter ein Codebuch benötigt, dass man ein solches aber erst dann effektiv schätzen kann, wenn eine Segmentierung – und damit Kennzeichnung – der Beobachtungen vorliegt, für die man aber ein bereits adaptiertes HMM-System benötigt. Die Lösung des Problems besteht darin, die Adaption in mehreren Schritten durchzuführen:

- Zunächst wird ein erstes Codebuch erzeugt. Dazu wird durch unüberwachte Clusterung der Beobachtungsraum in Klassen aufgeteilt.

- Mit dieser Klasseneinteilung erfolgt ein erstes *Baum-Welsh*-Training. Man verfügt dann über ein erstes einsatzfähiges Schrifterkennungssystem, das die Schriftproben rudimentär repräsentieren kann.

- Das System wird mit dem vorliegenden Parametersatz zur Erkennung der Trainingsdaten selber eingesetzt. Man erhält somit eine Segmentierung der Trainingsdaten in Einzelzeichen.

- Ein neues Codebuch wird erzeugt, indem jedem HMM-Zustand eine Klasse zugeordnet wird. Mit der Segmentierinformation können die Verteilungsfunktionen der Klassen geschätzt werden. Die Anzahl der so entstandenen Klassen wird durch Clusterung

reduziert.

- Das neue Codebuch wird benutzt, um die HMM-Parameter erneut zu trainieren.

Die Größe der Trainingsstichprobe beträgt je nach Anzahl der zu belehrenden Buchstabenmodelle etwa 5000–20000 Bilder. Für jedes Zeichenmodell müssen ausreichend Daten zur Verfügung stehen, damit die Wahrscheinlichkeitsverteilungen der Klassen bestimmt werden können. Ab dieser Größe hat man für die meisten Modelle genügend Daten. Seltene Klassen, die nicht oder nur schlecht geschätzt werden können, werden durch die Clusterung mit ähnlichen Modellklassen zusammengelegt, so dass ein Training auch mit wenigen Daten immer erfolgreich durchgeführt werden kann.

5.4.1 Erste Klasseneinteilung

Im ersten Schritt des Trainings geht es darum, ein beliebiges Codebuch zu erzeugen, um einen Startpunkt für das Training des gesamten Systems zu haben. Ein geeignetes Verfahren ist der LBG-Algorithmus (*Linde-Buzo-Gray*, [Lin80]). Er benötigt keine Information über die Struktur der Daten, und die Anzahl der zu erzeugenden Klassen kann vorgegeben werden. Zur Belehrung werden einfach sämtliche Merkmalsvektoren v_t aus allen Trainingsbildern genommen.

Ziel der Vektorquantisierung ist es, jeden Merkmalsvektor v_t einer bestimmten Klasse y_k zuzuordnen. Über eine Abstandsfunktion $d(v, y_k)$ lässt sich der Abstand des Vektors zu jeder Klasse berechnen. Die dazu notwendige Klassenbeschreibung besteht zumindest aus einem Mittelpunktsvektor. Die Klassenzuordnung erfolgt über den minimalen Abstand. Der Merkmalsraum wird in verschiedene disjunkte Teilräume S_k aufgeteilt, die gerade sämtliche Punkte minimalen Abstands zu der jeweiligen Klasse enthalten.

Der LBG-Algorithmus ist ein hierarchisches, agglomeratives Cluster-Verfahren. Beginnend mit einer einzigen Klasse, die den gesamten Merkmalsraum umfasst, wird die Menge der Klassen $V_M = \{y_1, \ldots, y_M\}$ immer weiter vergrößert, indem die einzelnen Cluster aufgespalten und neu berechnet werden. Man startet mit einem Codebuch $V_1 = \{y_1\}$, das nur aus einem einzigen Cluster besteht, dessen Zentrum durch den Mittelwert aller Vektoren gegeben ist. Die Größe des Codebuchs verdoppelt sich in jeder Iteration $V_M \rightarrow V_{2M}$, indem jeder Cluster in zwei Hälften aufgeteilt wird:

$$V_{2M} = \{y_k^{+\varepsilon}, y_k^{-\varepsilon}; k = 1 \ldots M\}. \tag{5.7}$$

Die Trennung der Cluster erfolgt über einen Zufallsvektor ε, dessen Betrag klein gegen den typischen Abstand der Clusterzentren ist. In diese Richtung werden die Zentren der neuen Cluster $y_k^{\pm\varepsilon}$ gegen die ursprüngliche Klassenbeschreibung verschoben. Durch die Klassenzuordnung über den minimalen Abstand entspricht dies einer Aufteilung der Gebiete S_k entlang der durch ihren Mittelpunkt und den Vektor ε definierten Fläche.

Nach jeder Aufspaltung werden die Cluster durch eine Variante des k-means Verfahrens neu berechnet. Es geht darum, eine möglichst gute Aufteilung in verschiedene Cluster zu erhalten. Man definiert dazu die Störung

$$D = \frac{1}{T} \sum_{k=1}^{M} \sum_{v_t \in S_k} d(v_t, y_k), \tag{5.8}$$

die beschreibt, wie weit die Datenpunkte im Mittel von den Clusterzentren entfernt sind. Die Klassenbeschreibungen werden iterativ neu bestimmt, bis die Störung ein lokales Minimum erreicht.

Beim Standard k-means Verfahren [Joh92, Bis95] werden alle Datenpunkte v_t dem Clusterzentrum mit dem minimalen Abstand $d(v_t, y_k)$ zugeordnet. Ändert sich die Klassenzugehörigkeit eines Vektors, wird er also einer Klasse zugeordnet, für deren Beschreibung er nicht herangezogen worden war, dann werden die Klassenbeschreibungen der beiden beteiligten Klassen neu berechnet. Dies geschieht iterativ so lange, bis sich die Zuordnungen der Vektoren nicht mehr verändern. Mit diesem Verfahren wird die Störung D lokal minimiert. Allerdings ist der Rechenaufwand für die Neubestimmung der Klassenbeschreibungen sehr hoch. Ein weiterer Nachteil ist, dass die Reihenfolge der Daten das Ergebnis beeinflussen kann.

Hier wird deshalb die so genannte *Batch*-Variante verwendet. Sämtliche Vektoren v_t werden in einem Schritt den verschiedenen Klassen zugeordnet, erst im Anschluss daran erfolgt mit der gespeicherten Zuordnung eine Neuberechnung der Klassenbeschreibungen. Auch diese Iteration kann so oft durchgeführt werden, bis sich die Cluster nicht mehr ändern. In der Praxis bricht man das Verfahren jedoch ab, sobald die Verringerung der Störung $(D_{i-1} - D_i)/D_i$ eine vorher bestimmte Grenze ε unterschreitet.

Die Aufspaltung des Codebuchs erfolgt so lange, bis eine vorbestimmte Anzahl von Klassen erreicht ist. In diesem Fall erfolgt eine Aufteilung in 256 Klassen. Es wird ein Abstandsklassifikator berechnet, das heißt, die Abstandsfunktion zur Berechnung der Klassenzugehörigkeit ist der euklidische Abstand $d(v, y_k) = |v - \mu_k|$ zum Clusterzentrum μ_k. Als weitere, zusätzliche Klasse wird anschließend der Nullvektor dazugenommen. Von ihm ist bekannt, dass er eine besondere Rolle spielt und deshalb auf jeden Fall durch eine Klasse repräsentiert sein soll. Durch die Art der Merkmalsgewinnung sind durch den Nullvektor nämlich Leerräume markiert, die als wichtige Segmentierhilfe für die Trainingsdaten dienen können.

In einem abschließenden Schritt wird für die endgültige Beschreibung der Klassen ein Normalverteilungsklassifikator berechnet. Dazu wird eine weitere Iteration des k-means Verfahrens, diesmal aber mit dem *Mahalanobis*-Abstand $d(v, y_k) \cong \mathcal{N}(v, \mu_k, K_k)$, durchgeführt. Für die Klassenbeschreibungen muss dazu zusätzlich die Kovarianzmatrix berechnet werden. Man beschränkt sich auf eine Iteration: Zum einen sind die Rechenzeiten sehr hoch, zum anderen haben die Normalverteilungen bei mehreren Iterationen die Tendenz, sehr breit zu werden. Da für das anschließende Training der HMMs gewichtete Entscheidungen des Vektorquantisierers verwendet werden, ist dies unerwünscht.

Neben dem Codebuch für die „normalen“ Merkmalsvektoren werden auf dieselbe Weise auch Codebücher für die Δ- und ΔΔ-Vektoren erzeugt. Man hat schließlich also drei Codebücher, mit denen für jedes Eingabebild drei Datenströme erzeugt werden. Für die weitere Verarbeitung sieht man die Symbole dieser Datenströme als voneinander unabhängig an. Der Sinn dieses Vorgehens ist der folgende: Indem die Vektorquantisierung in einem dreidimensionalen Raum erfolgt, kann man mit nur 3×257 Klassenbeschreibungen effektiv 257^3 Klassen unterscheiden. Derart viele Klassen könnte man durch direkte Clusterung mit den vorhandenen Daten prinzipiell nicht erzeugen. Durch die Integration der dynamischen Information der Bildmerkmale ist der betrachtete Merkmalsraum jedoch so hochdimensional, dass eine solch feine Unterteilung notwendig ist. Die Annahme der Unabhängigkeit der verschiedenen Vektoren kann unter diesen Umständen gerechtfertigt werden.

5.4.2 Erstes Training

Die durch die Clusterung erzeugte erste Klasseneinteilung ordnet den einzelnen Klassen keine Bedeutung zu. Durch die Vektorquantisierung wird lediglich die Dimension der Eingangsdaten verringert. Dennoch können mit den quantisierten Symbolen die Buchstaben-HMMs trainiert werden, so dass man ein erstes funktionsfähiges HMM-Erkennungssystem erhält. Es ist jedoch nicht zu erwarten, dass die Erkennungsleistung dieses Systems hoch ist, da man nicht davon ausgehen kann, dass das Codebuch für die Erkennung sehr diskriminativ ist. Sein einziger Einsatz besteht auch nur darin, im nächsten Schritt eine Segmentierung der Trainingsdaten durchzuführen, die dann der Bildung eines besseren Codebuchs dient, dessen Klassen Bedeutungen zugeordnet sind. Nur für diesen beschränkten Einsatzbereich macht es Sinn, ein Erkennungssystem mit dem jetzigen Codebuch aufzubauen. Da die Segmentierung zudem mit variablen Zuordnungen arbeitet, wirken sich Fehler des Systems bei schlecht repräsentierten Wörtern nicht so stark aus. Vom System wird auch keine Generalisierung verlangt wird, da es nur auf den Daten, mit denen es selber trainiert worden ist, eingesetzt wird.

Dass ein erstes Training mit dem vorhandenen Codebuch allerdings überhaupt verwertbare Ergebnisse liefert, liegt an der Struktur der Modelle, und daran, wie sie trainiert werden. Beim Training geht es darum, die Parameter der einzelnen Buchstabenmodelle zu belehren; das geschieht mit dem *Baum-Welsh*-Training, das in Abschnitt 3.4 vorgestellt wurde. Die Trainingsmenge besteht aus ganzen Wortbildern, für die jeweils eigene Wort-HMMs aufgebaut werden. Der Forward-Backward-Algorithmus berechnet die Vorwärts- und Rückwärtswahrscheinlichkeiten für die Zustände der vollständigen Wort-HMMs. Die Erwartungswerte der Zustands- und Sequenzwahrscheinlichkeiten, aus denen in jeder Iteration die neuen Schätzwerte für die HMM-Parameter gebildet werden, werden dann auf der Ebene der einzelnen Buchstabenmodelle berechnet. Hier ist die hierarchische Implementierung der Wort-HMMs (vgl. Abbildung 5.2) von Vorteil, die es erlaubt, die Zustände des Wort-HMMs direkt den einzelnen Buchstaben-HMMs zuzuordnen.

Da die einzelnen Buchstaben links-rechts-Modelle sind und Start- und Endzustand des Wort-HMMs festgelegt sind, erfolgt durch diese Art der Modellierung bereits eine grobe Zuordnung der einzelnen Zustände zu den Symbolen. Dadurch konvergiert das Training rasch, und eine Adaption mit der unüberwachten Aufteilung der Klassen ist überhaupt erst möglich.

Da zu Beginn des Trainings kein Zusammenhang zwischen den Zuständen und den Symbolen bekannt ist, gibt es auch keinen Anhaltspunkt für eine spezielle Initialisierung der Emissionswahrscheinlichkeiten. Deshalb werden bei allen Zuständen die Emissionswahrscheinlichkeiten aller $k_{\max}$ Symbole auf den konstanten Wert $b_i(k) = \frac{1}{k_{\max}}$ gesetzt. Für die drei Codebücher c ist im speziellen Fall also $b_{i,c}(k) = \frac{1}{257}$. Die Initialisierung der Übergangswahrscheinlichkeiten erfolgt ebenfalls für alle Zustände gleich. Die Selbstübergänge werden auf den festen Wert $a_{ii} = 0.60$ festgelegt, der die durchschnittliche Verweildauer in einem Zustand repräsentiert und durch die Statistik der Wortlängen gegeben ist. Zu beachten ist, dass durch den optionalen Pausenzustand der letzte Zustand jedes Schreibvarianten-Pfades zwei echte Nachfolger hat. Die Übergangswahrscheinlichkeiten werden in diesem Fall auf identische Werte $a_{i,\text{pause}} = a_{i,\text{end}} = 0.20$ gesetzt.

Die Eingangswahrscheinlichkeiten der Startzustände werden bei allen Buchstabenmodellen auf den Wert $\pi_{i_{\text{start}}} = 1$ gesetzt. Sie werden nicht adaptiert, sondern behalten diese Werte während des Trainings bei, damit seltene Schreibvarianten nicht unterrepräsentiert werden. Im Gegensatz zu den Übergangswahrscheinlichkeiten summieren die Eingangswahrscheinlichkeiten nicht zu eins; dadurch wird eine Verfälschung der Zustandswahrscheinlichkeiten vermieden, falls verschiedene Buchstaben-HMMs unterschiedlich viele Schreibvarianten enthalten. Wenn kein *Finelabel* des Trainingsworts vorhanden ist, dann werden alle Zustandspfade der Buchstabenmodelle trainiert, ansonsten wird nur der Zustandspfad, der zu der entsprechenden Schreibvariante gehört, neu geschätzt.

5.4.3 Endgültige Bestimmung der Klassen

Mit dem Erkennungssystem, das durch das erste Training zur Verfügung steht, ist eine grobe Zuordnung der Merkmalsvektoren der Trainingsstichprobe zu den einzelnen HMM-Zuständen möglich. Sie wird für die Bestimmung der Klassen, die das endgültige Codebuch des Systems darstellen, benutzt. Durch jeden Zustand der einzelnen Buchstabenmodelle wird jeweils eine eigene Klasse definiert.

Die Bestimmung der Modellparameter im HMM-Training erfolgte über die Berechnung der Zustandswahrscheinlichkeiten der Wort-HMMs. Die Klassen werden jetzt auf dieselbe Weise geschätzt. Es werden auch hier die in Gleichung (3.16) definierten Zustandswahrscheinlichkeiten $\gamma_t(i)$ verwendet, die mit dem Forward-Backward-Training für alle Trainingswörter berechnet werden. Dies erfolgt in der letzten Iteration des HMM-Trainings. Mit diesen Wahrscheinlichkeiten werden die Erwartungswerte der jeweiligen Vektoren bestimmt. Es erfolgt also eine *variable* Segmentierung der Trainingsdaten.

Die Menge aller Zustände der Buchstaben-HMMs sei $\{s_k\}$, wobei der Index k bereits auf

die neu zu berechnenden Klassen hinweist. Wie in Abschnitt 5.1 beschrieben, verweist jedes Zeichen des Wortmodells auf ein zugeordnetes Buchstabenmodell. Damit ist auch jedem Zustand s_i des Wort-HMMs ein Zustand s_k aus der Menge der Buchstaben-HMMs zugeordnet. Die Zustände der Wort-HMMs der Trainingsmenge können also in einzelne Mengen $S_k = \{s_i \mid s_i \mapsto s_k\}$ aufgeteilt werden. Mit dieser Definition erfolgt die Bestimmung der mit den Wahrscheinlichkeiten gewichteten Mittelwertvektoren der neuen Klassen durch

$$\mu_k = E\{v \mid s_i \in S_k\} = \frac{\sum_t \sum_{s_i \in S_k} \gamma_t(i) \cdot v(t)}{\sum_t \sum_{s_i \in S_k} \gamma_t(i)}. \tag{5.9}$$

Es wird über sämtliche Beobachtungen t aller Trainingswörter summiert. Für die Modellierung der Normalverteilungen müssen auch die Kovarianzen berechnet werden, was auf die gleiche Weise geschieht:

$$K_k = E\{(v-\mu_k)(v-\mu_k)^T \mid s_i \in S_k\}. \tag{5.10}$$

Die Vektoren v_t sind zur Repräsentation der Dynamik die in Abschnitt 4.2 vorgestellten, auf die fünffache Länge erweiterten Merkmalsvektoren. Im Unterschied zur ersten Klasseneinteilung erhält man ein einziges Codebuch $C = (\mu_k, K_k)$.

5.4.4 Clusterung

Nachdem die abschließenden Klassenbeschreibungen vorliegen, wird die Gesamtanzahl der Klassen auf eine vorher vorgegebene Anzahl reduziert. Das wird realisiert, indem so lange die am nächsten beieinander liegenden Klassen zusammengefasst werden, bis die maximale Anzahl erreicht ist. Die Vorteile einer Reduktion der Klassenanzahl bestehen – neben einer Reduktion des Rechenaufwands bei der Erkennung – insbesondere in einer erhöhten Robustheit der Adaption.

Wegen der endlichen Stichprobengröße kann es nämlich vorkommen, dass während des HMM-Trainings Zustände nur selten oder gar nicht besetzt werden. Das passiert zum Beispiel bei selten auftretenden Buchstaben, besonders wenn diese für die Repräsentation vieler Schreibvarianten modelliert sind. Das System wird gegenüber solchen Vorkommnissen robust, indem jene Klassen mit Hilfe verwandter Klassen beschrieben werden. Ein weiterer Aspekt dieser Robustheit ist, dass durch die Zusammenlegung zweier Klassen für die gemeinsame neue Klasse mehr Daten zur Verfügung stehen, so dass die Parameter der neuen Klasse besser geschätzt werden können.

Das Clustern der Klassen erfolgt in drei Phasen:

- Zunächst werden alle Klassen betrachtet, deren Zustände überhaupt nicht besetzt wurden, für die also gar keine Beispielvektoren für eine Belehrung zur Verfügung stehen. Man kann diese Klassen in Ermangelung einer Beschreibung deshalb auch keiner „benachbarten“ Klasse zuordnen, mit der man sie zusammenlegen könnte. Sie werden deshalb einfach entfernt.

- Bei Klassen, deren Zustände nur selten eingenommen wurden, sind die Spalten der Kovarianzmatrix häufig linear abhängig. Die Wahrscheinlichkeit dafür sinkt erst mit der Anzahl der Trainingsbeispiele. In einem solchen Fall ist es nicht möglich, die Inverse der Kovarianzmatrix zu bestimmen, die für die Berechnung der Klassenzugehörigkeit unbedingt benötigt wird. Die betreffende Klasse wird deshalb mit derjenigen Klasse zusammengelegt, bei der der euklidische Abstand der Mittelpunkte am geringsten ist. Eine spezifischere Schätzung der Ähnlichkeit über eine Abstandsfunktion, die die Struktur des Merkmalsraums besser widerspiegelt, ist nicht möglich, da für eine solche gerade eine geeignete Kovarianzmatrix zur Verfügung stehen müsste.

- Ist die geforderte maximale Klassenanzahl noch nicht erreicht, so erfolgt die weitere Clusterung über die *Kullback-Leibler-Divergenz*. Mit ihr wird eine Abstandsmatrix für sämtliche Klassen berechnet. In mehreren Schritten werden jeweils die am nächsten beieinander liegenden Klassen zusammengelegt und die Abstandsmatrix anschließend an die neue Klassenaufteilung angepasst. Dieser Vorgang wird so lange wiederholt, bis die vorbestimmte Klassenanzahl erreicht ist.

Man bezeichnet die Kullback-Leibler-Divergenz auch als *Cross-Entropie* oder *asymmetrische Divergenz*. Mit ihr wird die Abweichung einer Modellverteilung $\tilde{P}(x)$ von der gegebenen Verteilung $P(x)$ dargestellt:

$$D_K(P,\tilde{P}) = -\int P(x)\ln\frac{\tilde{P}(x)}{P(x)}dx. \tag{5.11}$$

Die Divergenz ist asymmetrisch, weil $D_K(P,\tilde{P}) \neq D_K(\tilde{P},P)$. Es handelt sich bei der Kullback-Leibler-Divergenz also um kein echtes Distanzmaß. Es kann aber gezeigt werden, dass für alle Verteilungen $D_K(P,\tilde{P}) \geq 0$ gilt, mit Gleichheit genau dann, wenn $\tilde{P} = P$ [Bis95]. Für die Zwecke hier ist es jedoch lediglich notwendig, das Minimum der Divergenz zu bestimmen. Man addiert die konstante Entropie $-\int P(x)\ln P(x)dx$ der Verteilung, und nimmt als Maß die Cross-Entropie der beiden Verteilungen:

$$D(P,\tilde{P}) = -\int P(x)\ln\tilde{P}(x)\,dx. \tag{5.12}$$

Für die konkrete Anwendung muss die Divergenz der beiden Gaußverteilungen $\mathcal{N}_1, \mathcal{N}_2$ mit den Mittelwerten μ und Kovarianzmatrizen K berechnet werden. Das Einsetzen der Wahrscheinlichkeitsverteilungen ergibt

$$\begin{aligned}
&D(\mathcal{N}_1,\mathcal{N}_2)\\
&= -\int \mathcal{N}_1(x)\cdot\ln\mathcal{N}_2(x)\,dx &(5.13)\\
&= -\int \mathcal{N}_1(x)\cdot\left[\ln\frac{1}{\sqrt{(2\pi)^N\det K_2}} - \tfrac{1}{2}(x-\mu_2)^T K_2^{-1}(x-\mu_2)\right]dx &(5.14)\\
&= -\ln\frac{1}{\sqrt{(2\pi)^N\det K_2}} + \tfrac{1}{2}\int \mathcal{N}_1(x)\cdot\left[(x-\mu_2)^T K_2^{-1}(x-\mu_2)\right]dx. &(5.15)
\end{aligned}$$

Der Ausdruck innerhalb des Integrals lässt sich erweitern und umsortieren. Dabei wird ausgenutzt, dass die Kovarianzmatrix und ihre Inverse K_2^{-1} symmetrisch sind:

$$
\begin{aligned}
&(x-\mu_2)^T K_2^{-1}(x-\mu_2) \\
&= x^T K_2^{-1} x - 2\mu_2^T K_2^{-1} x + \mu_2^T K_2^{-1}\mu_2 && (5.16)\\
&= x^T K_2^{-1} x - 2\mu_1^T K_2^{-1} x + \mu_1^T K_2^{-1}\mu_1 \\
&\qquad + 2\mu_1^T K_2^{-1} x - \mu_1^T K_2^{-1}\mu_1 - 2\mu_2^T K_2^{-1} + \mu_2^T K_2^{-1}\mu_2 && (5.17)\\
&= (x-\mu_1)^T K_2^{-1}(x-\mu_1) \\
&\quad + 2\mu_1^T K_2^{-1} x - 2\mu_2^T K_2^{-1} x - \mu_1^T K_2^{-1}\mu_1 + \mu_2^T K_2^{-1}\mu_2 && (5.18)\\
&= \text{spur}\left[K_2^{-1}((x-\mu_1)(x-\mu_1)^T)\right] \\
&\quad + (2\mu_1^T K_2^{-1} - 2\mu_2^T K_2^{-1})\, x \\
&\quad - \mu_1^T K_2^{-1}\mu_1 + \mu_2^T K_2^{-1}\mu_2 && (5.19)
\end{aligned}
$$

Die Ausführung der Integration entspricht jetzt der Bildung der Erwartungswerte für x und $(x-\mu_1)(x-\mu_1)^T$, und hat als Ergebnis gerade den Mittelwert und die Kovarianz der Verteilung $\mathcal{N}_1$:

$$
\begin{aligned}
&\int \mathcal{N}_1(x)\cdot\left[(x-\mu_2)^T K_2^{-1}(x-\mu_2)\right] dx \\
&= \text{spur}[K_2^{-1}K_1] + \mu_1^T K_2^{-1}\mu_1 - 2\mu_1^T K_2^{-1}\mu_2 + \mu_2^T K_2^{-1}\mu_2 && (5.20)\\
&= \text{spur}[K_2^{-1}K_1] + (\mu_1-\mu_2)^T K_2^{-1}(\mu_1-\mu_2). && (5.21)
\end{aligned}
$$

Die Divergenz zweier Normalverteilungen lässt sich also durch den Ausdruck

$$D = \tfrac{1}{2}N\ln(2\pi) + \det K_2 + \tfrac{1}{2}\,\text{spur}(K_2^{-1}K_1) + \tfrac{1}{2}\,(\mu_1-\mu_2)^T K_2^{-1}(\mu_1-\mu_2) \qquad (5.22)$$

berechnen. Für die Bestimmung der Abstandsmatrix wird die Kullback-Leibler-Divergenz symmetrisiert, da keine der Verteilungen gegenüber der anderen ausgezeichnet ist:

$$D = D(\mathcal{N}_1,\mathcal{N}_2) + D(\mathcal{N}_2,\mathcal{N}_1). \qquad (5.23)$$

Dieser Abstand dient als Ähnlichkeitskriterium für das Zusammenlegen der Klassen. Dass die Clusterung der Klassen auf diese Weise überhaupt funktioniert, liegt an den Annahmen, die man über die Charakteristik der Schrift machen kann: Die einzelnen HMM-Zustände repräsentieren Teile verschiedener Buchstaben; die Ähnlichkeiten, die zwischen den Buchstaben existieren, werden bei dem eingesetzten Verfahren ausgenutzt.

Die Anzahl der Klassen des neuen Codebuchs ist vorgegeben und steht nach der Clusterung in keinem direkten Zusammenhang mehr mit der Anzahl der HMM-Zustände, aus denen die Klassen zunächst gebildet wurden. Für die Initialisierung der Parameter im anschließenden HMM-Training ist es aber notwendig, den Zusammenhang zwischen den Codebuchklassen und HMM-Zuständen zu kennen. Aus diesem Grund wird bei der Clusterung darüber Buch geführt, welche Klassen zusammengelegt wurden. Es wird eine Tabelle angelegt, in der zu jedem Zustand das zugehörige Modell verzeichnet ist.

5.4.5 Endgültiges Training

Mit dem neu berechneten Codebuch liegt ein Satz von Verteilungsfunktionen vor, mit denen die Emissionsdichten der Buchstaben-HMMs so modelliert werden können, dass sie die Beobachtungen realistisch beschreiben. Die HMM-Parameter müssen dafür erneut von Grund auf belehrt werden. Das Vorgehen entspricht ziemlich genau dem beim ersten HMM-Training.

Einen Unterschied gibt es aber bei der Initialisierung der HMM-Parameter. Im Vergleich zum ersten Training besitzt man nun Informationen über die Bedeutung der einzelnen Modelle. Den einzelnen HMM-Zuständen s_i sind direkt Modelle k zugeordnet, wobei eine solche Zuordnung $k(s_i)$ auch nach der Clusterung der Modelle noch besteht. Es macht Sinn, die Wahrscheinlichkeit für die Emission des entsprechenden Modells auf einen hohen Startwert, zum Beispiel $P_{\text{init}} = 0.60$, zu setzen:

$$b_i(k) = \begin{cases} P_{\text{init}} & \text{für } k = k(s_i), \\ \frac{1}{k_{\max}-1}(1-P_{\text{init}}) & \text{sonst.} \end{cases} \tag{5.24}$$

Die Eingangs- und Übergangswahrscheinlichkeiten werden dagegen auf dieselbe Weise wie im ersten Training initialisiert.

Eine Besonderheit ist die *Glättung der Parameter*. Es gibt eine untere Grenze für die Emissionswahrscheinlichkeiten. Wird diese Grenze im Training bei der Bildung des Erwartungswertes unterschritten, dann wird die Emissionswahrscheinlichkeit der betroffenen Klasse auf die festgelegte untere Grenze angehoben. Auch bei der Fuzzy-Vektorquantisierung werden nämlich viele Klassen, die weit von der am besten bewerteten Klasse entfernt liegen, mit der Wahrscheinlichkeit Null bewertet. Um zu verhindern, dass wegen eines einzigen Merkmalsvektors, der ungünstig klassifiziert wird, die Wahrscheinlichkeit einer vollständigen Zustandsfolge auf Null gesetzt wird, haben sämtliche Klassen eine von Null verschiedene Emissionswahrscheinlichkeit. Die Größenordnung der unteren Grenze liegt bei etwa 10^{-5}.

Zusätzlich zu den eigentlichen Buchstaben-HMMs werden die so genannten *Joker*-Modelle trainiert, die in Abschnitt 5.1 beschrieben wurden. Sie modellieren die Obermenge sämtlicher Buchstaben mit all ihren Schreibweisen. Sie werden in einem zweiten, separaten Trainingsschritt mit denselben Trainingsdaten adaptiert. Das *Label* der Trainingswörter besteht in diesem Fall aber aus einer Folge von Jokerzeichen. Die Länge dieser Sequenz ergibt sich durch die Länge des eigentlichen *Label*. Auch bei *Joker*-Modellen werden verschiedene Schreibvarianten bei der Adaption unterschieden: Liegt ein *Finelabel* vor, so wird nur der zugehörige Zustandspfad der jeweiligen Modelle neu geschätzt.

Nach diesem HMM-Training sind die Parameter aller Buchstaben-HMMs an die Trainingsmenge optimal angepasst. Es liegt eine vollständige Beschreibung des Schriftmodells vor, mit dem ein Handschrifterkennungssystem aufgebaut werden kann.

5.5 Visualisierung

Es ist nützlich, Buchstabenmodelle visualisieren zu können: Anhand von Bildern können die Modellierung und das Ergebnis des Parametertrainings plausibel gemacht werden, und es können Schwachstellen in der Arbeitsweise der beschriebenen Algorithmen aufgedeckt werden. Das gilt besonders im Hinblick auf die im weiteren zu entwickelnde Bestimmung der Modelltopologie.

Es wurde eine einfache Methode entwickelt, lineare links-rechts-HMMs zu visualisieren. Sie berechnet das Bild nicht direkt und ausschließlich aus den Modellparametern, sondern benutzt als Hilfsmittel klassifizierte Trainingsdaten, um Bilder der abstrakten Modelle zu erzeugen. Auf diese Weise wird die Schwierigkeit umgangen, dass eine Umkehrung der Merkmalsextraktion (Abschnitt 4.2) realisiert werden muss, um aus den Klassenbeschreibungen Bilder zu generieren. Trotz dieser Vereinfachung erweist sich die Methode als geeignetes Werkzeug, um Beschränkungen der Schriftmodelle und der Adaptionsalgorithmen aufzudecken.

5.5.1 Visualisierung der Klassen

Der semikontinuierlichen Struktur der HMMs entsprechend wird die Visualisierung in zwei Schritten durchgeführt. Zuerst werden Visualisierungen der Codebuchklassen berechnet, bevor die so generierten Bilder der Klassen zu Bildern der Modellzustände kombiniert werden.

Da die durch das Codebuch repräsentierten Merkmalsvektoren von geometrischen Strukturen des Konturbildes berechnet werden (Abschnitt 4.2), ist es schwierig, durch Rücktransformation ein Bild der Klasse zu erhalten. Stattdessen werden Trainingsdaten für die Generierung der Bilder verwendet. Jeder Vektor der Trainingsdaten wird klassifiziert, und die ursprünglichen Segmente innerhalb der Fensterausschnitte werden in das Bild der Klasse gerastert, mit der Klassenwahrscheinlichkeit als Wert für die Transparenz. Für jede Klasse wird so ein Rasterbild mit Grauwerten erstellt. Als Ergebnis erhält man recht klare Bilder, da nur positive Beispiele in die Visualisierung eingehen. Eine schlechte Klasseneinteilung wird sich nur darin zeigen, dass die Visualisierung sehr verschwommen ist.

Ergebnisse für Ziffern im System für deutsche Postleitzahlen werden in Abbildung 5.4 gezeigt. Jeder vertikale Streifen entspricht einer einzelnen Klasse. Wegen der besonderen Art der Klassenberechnung der semikontinuierliche Modelle, die in Abschnitt 5.4.3 beschrieben wurde, zeigen bereits die Klassen erkennbar alle Eigenschaften der Schrift. Man sieht deutlich die einzelnen Ziffern und erkennt auch die Modellierung mit drei Zuständen. Auffällig ist, dass die Ziffer „2“ nur durch zwei Klassen gebildet wird. Grund dafür ist die Clusterung (Abschnitt 5.4.4). Die den ersten Zustand der Ziffer „2“ repräsentierende Klasse wurde mit anderen Klassen zusammengelegt.

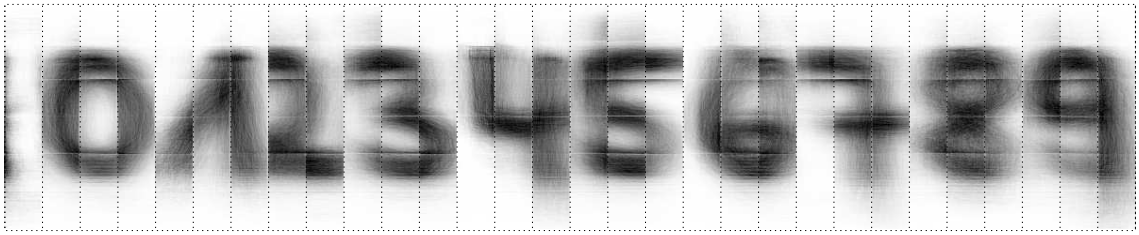

Abbildung 5.4: Visualisierung der Klassen eines Erkennungssystems für deutsche Postleitzahlen. Man erkennt ein Streifenmuster; jeder Streifen stellt die Visualisierung einer Klasse dar. Durch die besonderen Art der Klassenberechnung werden bereits durch das Nebeneinanderstellen der Klassenbilder lesbare Zeichen erzeugt.

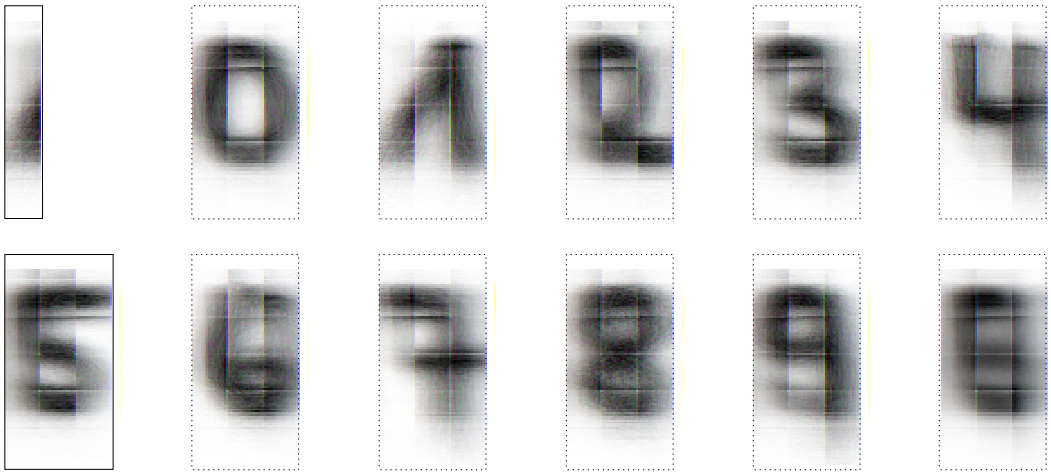

Abbildung 5.5: Visualisierung der Modelle eines Powerscript-Erkenners für deutsche Postleitzahlen. Das Modell links oben repräsentiert die *Pause*, einen Leerraum, rechts unten ist das *Joker*- oder *Garbage*-Modell.

5.5.2 Visualisierung der Modelle

Im zweiten Schritt werden die Bilder, die Klassen repräsentieren, zu Bildern von Zuständen kombiniert; dazu werden die Grauwerte der Pixel addiert, durch die Emissionswahrscheinlichkeiten der Klassen gewichtet, und normiert. Um ein Bild des vollständigen Modells zu erhalten, werden die Bilder der aufeinanderfolgenden Zustände von links nach rechts angeordnet; eine Modellierung der Zustandsbreite findet nicht statt.

Bei der Visualisierung für das System deutscher Postleitzahlen in Abbildung 5.5 sind alle Modelle gut erkennbar. Das schmale Modell links oben repräsentiert das „Pause“ Modell; das Modell rechts unten ist das *Joker*- oder *Garbage*-Modell, das die Menge aller vorkommenden Zeichen repräsentiert und für die Berechnung der Ergebnisgüte (Abschnitt 6.3) benutzt wird. Interessant ist die Auswirkung der Clusterung auf die Modellbildung. Die bei der Clusterung entfernte Klasse, die aus dem ersten Zustand der Ziffer „2“ generiert wurde (Vgl. Abbildung 5.4), wurde offensichtlich mit den Klassen der Ziffern „5“ und „9“ zusam-

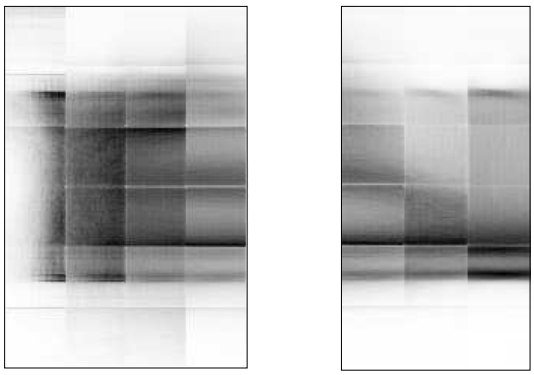

Abbildung 5.6: Visualisierung der Modelle „B“ und „u“ im Erkennungssystem für Städtenamen der CEDAR Datenbank. Der Modell für „u“ repräsentiert die rechte Seite des Buchstabens „B“, da die Trainingsdaten durch den Städtenamen „Buffalo“ dominiert werden.

mengelegt. Das Bild des Modells zeigt, dass das Ergebnis durchaus plausibel ist.

5.5.3 Analyse von Modellbildern

Die Visualisierung der Buchstabenmodelle erwies sich zum Beispiel bei der Analyse des Erkennungssystems für die CEDAR Datenbank (siehe Abschnitt 6.4.1) als sehr nützlich. Es zeigte sich nach der Adaption der Parameter ein unerwartetes Bild für den Buchstaben „u“, wie in Abbildung 5.6 zu sehen ist. Der Grund dafür ist, dass die Trainingsdaten durch den Städtenamen „Buffalo“ dominiert ist, so dass in 62 Prozent aller Fälle der Buchstabe „u“ im Kontext von „Bu“ erscheint. Der Trainingsalgorithmus kann dieses Ungleichgewicht nicht ausgleichen, da er auf der Grundlage unsegmentierter Trainingsdaten arbeitet. Die einzig mögliche Lösung des Problems besteht also darin, die Auswahl der Trainingsdaten zu korrigieren.

Kapitel 6

Erkennung

Um aus dem im letzten Abschnitt entwickelten Schriftmodell ein funktionsfähiges Erkennungssystem aufzubauen, sind einige weitere Überlegungen notwendig. Als wesentlicher Aspekt wird das Augenmerk auf den Rechenaufwand für die Erkennung gerichtet sein. Bei Verfahren zur Beschleunigung der Erkennung ist man allerdings meist gezwungen, zwischen höherer Geschwindigkeit und schlechteren Erkennungsraten abzuwägen. Die Rechenzeit bestimmt jedoch letztlich den möglichen Einsatzbereich des Systems. Dies betrifft insbesondere die mögliche Größe der verwendeten Wörterbücher.

Es wird meist mit dynamischen Lexika gearbeitet, da so auf einfach konfigurierbare Weise Vorwissen in die Erkennung eingebracht werden kann. Ein Punkt der weiteren Überlegungen betrifft die Frage, wie aus diesen Lexika konkrete HMMs aufzubauen sind, damit die Erkennung effizient durchgeführt werden kann. Hierzu muss zwischen dem Aufwand für den Aufbau der Struktur und dem für die eigentliche Berechnung abgewogen werden.

Schließlich ist es eine nicht triviale Fragestellung, was überhaupt das Ergebnis der Erkennung sein soll. Generell wird verlangt, die Resultate mit *Glaubwürdigkeiten* zu versehen. Durch die dynamischen Lexika zeigt sich aber die Problematik, dass die Glaubwürdigkeiten vom jeweiligen Wörterbuch abhängig sind. Es wird ein Verfahren vorgestellt, das dieser Eigenschaft Rechnung trägt und sich in der Praxis bewährt hat.

6.1 Merkmalsklassifikation

Die Klassifikation der Merkmalsvektoren ist mit sehr hohen Rechenaufwand verbunden. Für die benötigten Rechenschritte multiplizieren sich die Anzahl der Merkmalsvektoren mit der hohen Dimension des Merkmalsraums und den vielen unterschiedlichen Klassen. Bei der Erkennung isolierter Einzelwörter beträgt die durchschnittliche Anzahl der Merkmalsvektoren etwa 60, sie bestehen – wegen der Berücksichtigung der Dynamik – aus 100 Komponenten, und die Anzahl der Klassen beträgt etwa 200.

Das größte Beschleunigungspotential besteht in der Reduktion der Merkmalsdimension, da sie sich quadratisch auf die Anzahl der notwendigen Rechenschritte auswirkt. Für die

Berechnung der notwendigen Transformation auf weniger Dimensionen wird eine *Lineare Diskriminanzanalyse* (LDA) durchgeführt [Hae92]. Des weiteren kann man das Problem der vielen Klassen, von denen der größte Teil ohnehin nur mit sehr niedrigen Wahrscheinlichkeiten bewertet wird, durch Berechnung des Klassifikators in mehreren Stufen lösen.

6.1.1 Lineare Diskriminanzanalyse

Eine Lineare Diskriminanzanalyse wird durchgeführt, um eine Transformation des Merkmalsraums auf weniger Dimensionen zu erreichen und so die Klassifikation zu beschleunigen. Es geht dabei um die Auswahl der „wichtigen“ Merkmale, in denen die klassenspezifische Information enthalten ist. Das ähnelt dem Vorgehen bei einer Hauptachsentransformation, nur dass Verteilungen von mehreren Klassen vorhanden sind, was explizit zu berücksichtigen ist: Durch die Transformation sollte die Trennbarkeit der Klassen nicht verschlechtert werden.

Man hat bei der Auswahl der Transformation zwei verschiedene Ziele: Zum einen sollen die Varianzen der einzelnen, klassenspezifischen Verteilungen minimiert werden, zum anderen soll der mittlere Abstand der Klassen maximal sein. Diese Anforderungen werden durch die Definition von zwei *Streuungsmatrizen* formalisiert. Es wird die *Intra-Klassen*-Streuungsmatrix S_{intra} eingeführt, die allgemein die klassenspezifischen Verteilungen charakterisiert, und die *Inter-Klassen*-Streuungsmatrix S_{inter}, mit der die Raumverteilung der Klassen selber dargestellt wird:

$$S_{\text{intra}} = E((v_k - \mu_k)(v_k - \mu_k)^T) \tag{6.1}$$

$$S_{\text{inter}} = E((\mu_k - \mu)(\mu_k - \mu)^T). \tag{6.2}$$

Die Erwartungswerte werden in beiden Fällen über sämtliche Vektoren v der mit Klassenzugehörigkeiten gekennzeichneten Stichprobe gebildet. Die Zuordnung eines Vektors zu einer Klasse k ist durch das Hinzufügen eines Index v_k angezeigt, entsprechend bezeichnet μ_k den Mittelwertvektor der zu v_k gehörenden Klasse. Der Mittelwert sämtlicher Vektoren ist μ. Man beachte, dass die Summe der beiden Streuungsmatrizen gerade die Kovarianzmatrix der Verteilung sämtlicher Vektoren darstellt.

Es wird nun eine lineare Transformation $\Phi : v \mapsto \tilde{v}$ gesucht, die v in einen m-dimensionalen Unterraum abbildet. Die Abbildung wird durch eine $m \times n$ Matrix, $m < n$, dargestellt. Da die Streuungsmatrizen S die Form von Kovarianzmatrizen haben, ist eine Darstellung von S im neuen Koordinatensystem durch $\tilde{S} = \Phi^T S\, \Phi$ gegeben. Weil die Determinante der Kovarianzmatrix als Maß für die Breite von Verteilungen genommen werden kann, besteht eine geeignetes Kriterium in der Maximierung von

$$\det(\tilde{S}_{\text{intra}}^{-1}\, \tilde{S}_{\text{inter}}) = \frac{\det(\tilde{S}_{\text{inter}})}{\det(\tilde{S}_{\text{intra}})} = \frac{\det(\Phi^T S_{\text{inter}}\, \Phi)}{\det(\Phi^T S_{\text{intra}}\, \Phi)} \to \max. \tag{6.3}$$

Die Klassen sind dann wie gefordert weit im Merkmalsraum gestreut, während die Verteilung der Klassen selber kompakt ist.

Die Bestimmung des Maximums erfolgt über die Lösung der Eigenwertaufgabe

$$S_{\text{intra}}^{-1} S_{\text{inter}} \Phi_i = \lambda \Phi_i. \tag{6.4}$$

[Fuk90]. Die Eigenvektoren Φ_i stellen die Spalten einer $n \times n$ Transformationsmatrix dar. Wie bei einer Karhunen-Loève Transformation werden die Eigenwerte nach ihrer Größe sortiert. Die zu den m größten Eigenwerten gehörenden Vektoren bilden die Matrix für die gesuchte optimale Abbildung Φ.

Die Lineare Diskriminanzanalyse wird bereits während des Trainings, nach der Clusterung des endgültigen Codebuchs (vgl. Abschnitt 5.4.4), durchgeführt. Die Lösung der Eigenwertaufgabe erfolgt mit dem Jacobi-Verfahren [Bro87]. An dieser Stelle wird auch gleich die Darstellung der Klassenbeschreibungen in den neuen Merkmalsraum transformiert. Die Länge der Merkmalsvektoren wird durch die Transformation von 100 auf 32 reduziert. Die Reduktion der Merkmalsdimensionen und die damit einhergehende direkte Verringerung des Rechenaufwands ist aber nur ein Ergebnis der Transformation. Ein weiterer positiver Aspekt besteht darin, dass die Elemente der Merkmalsvektoren nun nach ihrer Relevanz für die Klassifikation sortiert sind. Diese Eigenschaft kann im nächsten Schritt für die weitere Beschleunigung der Klassifikation ausgenutzt werden.

6.1.2 Mehrstufige Klassifikation

Ein weiterer Grund für den großen Rechenaufwand bei der Klassifikation der Merkmalsvektoren liegt in der hohen Anzahl der Klassen. Bei typischen Anwendungen werden bis zu 200 Klassen unterschieden. Obwohl die Vektorquantisierung fuzzy erfolgt, also jede der Klassen mit einer Bewertung versehen wird, ist nur für einige wenige Klassen die Wahrscheinlichkeit faktisch von Null verschieden. In der Praxis handelt es sich meist um etwa zehn, in Einzelfällen um bis zu dreißig Klassen. Die Klassifizierungsergebnisse aller übrigen Klassen gehen nicht in die weiteren Berechnungen ein. Es ist anzustreben, solche Klassen frühzeitig zu identifizieren, um die Durchführung von Berechnungen, deren Ergebnisse nicht benutzt werden, zu vermeiden. Dies wird realisiert, indem die Klassifikation in mehreren Stufen durchgeführt wird.

Die Klassifikation des Vektors v erfolgt in diesem Zusammenhang über die Berechnung der logarithmierten Wahrscheinlichkeit zu jeder Klasse k:

$$\begin{aligned} p_k(v) &= \ln \mathcal{N}_k(v) \\ &= -\tfrac{1}{2} N \ln 2\pi - \tfrac{1}{2} \ln(\det K_k) - \tfrac{1}{2}(v-\mu_k)^T K_k^{-1}(v-\mu_k) \\ &= -\tfrac{1}{2} N \ln 2\pi - \tfrac{1}{2} \ln(\det K_k) - \tfrac{1}{2}(\mu_k^T K_k^{-1}\mu_k) + (\mu_k^T K_k^{-1})v - \tfrac{1}{2} v^T K_k^{-1} v \\ &= a_{0k} + a_k^T v - \tfrac{1}{2} v^T K_k^{-1} v. \end{aligned} \tag{6.5}$$

Die für die Klassen konstanten Werte

$$a_{0k} = -\tfrac{1}{2} N \ln 2\pi - \tfrac{1}{2} \ln(\det K_k) - \tfrac{1}{2}(\mu_k^T K_k^{-1}\mu_k) \tag{6.6}$$

und

$$a_k = K_k^{-1}\mu_k \tag{6.7}$$

können schon vor der eigentlichen Klassifikation berechnet werden. Wertet man als Rechenschritt eine Multiplikation mit anschließender Addition, so beträgt bei einer Dimension d des Merkmalsvektors die Anzahl der für jede Klasse notwendigen Rechenschritte $d + \frac{1}{2}d(d+1) = \frac{1}{2}d(d+3)$. Zur Herleitung dieses Ausdrucks wird die Symmetrie der Kovarianzmatrix ausgenutzt. Der Rechenaufwand wächst quadratisch mit der Dimension des Merkmalsraums.

Man macht sich nun die Eigenschaft zunutze, dass die Elemente der Merkmalsvektoren durch die LDA-Transformation nach der Relevanz für die Klassifikation sortiert sind. Das Verfahren besteht darin, die Klassifikation mit verschiedenen Ansatzlängen s durchzuführen. Die Ansatzlänge entspricht der Anzahl der Komponenten des Merkmalsvektors, die bei der Klassifikation berücksichtigt werden. Alle weiteren Elemente des Vektors werden ignoriert und auf Null gesetzt. Man betrachtet also den Vektor mit reduzierter Ansatzlänge $v^s = (v_1, \ldots, v_s, 0, \ldots, 0)$, der die gleiche Dimension wie der ursprüngliche Merkmalsvektor v besitzt. Für die Klassifikation dieses Vektors sind nur $\frac{1}{2}s(s+3)$ Rechenschritte notwendig.

In mehreren Stufen wird die Klassifikation mit wachsenden Ansatzlängen durchgeführt, wobei in jeder Stufe die Klassen mit niedrigen Bewertungen von den nachfolgenden Berechnungen ausgeschlossen werden. Für die konkrete Durchführung muss eine feste Anzahl von Stufen i vorgegeben werden, die durch Ansatzlängen s_i und Schwellen δ_i definiert sind. Zu Beginn werden sämtliche Klassen aktiviert. In jeder Berechnungsstufe i werden dann die folgenden Schritte ausgeführt:

- Berechne die logarithmierten Wahrscheinlichkeiten aller aktiven Klassen für den Vektor v^{s_i} mit der verkürzten Ansatzlänge s_i. Die Ergebnisse der letzten Stufe dienen bei der Berechnung als Ausgangspunkt, so dass nur noch $\frac{1}{2}(s_i - s_{i-1})(s_i + s_{i-1} + 3)$ zusätzliche Rechenoperationen durchzuführen sind.

- Bestimme für die aktuelle Ansatzlänge diejenige Klasse $k_{\max}$, die die höchste logarithmierte Wahrscheinlichkeit $p_k(v^{s_i})$ aller aktiven Klassen besitzt.

- Entferne alle Klassen k, bei denen die Differenz zur Wahrscheinlichkeit der besten Klasse über der gesetzten Schwelle $p_{k_{\max}}(v^{s_i}) - p_k(v^{s_i}) > \delta_i$ liegt, aus der Menge der aktiven Klassen.

In der letzten Stufe werden die Wahrscheinlichkeiten aller noch aktiven Klassen vollständig berechnet. Für diese Klassen erhält man den genauen Wert für p, alle anderen Klassen werden verworfen und nicht weiter beachtet. Als Ergebnis der Klassifikation werden ausschließlich die auf eins normierten Klassenwahrscheinlichkeiten

$$P_k(v) = \frac{\exp p_k(v)}{\sum_k \exp p_k(v)} \tag{6.8}$$

dieser Klassen verwendet, alle anderen Klassenwahrscheinlichkeiten sind Null. Dabei kann es passieren, dass Klassen, obwohl sie bei vollständiger Klassifikation ein gutes Ergebnis erreichen würden, durch dieses Verfahren fälschlicherweise nicht berücksichtigt werden. Es ist deshalb eine sorgsame Auswahl der Stufen und Schwellwerte nötig, um einen guten Kompromiss zwischen Reduktion und Fehlerrate zu finden.

6.1.3 Berechnung der Schwellen

Zur Berechnung der Reduktionsparameter wird eine Trainingsstichprobe mit typischen Merkmalsvektoren benötigt. Es kann sich um die selben Daten handeln, die für die übrige Adaption verwendet werden. Man möchte die Ansatzlängen mit den zugehörigen Schwellwerten finden, bei denen möglichst wenig Fehler auftreten und trotzdem viel Rechenzeit eingespart werden kann. Als Fehler zählt die Entfernung jeder Klasse, die im Rahmen einer normalen, einstufigen Klassifikation mit einer relevanten Wahrscheinlichkeit bewertet werden würde. Die Suche nach geeigneten Ansatzlängen für eine Reduktion ist eine Art empirische Analyse der Struktur des transformierten Merkmalsraums.

Für jeden Vektor der Stichprobe wird eine einstufige Klassifikation ohne Reduktion durchgeführt. Deren Ergebnisse werden als Referenz für die Güte der Schwellen verwendet: Aufgrund der Klassifikation werden die Klassen in relevante Klassen k^+ und nicht zu berücksichtigenden Klassen k^- eingeteilt. Die relevanten Klassen zeichnen sich dadurch aus, dass bei ihnen die normierte Wahrscheinlichkeit (6.8) oberhalb einer gesetzten Schwelle, zum Beispiel $P_{k^+}(v) > 10^{-4}$, liegt. Die Parameter sind möglichst so zu wählen, dass diese Klassen bei der mehrstufigen Klassifikation vollständig berechnet werden. Als nächstes wird dann die Klassifikation für sämtliche Ansatzlängen durchgeführt. Dabei werden die jeweiligen Zwischenergebnisse in einer Matrix abgelegt. Für jede Ansatzlänge und jede Klasse wird vermerkt, wie groß der Abstand zu der Klasse ist, die in diesem Schritt am besten bewertet wurde. Dies gibt gerade die Schwelle an, ab der die Klasse bei der mehrstufigen Klassifikation entfernt werden würde.

Mit diesen beiden Ergebnissen werden nun Histogramme aufgebaut, aus denen die besten Schwellen ermittelt werden. Für die Klassen aus k^+ und k^- werden jeweils eigene Histogramme erstellt. Die Histogramme sind zweidimensional: Für die verschiedenen Ansatzlängen s wird jeweils gezählt, wie häufig der Abstand jeder Klasse zum Maximum in einem bestimmten Intervall δ liegt. Man unterteilt dazu den Bereich für die Abstände in etwa 40 Intervalle mit der Breite eins.

Zur Bestimmung einer optimalen Kombination von Ansatzlänge und Schwellwert muss ein Gütekriterium bestimmt werden. Dazu lassen sich aus dem Histogramm verschiedene Werte ablesen:

- Jeder Kombination aus Ansatzlänge und Schwelle lässt sich eine Reduktionsrate zuordnen, die mit $f_{\text{red}}(s,\delta)$ bezeichnet wird. Dies geschieht durch einfaches Auszählen des Histogramms für k^-.

- Auf die gleiche Weise lässt sich durch Auszählen des Histogramms für k^+ eine Fehlerrate $f_{\text{err}}(s,\delta)$ bestimmen.
- Unabhängig von den Histogrammen lässt sich für jede Ansatzlänge s der eingesparte Rechenaufwand pro Klasse bestimmen. Er ist proportional zur Anzahl der Rechenoperationen und beträgt $n_{\text{op}}(s) = \frac{1}{2}(s_{\text{end}} - s)(s_{\text{end}} + s + 3)$.

Mit diesen drei Werten wird eine Bewertung für die Güte der Reduktion definiert, durch deren Maximierung die Stufenparameter bestimmt werden. Im vorliegenden Fall setzt man jedoch als feste Bedingung eine maximale Fehlerrate $f_{\text{err}}(s,\delta) < 1\%$ und wählt als Gütekriterium die absolute Einsparung an Rechenzeit $g = f_{\text{red}}(s,\delta) \cdot n_{\text{op}}(s)$. Damit ist eine optimale Kombination aus Ansatzlänge und Schwellwert definiert. Für die Berechnung weiterer Schwellen wird das beschriebene Verfahren für größere Ansatzlängen wiederholt. Mit fünf Stufen werden in der Praxis noch gute Ergebnisse in der Erkennungsleistung erzielt, bei einer Reduktion der Rechenzeit um etwa einen Faktor zehn.

6.2 Viterbi-Berechnung

Neben dem Wortbild wird die Erkennungsaufgabe des Systems durch ein Wörterbuch, also eine Liste mit allen gültigen Wörtern, definiert. In Abschnitt 5.1 wurde gezeigt, wie mit den einzelnen Buchstaben-HMMs ganze Wort-HMMs aufgebaut werden, mit denen anschließend durch den Viterbi-Algorithmus die Wörter klassifiziert werden. Allerdings ist es sehr rechenintensiv, sämtliche Wörter einzeln zu bewerten und erst im Anschluss daran die Ergebnisse zu vergleichen. HMMs werden deshalb für ganze Wörterbücher aufgebaut. Dadurch wird es möglich, viele Doppelberechnungen zu vermeiden. Man muss sich aber noch einige Gedanken über die optimale Struktur dieser Wörterbuch-HMMs machen, und darüber, wie man die Ergebnisse des Viterbi-Algorithmus für die Auswertung der Erkennung verwendet.

Neben der direkten Einsparung von Rechenschritten hat die Erkennung auf ganzen Wörterbuch-HMMs noch einen anderen Vorteil. Dadurch, dass die Klassifikation für alle Wörter parallel erfolgt, ist es möglich, schon während der Viterbi-Berechnung unwahrscheinliche Alternativen von der Berechnung auszuschließen. Durch dieses *Pruning* kann die Erkennung noch einmal drastisch beschleunigt werden.

6.2.1 Wörterbuch-Trie

Eine effiziente Struktur für die Repräsentation von Wörterbüchern ist der so genannte *Trie*. Es handelt sich um eine Baumstruktur, bei der jeder Knoten einen Buchstaben repräsentiert (Abbildung 6.1). Die gemeinsame Präfixe verschiedener Wörter werden zusammengefasst. Zur Vermeidung von Mehrdeutigkeiten ist jedes Wortende mit einem separaten Wortende-Knoten gekennzeichnet. Der Begriff Trie leitet sich von *retrieval* ab [Knu98], da mit Tries ein schneller Zugriff auf Datenbasen möglich ist.

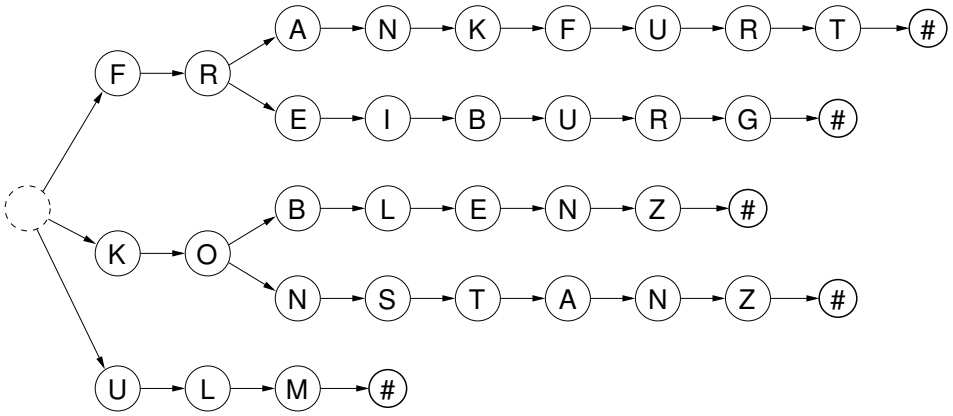

Abbildung 6.1: Beispiel für einen Wörterbuch-Trie: Durch die Baumstruktur werden die Wortanfänge zusammengefasst. Nach der Viterbi-Berechnung finden sich die Wortwahrscheinlichkeiten an den Endknoten, die mit „#“ eindeutig gekennzeichnet sind.

Der Aufbau des Trie-HMM erfolgt geradlinig. Ausgehend von einem einzelnen Wurzelknoten werden die verschiedenen Wörter nacheinander in den Trie eingefügt. Zunächst wird der Wortanfang bestimmt, der schon im Trie repräsentiert ist, dann werden die Buchstaben-HMMs für die restlichen Buchstaben an der entsprechenden Stelle eingefügt. Die Absorptionszustände der Buchstaben am Wortende erhalten dabei eine besondere Kennzeichnung. Eine notwendige Überlegung betrifft die Übergangswahrscheinlichkeiten an den Verzweigungen: Sie werden sämtlich auf eins gesetzt, da die Bewertungen der Zustandsfolgen vom gemeinsamen Wurzelknoten bis zu den Endknoten von den übrigen Wörterbucheinträgen des Tries jeweils unabhängig sind.

Die Viterbi-Berechnung auf dem so erzeugten Trie unterscheidet sich in keiner Weise von der Berechnung bei einzelnen Wort-HMMs. Es muss nur beachtet werden, dass es mehrere mögliche Endzustände gibt. Jeder der markierten Absorptionszustände enthält nach Abschluss der Berechnung die Wahrscheinlichkeit der besten Zustandsfolge für das entsprechende Wort-HMM. Aus den Endknoten lassen sich also direkt die Erkennungsergebnisse extrahieren.

6.2.2 Wörterbuch-Automaten

Bei sehr großen Wörterbüchern mit einer sehr dichten Belegung der Einträge ist ein Trie keine sehr effiziente Struktur zur Repräsentation mehr. Man denke zum Beispiel an die Menge aller Ziffernfolgen der Länge fünf. Eine Trie-Repräsentation würde dafür $\sum_i 10^i \approx 10^5$ Knoten benötigen. Es ist in diesem Fall sinnvoller, einen vollständigen Automaten zu verwenden, bei dem die Knoten nicht nur gemeinsame Vorgänger, sondern auch gemeinsame Nachfolger haben (Abbildung 6.2). Man kommt in diesem Fall mit $5 \times 10 = 50$ Knoten aus.

Allerdings ist es jetzt nicht mehr so einfach wie bei einem Trie, das Erkennungsergebnis zu bestimmen, da es keine eindeutigen Endknoten für Wörterbucheinträge mehr gibt. Das Ergebnis muss in einem zweiten Schritt des Viterbi-Algorithmus, der Rückverfolgung (vgl. Ab-

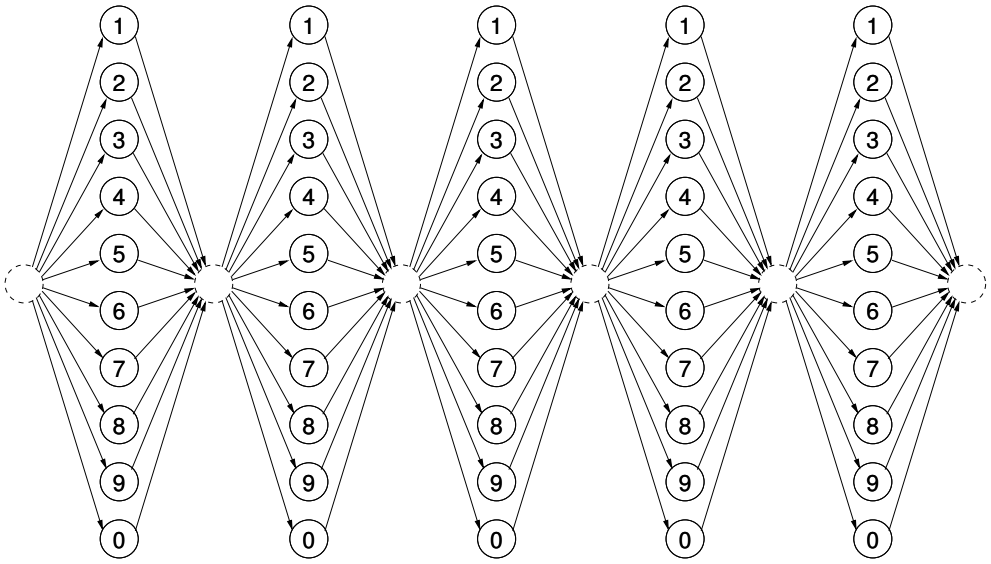

Abbildung 6.2: Beispiel für einen Wörterbuchautomaten: Der dargestellte Automat erzeugt sämtliche fünfstelligen Ziffernfolgen. Durch eine Viterbi-Berechnung mit anschließender Baumsuche im Rückverfolgungsschritt können diese Ziffernfolgen klassifiziert werden. In einem zweiten Schritt erfolgt der Abgleich mit einer Liste gültiger Zahlen. Auf diese Weise können zum Beispiel deutsche Postleitzahlen erkannt werden.

schnitt 3.3.3), bestimmt werden. Bei der Rückverfolgung wird die optimale Zustandsfolge offengelegt, aus der sich das Ergebnis rekonstruieren lässt. Für diesen Schritt muss jedoch der vollständige Trellis im Speicher gehalten werden. Eine Erkennung auf Automaten ist deshalb nur möglich, wenn durch die Repräsentation das Wörterbuch stark komprimiert wird.

Meist ist für die Erkennung zusätzlich gefordert, dass mehrere Alternativen aus dem Wörterbuch bewertet werden. Es werden auch die suboptimalen Zustandsfolgen gesucht, deren Zustände zu unterschiedlichen Buchstabenmodellen gehören und somit andere Wörter repräsentieren. Es wird deshalb eine Variante des N-best Viterbi-Algorithmus [Pur94] implementiert. Aus Laufzeitgründen beschränkt man sich auf einen Algorithmus, der nur Näherungen für die Ergebnisse liefert, was aber in der Praxis ausreicht. Das Verfahren führt eine Baumsuche auf dem Trellis aus. Dazu werden bei der Vorwärtsberechnung des Viterbi-Algorithmus neben den Wahrscheinlichkeiten jeweils auch Listen aller konkurrierenden Vorgängerzustände, sortiert nach Wahrscheinlichkeit, im Trellis mit abgelegt. Die Näherung besteht darin, dass diese Verzeigerung nur für die Absorptionszustände, also die Zustände zwischen den einzelnen Buchstabenmodellen, angelegt wird. Nur an diesen Verzweigungsstellen werden tatsächlich alternative Wörter generiert, und gleichzeitig wird die Größe des zu untersuchenden Baums auf diese Weise deutlich eingeschränkt. Man extrahiert alle Zustandsfolgen, die durch diese Rückwärts-Verzeigerung erzeugt werden, und sortiert sie nach ihrer Wahrscheinlichkeit. Diese Liste stellt durch die Zuordnung der Zustände zu den Buchstabenmodellen direkt die Klassifikation alternativer Ergebniswörter dar. Aufgrund der Näherung erhält man für eine Folge von Buchstabenmodellen zwar nicht unbedingt die optimale Zustandsfolge und damit die korrekte Wahrscheinlichkeit, in der Praxis ist die Qualität

der Ergebnisse jedoch ausreichend. Durch *Pruning* wird das Verfahren weiter beschleunigt: Während der Baumsuche wird die Differenz zur Wahrscheinlichkeit der besten Zustandsfolge aufsummiert. Überschreitet sie einen bestimmten Wert, so wird die weitere Berechnung an dem betreffenden Zweig abgebrochen.

Meist ist es nicht möglich, ausgehend von einer durch die Anwendung vorgegebenen Wortliste direkt einen geeigneten Automaten aufzubauen, dessen Knotenanzahl so gering ist, dass sich der erhöhte Berechnungsbedarf auszahlt. Ein Beispiel dafür ist die Liste gültiger deutscher Postleitzahlen, die aus etwa 27500 fünfstelligen Ziffernfolgen besteht. Die Repräsentation des Lexikons findet in diesem Fall auf zwei Stufen statt. Auf einem vereinfachten Automaten, der durch eine Erweiterung des Wörterbuchs erzeugt wird, erfolgt mit dem hier vorgestellten Algorithmus die Berechnung der Wahrscheinlichkeiten. Im Fall deutscher Postleitzahlen sind dies alle fünfstelligen Ziffernfolgen. Der Abgleich mit der Liste tatsächlich gültiger Postleitzahlen erfolgt dann erst beim abschließenden Erstellen der Ergebnisliste.

6.2.3 Pruning

Eine Methode zur Beschleunigung des Viterbi-Algorithmus besteht darin, nur die Erfolg versprechenden Zustandspfade bei der Berechnung zu verfolgen, da das Ergebnis ohnehin ausschließlich durch die beste Zustandsfolge gegeben ist. Dies geschieht durch die Einführung von „aktiven“ Zuständen und die dadurch realisierbare Möglichkeit des *Pruning*.

Die zu klassifizierenden Wortmodelle sind *links-rechts*-HMMs (vgl. Abschnitt 3.1). Dies bringt mit sich, dass zu Beginn der Berechnung nur der erste Zustand eine von Null verschiedene Wahrscheinlichkeit hat, und damit $\delta_1(i) = 0, \forall i \neq 1$ gilt. Dementsprechend macht es keinen Sinn, die Iterationsvorschrift (3.20) für andere Zustände als den ersten und seine direkten Nachfolger durchzuführen. Dies gilt in entsprechender Weise auch für die folgenden Berechnungsschritte. Man führt deshalb eine Liste von aktiven Zuständen ein, die gerade die Zustände enthält, für die die Viterbi-Berechnung im nächsten Schritt durchgeführt werden soll. In jedem Schritt werden dann die direkten Nachfolger aller bereits aktiven Zustände zu der Liste hinzugefügt. Bereits diese Maßnahme bewirkt eine deutliche Abnahme des Rechenaufwands, da insbesondere bei der Trie-Struktur der Großteil aller Knoten erst spät im Lauf der Berechnung aktiviert wird. In der Implementierung ersetzt man das Konzept der aktiven Zustände durch aktiven Knoten, da diese wegen der geringeren Anzahl einfacher und damit schneller zu verwalten sind.

Das Konzept der aktiven Zustände dient auch zur Realisierung des Pruning. Es bedeutet wörtlich „beschneiden“ und versinnbildlicht damit das Entfernen von Zweigen aus dem Trie. In jedem Schritt wird die maximale Wahrscheinlichkeit aller Zustände ermittelt. Jeder Zustand, dessen Wahrscheinlichkeit diesen Wert um einen bestimmten Faktor – den so genannten *Pruningfaktor* – unterschreitet, wird aus der Liste der aktiven Zustände entfernt. Es ist damit vorerst von weiteren Berechnungen ausgeschlossen, bis er durch einen Vorgängerzustand eventuell erneut aktiviert wird. Damit wird zwei verschiedenen Situationen Rechnung

getragen: Die eine spiegelt direkt die Namensgebung des Pruning wider. Wenig Erfolg versprechende Wortalternativen werden abgeschnitten und nicht weiter berechnet. Die andere betrifft die Zustände am Anfang des Wort-HMMs, die zum Ende der Beobachtungsfolge hin nicht mehr zur Maximumsbildung der gesamten Zustandsfolge beitragen können. Dies ist die analoge Situation wie bei den noch nicht aktiven Zuständen zu Beginn der Beobachtungen. Aufgrund der asymmetrischen Struktur der Viterbi-Berechnung, die sich in der Wahl einer bestimmten Richtung zur Berechnung zeigt, werden diese Zustände aber nicht deterministisch deaktiviert, sondern durch Pruning von der weiteren Berechnung ausgeschlossen.

Die Bestimmung der konkreten Pruningschwelle erfolgt anhand der Erkennungsleistung des Systems auf einer Teststichprobe. Es wird die kleinstmögliche Pruningschwelle gewählt, bei der sich die Fehlerrate des Systems nicht um mehr als einen festgelegten Faktor erhöht. Die Bestimmung anhand der Erkennungsrate ist deshalb notwendig, da die optimale Pruningschwelle stark von den Wörterbüchern abhängt. Es ist deshalb bei der Bestimmung der Pruningschwelle besonders auf die Auswahl der Tests zu achten, die eine typische Anwendung simulieren sollten. Anzumerken ist, dass durch eine geschickte Wahl der Pruningschwelle teilweise auch eine Verbesserung der Ergebnisse eintreten kann. Dies tritt dann auf, wenn Wörter aus der Wortliste entfernt werden, die eine korrekte Erkennung verhindern. Die Größenordnung solcher Verbesserungen ist jedoch meist nicht statistisch relevant.

6.2.4 Visualisierung des Trellis

Zur Analyse der dynamischen Vorgänge bei der Viterbi-Berechnung wurde ein Tool entwickelt, das interaktiv, über die Wahl des Merkmalsfensters, den Viterbi-Trellis visualisiert. Im Gegensatz zu Tools, die den Trellis für ein einzelnes Wortmodell darstellen, wie zum Beispiel in [Auf97] beschrieben, wird der Trellis für den gesamten Wörterbuch-Trie oder Automaten dargestellt. In Abbildung 6.3 werden am Beispiel der Erkennung des Wortbildes *„Konstanz“* zwei Momentaufnahmen gezeigt, in denen die Aktivierung der Modellzustände gut ersichtlich ist. Mit dem Tool ist es einfach möglich, das Pruning-Verhalten und die Wahl des Pruningfaktors zu kontrollieren.

Anhand des Viterbi-Trellis lässt sich jedem Fenster t des Wortbildes der wahrscheinlichste Zustand q_t^* aus der Viterbi-Zustandssequenz (3.26) zuordnen. Damit ist es möglich, das Wortbild in Einzelzeichen zu segmentieren. Neben der eigentlichen Erkennung ist dies ein zusätzliches Leistungsmerkmal des Worterkenners. Abbildung 6.4 zeigt die Segmentierung anhand einiger Wortbilder aus verschiedenen Projekten.

6.3 Glaubwürdigkeitsmaß

Die Viterbi-Berechnung liefert für jeden Eintrag des Wörterbuchs eine Bewertung durch die Wahrscheinlichkeit P_{seq} der am besten passenden Zustandsfolge. Nach diesem Maß lassen sich die Wörter sortieren. Das am höchsten bewertete Wort gilt dann als das Ergebnis a_1

Abbildung 6.3: Visualisierung des Trellis: Für die Erkennungs des Wortbildes *„Konstanz“* wird ein Fenster von links nach rechts über das Bild geschoben. Gezeigt werden zwei Momentaufnahmen des Erkennungsprozesses, bei denen sich das Fenster am Ende der Buchstaben „o“ bzw. „a“ befindet. Unter den Wortbildern ist jeweils der Trie eines Wörterbuchs mit sechs Einträgen, in dessen quadratisch dargestellten Knoten die Zustandsstruktur der Buchstaben gezeigt wird. Die Färbung der Zustände zeigt den Aktivierungsgrad der HMM-Zustände, errechnet aus dem Viterbi-Trellis.

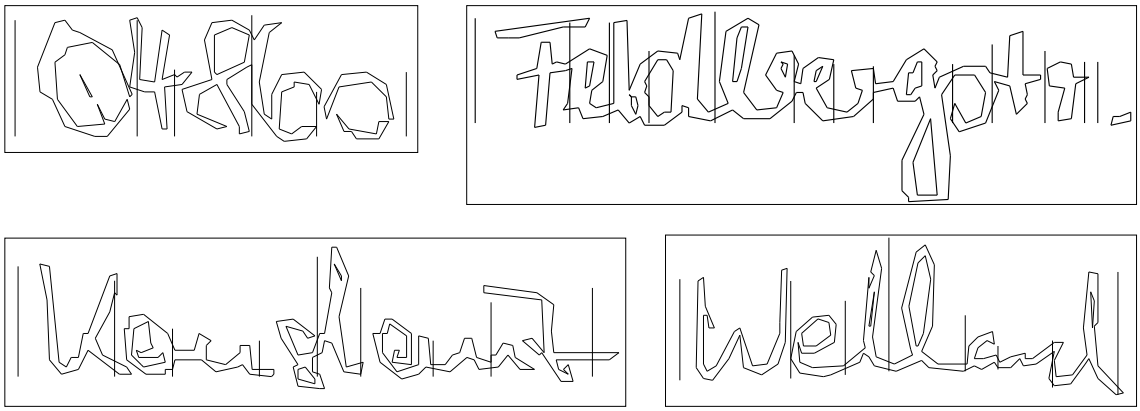

Abbildung 6.4: Zeichensegmentierung: Die Viterbi-Zustandssequenz ordnet jedem Fenster des Wortbildes einen HMM-Zustand zu und segmentiert so das Bild in einzelne Zeichen. Gezeigt werden Beispiele für die Projekte Deutschland Postleitzahlen („*04860*“), Straßennamen („*Feldbergstr.*“) und Ortsnamen („*Konstanz*“), sowie Kanada Ortsnamen („*Welland*“).

der Erkennung, die nachfolgenden Wörter stellen mögliche Alternativen $a_{i\in\{2,3,\ldots\}}$ dar. Den ermittelten Wahrscheinlichkeitswert $P_{\text{seq}}(a_i)$ kann der Benutzer des Systems dann zumindest als Anhaltspunkt für die Relevanz dieser Alternativen ansehen.

Man kann jedoch nicht davon ausgehen, dass das System in jedem Fall perfekt arbeitet. Für den Fall, dass das System nicht in der Lage ist, das Wortbild einem Wort zuzuweisen, gibt es deshalb die Möglichkeit der *Rückweisung*. Zurückgewiesen wird zum Beispiel immer, wenn während der Berechnung Fehler – etwa durch Überschreitung von erlaubten Zahlenbereichen – auftreten. Von Bedeutung ist insbesondere die Rückweisung von schwer oder gar nicht zu entziffernden Wortbildern. Ein effektives Kriterium für deren Identifikation stellt die Bewertung der so genannten *Nicht-Wörter* dar. Nicht-Wörter sind Wörter, die vollständig aus Joker-Modellen (Abschnitt 5.1) bestehen, und die bei jeder Erkennung mitberechnet werden. Es werden jeweils mehrere Nicht-Wörter erzeugt. Ihre Länge ist für unterschiedliche Wortbilder verschieden und richtet sich nach der Anzahl der Merkmalsvektoren. Für die weitere Auswertung betrachtet man nur die Wahrscheinlichkeit des am besten bewerteten Nicht-Wortes $P_{\text{seq}}(a_{\text{non}})$. Ist diese größer als die Wahrscheinlichkeit der besten Alternative $P_{\text{seq}}(a_1)$, so wird das Wortbild zurückgewiesen. Um aber in jedem Fall eine Bewertung der Güte des Wortbildes zu haben, werden Nicht-Wörter bei der Berechnung vom Pruning ausgenommen.

Neben der Möglichkeit einer Rückweisung wird für den Fall, dass eine Erkennung durchgeführt werden kann, eine Bewertung gewünscht, wie glaubwürdig das Ergebnis dieser Erkennung ist. Gesucht ist ein Glaubwürdigkeitsmaß P_{cred}, das die Wahrscheinlichkeit angibt, dass es sich bei der gelieferten besten Alternative tatsächlich um das korrekte Ergebnis handelt. Dieses Maß muss also die Statistik der Erkennungsgüte des Systems widerspiegeln. Es wird anhand einer Teststichprobe ermittelt. Für die Berechnung der Glaubwürdigkeit werden verschiedene Parameter herangezogen [Glo97]:

- Um die Wahrscheinlichkeitswerte bei verschiedenen Wortbildern vergleichbar zu machen, wird die logarithmierte Wahrscheinlichkeit des Zustandspfades mit der Anzahl der Merkmalsvektoren normiert.

$$p_{\text{seq}}(a_i) = \ln P_{\text{seq}}(a_i) \;/\; \text{Anzahl Vektoren} \tag{6.9}$$

- Als wichtiges Kriterium hat sich der Abstand zum besten Nicht-Wort herausgestellt. Ist der Abstand groß, so kann man davon ausgehen, dass das Wort deutlich erkannt worden ist.

$$(\Delta p)_{\text{non}}(a_i) = p_{\text{seq}}(a_i) - p_{\text{seq}}(a_{\text{non}}) \tag{6.10}$$

- Nur für das beste Ergebnis lässt sich der Abstand zur zweiten Alternative berechnen. Dieses Kriterium trägt der Begebenheit Rechnung, dass Wortverwechslungen eine häufige Ursache für eine falsche Erkennung sind.

$$(\Delta p)_{\text{2nd}} = p_{\text{seq}}(a_1) - p_{\text{seq}}(a_2) \tag{6.11}$$

Für jeden dieser drei Parameter werden anhand der Teststichprobe zwei Histogramme aufgebaut, in denen jeweils die Anzahl der richtig und falsch erkannten Wortbilder gezählt werden. Als falsche Erkennung wird jedes Bild gezählt, dessen Label nicht als erste Alternative bewertet wird. Anhand dieser Histogramme lässt sich durch einfaches Auszählen für jeden einzelnen Parameter eine Zuordnung der Wertebereiche zu Glaubwürdigkeiten des Ergebnisses erstellen. Für die Berechnung eines einzigen Glaubwürdigkeitswerts $P_{\text{cred}}(a_1)$ nimmt man an, dass die drei Parameter voneinander unabhängig und gleich bedeutend sind, und mittelt ihre Glaubwürdigkeiten.

Die Glaubwürdigkeiten der weiteren Alternativen $P_{\text{cred}}(a_{i\in\{2,3,\ldots\}})$ leiten sich von der Glaubwürdigkeit des besten Ergebnisses ab. Es ist der gleiche Wert, der aber dem Verhältnis der normierten Zustandswahrscheinlichkeiten entsprechend reduziert ist, also $P_{\text{cred}}(a_i) = P_{\text{cred}}(a_1) \cdot \exp(p_{\text{seq}}(a_i) - p_{\text{seq}}(a_1))$. Man beachte, dass die Glaubwürdigkeiten der Alternativen nicht zu eins summieren.

Für die Bewertung der Qualität eines Erkennungssystems ist das Verhältnis von Erkennungs- und Fehlerrate ausschlaggebend. Mit dem vorgestellten Glaubwürdigkeitsmaß kann eine Schwelle vorgegeben werden, unterhalb der das Wortbild zurückgewiesen wird. Dadurch ist es möglich, die Leistung des Systems an die jeweilige Anwendung anzupassen. Meistens ist eine feste Fehlerrate vorgegeben, die nicht überschritten werden darf. Erstaunlich ist, dass sich mit den hier vorgestellten Kriterien und ihrer recht einfachen Auswertung für die Praxis brauchbare Ergebnisse erzielen lassen.

Land	Alphabet	Modelle/ Zustände	Trainingsdaten Testdaten	Anzahl	Wörterbuch
Arabische Emirate	arabisch	32/101	Emiratsnamen	17050	108
			Emiratsnamen	971	108
Kanada	alphanumerisch	41/246	Adresselemente	19443	195
			Ortsnamen	1029	109
			Straßennamen	857	100
Deutschland	alphabetisch	34/264	Adresselemente	21726	196
			Ortsnamen	1047	100
			Straßennamen	1007	200
Deutschland	numerisch	10/35	Postleitzahlen	5793	27540
			Postleitzahlen	580	27540
USA	alphanumerisch	39/324	Adresselemente	19438	150
			Ortsnamen	1143	100
			Straßennamen	998	400
			Staaten	995	234
USA	numerisch	11/36	ZIP Codes	22088	43392
			ZIP Codes	2210	43392
USA (CEDAR-DB)	alphabetisch	27/167	Adresselemente	9565	100
			Ortsnamen	532	∅ 107
			Staaten	470	∅ 18
USA (CEDAR-DB)	numerisch	11/36	ZIP Codes	9019	43392
			ZIP Codes	435	43392

Tabelle 6.1: Trainings- und Testdaten: Es werden Konfigurationen für acht Projekte aus vier Ländern untersucht. Die Trainingsdaten haben jeweils etwa den zehnfachen Umfang der Testdaten. Getestet wird mit künstlich generierten Wörterbüchern, die typische Anwendungsszenarien simulieren.

6.4 Erkennungsleistung

Powerscript wurde für den Einsatz in der Postautomatisierung entwickelt, deshalb handelt es sich bei den Trainings- und Testdaten in dieser Arbeit ausschließlich um Wortbilder aus gescannten Adressdaten. Zur Verfügung stehen Adressbilder aus den Vereinigten Staaten, Kanada, Deutschland und den Vereinigten Arabischen Emiraten. Zusätzlich wurden Daten der Datenbank des *Center of Excellence for Document Analysis and Recognition* (CEDAR) verwendet, die in [Hul94] beschrieben wird. Es werden acht verschiedene Konfigurationen untersucht, die in Tabelle 6.1 spezifiziert sind.

6.4.1 Konfigurationen

Die Konfigurationen unterscheiden sich hauptsächlich im verwendeten Alphabet. Getestet werden Erkenner für lateinische und arabische Schrift sowie für Ziffern. Für die Erkennung numerischer Straßennamen in Nordamerika werden alphanumerische Erkenner benötigt. Die Anzahl der modellierten Zeichen ist aus der Tabelle ersichtlich und entspricht dem verwendeten Alphabet, das je nach Projekt um relevante Sonderzeichen erweitert wird. Auch die Anzahl der Modellzustände wird vorgegeben: Pro Modell werden eine bis vier Schreibvarianten mit je drei Zuständen modelliert; es werden zwischen 36 und 324 Zustände verwendet. Die Anzahl der Klassen ist abhängig von der Anzahl der Zustände und liegt durch die Clusterung (vgl. Abschnitt 5.4.4) etwa 10 bis 20 Prozent unter diesen Werten.

Mit jeder Konfigurationen werden ein oder mehrere Tests durchgeführt, da zum Teil dieselbe Konfiguration für verschiedene Aufgaben verwendet wird. In diesem Fall werden die Erkenner mit Daten aller relevanten Adresselemente, also Orts-, Straßen- und Staatennamen, gemeinsam trainiert, aber einzeln getestet. Für Training und Test stehen jeweils etwa 10000 bis 20000 Wortbilder zur Verfügung, die im Verhältnis zehn zu eins aufgeteilt werden. Getestet wird die Erkennungsleistung mit künstlich generierten Wörterbüchern, deren Größe und Zusammensetzung typischen Anwendungsfällen entsprechen. In der CEDAR-Datenbank ist jedem einzelnen Testbild ein eigenes Wörterbuch zugeordnet, wie es in einer konkreten Situation von einem Adressinterpretationssystem erzeugt worden war. Dadurch schwankt die Größe des Wörterbuchs, und es kann nur der Durchschnittswert angegeben werden. Der Einsatz des Worterkennungssystems wird durch die Dynamik besonders realistisch simuliert.

Für eine Bewertung der Erkennungsraten muss die Größe des Wörterbuchs berücksichtigt werden. Das gilt besonders für die Erkennung von ZIP-Codes und Postleitzahlen, die mit deutlich größeren Wörterbüchern getestet werden als die restlichen Adresselemente. Ihre Erkennung erfolgt zweistufig, wie in Abschnitt 6.2.2 beschrieben wird. Zunächst wird ein Automat aufgebaut, der alle Ziffernfolgen zulässt, anschließend werden die Ergebnisse mit dem Wörterbuch aller gültigen Codes, deren Anzahl in der Tabelle angegeben ist, gefiltert. Bei den USA-Daten ist zu beachten, dass der Hauptteil der ZIP-Codes in den Testdaten fünfstellig ist, ein Anteil von 12,7% aber neun Stellen aufweist, bei denen die letzten vier Ziffern nicht abgeglichen werden. In diesen Fällen ist das reale Wörterbuch somit um den Faktor 10000 größer.

6.4.2 Erkennungsraten

Die Erkennungsleistungen der Konfigurationen werden in Tabelle 6.2 gezeigt. Es sind sowohl für die Trainings-, als auch die Testdaten Erkennungsraten angegeben, damit ersichtlich ist, wie gut die Systeme generalisieren. Einen deutlichen Abfall in der Fähigkeit zur Generalisierung sieht man bei der Erkennung der arabischen Emiratsnamen. Dies ist darauf zurückzuführen, dass in Ermangelung ausreichender Stichproben die Trainingsdaten aus künstlich degradierten Maschinenschriften bestehen, die Testdaten aber aus real gescannten

Land	Trainingsdaten Testdaten	Erkennungsrate „forced"	 1% Fehler	Zeichen	Tests gemittelt
Arabische Emirate	Emiratsnamen	93,1 %	57,7 %	99,0 %	
	Emiratsnamen	76,9 %	28,7 %	97,3 %	76,9 %
Kanada	Adresseelemente	94,2 %	71,2 %	99,5 %	
	Ortsnamen	91,0 %	81,0 %	99,0 %	
	Straßennamen	96,3 %	93,0 %	99,9 %	93,4 %
Deutschland	Adresseelemente	92,0 %	75,1 %	99,3 %	
	Ortsnamen	92,0 %	75,2 %	98,7 %	
	Straßennamen	93,6 %	82,0 %	99,7 %	92,8 %
Deutschland	Postleitzahlen	73,6 %	0,0 %	91,8 %	
	Postleitzahlen	69,1 %	0,0 %	90,8 %	69,1 %
USA	Adresseelemente	94,3 %	87,2 %	99,4 %	
	Ortsnamen	91,8 %	78,9 %	99,0 %	
	Straßennamen	95,9 %	89,6 %	99,7 %	
	Staaten	54,3 %	3,5 %	83,2 %	81,2 %
USA	ZIP Codes	59,9 %	5,0 %	88,7 %	
	ZIP Codes	60,8 %	4,8 %	88,9 %	60,8 %
USA (CEDAR-DB)	Adresseelemente	83,3 %	54,7 %	97,3 %	
	Ortsnamen	85,3 %	31,0 %	97,9 %	
	Staaten	82,9 %	32,2 %	96,3 %	84,2 %
USA (CEDAR-DB)	ZIP Codes	56,6 %	10,9 %	86,4 %	
	ZIP Codes	58,2 %	11,7 %	85,9 %	58,2 %
Gesamt					77,1 %

Tabelle 6.2: Erkennungsraten der untersuchten Konfigurationen: In der ersten Zeile jedes Projekts ist die Erkennungsraten der Trainingsdaten, darunter die der Testdaten angegeben. Als Vergleichsmaßstab für die weiteren Untersuchungen dienen die gemittelten Erkennungsraten über alle Tests in den Spalten rechts.

Wortbildern.

Es sind zwei Erkennungsraten angegeben. Zum einen die reine Erkennungsrate, die in der Spalte „*forced*“ eingetragen ist, und bei der unabhängig von der Erkennungsgüte ein Ergebnis „erzwungen“ wird. Bei ihr ist die Fehlerrate komplementär zur Erkennungsrate. Zum anderen wird die Erkennungsrate bei maximal einem Prozent Fehler angegeben. Dies wird erreicht, indem Erkennungsergebnisse, deren Glaubwürdigkeit (vgl. Abschnitt 6.3) eine bestimmte Schwelle unterschreitet, zurückgewiesen werden. Daneben wird die Erkennungsrate bezogen auf die Erkennung der einzelnen Zeichen angegeben. Die außerordentlich guten Ergebnisse werden hauptsächlich durch den Einfluss von Kontextwissen aus dem Wörterbuch erzeugt. Da die Ziffernerkenner wegen der größeren Wörterbücher auf weniger Kontextwissen zurückgreifen können, erscheint ihre Erkennungsleistung geringer.

In der Spalte rechts sind die über sämtliche Tests gemittelten Erkennungsraten angege-

ben. Im Durchschnitt wird eine Erkennungsrate von 77,1 Prozent erreicht. Dieser Wert dient als Ausgangspunkt für die Bewertung der weiteren Untersuchungen.

Kapitel 7

Zusammenfassung Teil I

In diesem ersten Teil der Arbeit wurde ein vollständiges System zur Handschrifterkennung vorgestellt, das bei Siemens Dematic entwickelt wurde und mit Erfolg in den Postverteilzentren vieler Länder eingesetzt wird. Es wird die Theorie der Hidden-Markov-Modelle vorgestellt, und das System mit Bildverarbeitung, der Entwicklung des Schriftmodells und den Besonderheiten bei der Erkennung detailliert beschrieben. Die gezeigten Erkennungsraten ermöglichen den erfolgreichen Einsatz in Adresslesesystemen. Dennoch ist die Erkennungsleistung – gerade bei großen Wörterbüchern – noch verbesserungsfähig und stellt eine Herausforderung für weitere Entwicklungen dar.

Als Ansatz für Verbesserungen wird in dieser Arbeit ein System zur automatischen Bestimmung der Modellstruktur entwickelt, das im folgenden zweiten Teil der Arbeit vorgestellt wird. Zur Realisierung des neuen Systems war es notwendig, das bestehende System für die Flexibilisierung der festen Strukturen vorzubereiten. Verschiedene Arbeiten wurden dazu durchgeführt:

- Das System wurde für die Verarbeitung von variablen Modellstrukturen angepasst, und die bestehenden Trainingsalgorithmen wurden für einen flexiblen, modularen Einsatz im Rahmen der neuen Algorithmen vorbereitet. Die bis dahin in der Implementierung getrennten Systeme für Training und Erkennung wurden integriert. Dies konnte nur durch eine vollständige Neuimplementierung der Trainingsalgorithmen erreicht werden.

  Als „Abfallprodukt“ erhielt man aber ein Erkennungssystem, das in einfacher Weise – auch während des Betriebs – nachtrainiert werden kann. Auch das Trainingssystem besitzt zahlreiche neue Merkmale, zum Beispiel die Möglichkeit der Wahl zwischen Viterbi- und Forward-Backward-Training (Abschnitt 3.4.4) oder die automatische Berechnung der Schwellen für die mehrstufige Klassifikatorberechnung (Abschnitt 6.1.3). Auch ist es möglich, verschiedene Berechnungen des Erkennungssystems, zum Beispiel die Segmentierung in Einzelzeichen und die Likelihood dieser Einzelzeichen, auch im Trainingssystem zu verwenden.

- Zur Beherrschung der sich durch die folgenden Arbeiten weiter erhöhenden Systemkomplexität wurden zahlreiche Tools entwickelt. Mit Visualisierungstools ist es nun möglich, die Ergebnisse der Bildverarbeitung (Abschnitt 4.1), die Struktur der Buchstabenmodelle (Abschnitt 5.2), Klassenbilder (Abschnitt 5.5.1), Modellbilder (Abschnitt 5.5.2) und – über den Trellis (Abschnitt 6.2.4) – den Fortgang der Viterbi-Berechnung zu betrachten. Trainings- und Testtools erlauben die automatisierte Durchführung von Modellanpassung und Auswertung für mehrere Projekte und Konfigurationen.

- Im Rahmen der Bearbeitung von Projekten wurden zahlreiche Verbesserungen des Systems durchgeführt, die nicht in direkter Verbindung mit der Bestimmung der Modellstruktur stehen. Zum Beispiel wurden die Erkennung auf Automaten (Abschnitt 6.2.2) und eine Teilworterkennung implementiert. Arbeiten zur Erkennung von Arabischer Schrift führten unter anderem auch zu einer Erhöhung der Robustheit des Systems. Durch die Berechnung des Klassifikators mit dem SIMD-Befehlssatz (*Single instruction, multiple data*) des Prozessors [Ban00] wurde eine Beschleunigung von Training und Erkennung um etwa den Faktor zwei erreicht.

Mit diesen Arbeiten aus dem ersten Teil der Arbeit ist ein System entstanden, dass eine solide Basis für die automatische Modellierung der Schriftstruktur bietet, die in den folgenden Kapiteln entwickelt werden wird.

Teil II

Automatische Modellierung von Handschrift

Kapitel 8

Automatische Modellierung

Das in den vorigen Kapiteln vorgestellte System zur Handschrifterkennung beruht auf erprobten Standardtechniken: Die Schriftzeichen werden durch HMMs mit vorgegebener Struktur modelliert, deren freie Parameter mit dem bewährten Baum-Welsh-Algorithmus durch alleinige Vorgabe von Schriftproben belehrt werden. Mit dem Viterbi-Algorithmus können dann neue, dem System bis dahin unbekannte Wörter erkannt werden. Es wird eine Erkennungsleistung erzielt, die zum praktischen Einsatz in Adresslesesystemen befähigt. Man stößt mit diesem System allerdings auch an Grenzen:

- Jede Anpassung an spezielle äußere Anforderungen, zum Beispiel länderspezifische Eigenheiten der Schrift, erfordert hohen Entwicklungsaufwand. Trotz des rein statistischen Ansatzes reicht es meist nicht aus, lediglich neue, zusätzliche Schriftproben bereitzustellen, um das System an neue Schriften oder Schreibweisen anzupassen. Stattdessen muss über die Vorgabe der HMM-Topologie Wissen über die Struktur der Schrift, also über die Art und Komplexität der verschiedenen Schreibweisen, eingebracht werden,

- Die verwendete Bildrepräsentation und die Modellierung der Buchstaben beeinflussen sich gegenseitig. Schon eine Änderung der Fensterbreite für die Bestimmung der Merkmale kann die Qualität der Modellierung verschlechtern. Es ist deshalb meist nicht möglich, eine andere Vorverarbeitung oder Merkmalsgenerierung zu verwenden, ohne manuelle Eingriffe am System vornehmen zu müssen. Gerade die Wahl der Bildmerkmale ist für die Leistung des Systems aber von großer Bedeutung.

- Die Modelltopologie beeinflusst selber die Erkennungsleistung des Systems. Durch die manuelle Vorgabe der Topologie ist nicht garantiert, dass das volle Potential des Erkennungssystems ausgeschöpft wird.

Eine naheliegende Methode, die genannten Probleme zu lösen, besteht darin, das der Schrift zugrundeliegende Modell zu erweitern. Als zusätzlicher Freiheitsgrad wird die *Wahl der Modelltopologie* in die Adaption des Systems mit aufgenommen (Abbildung 8.1). Sie soll,

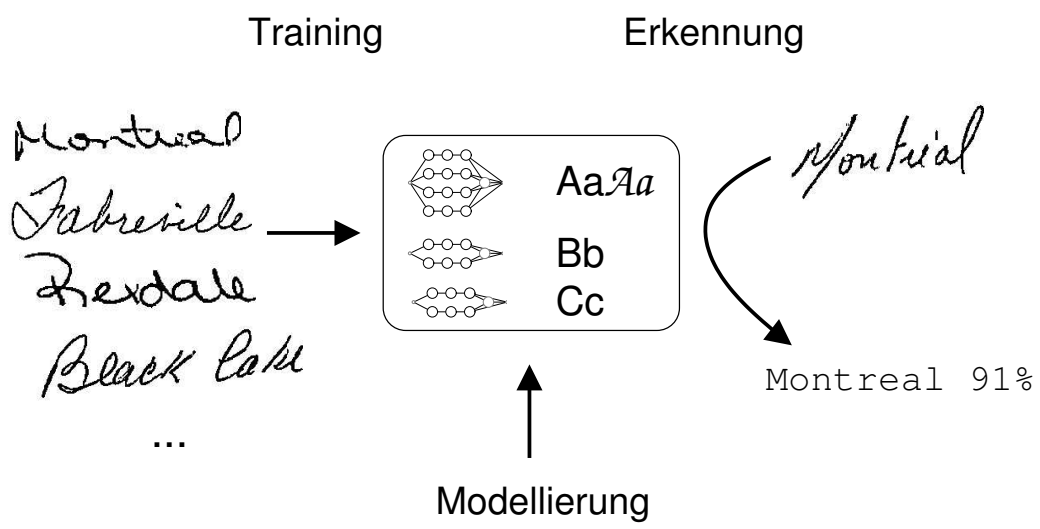

Abbildung 8.1: Modellierung im Erkennungssystem. Viele Erkennungssysteme arbeiten in zwei Schritten: In einer Trainingsphase werden Modellparameter gelernt, mit denen neue Daten erkannt werden können. Die eigentliche Modellierung erfolgt *vor* diesen beiden Schritten. Sie zu automatisieren ist die Idee für diese Arbeit.

wie die Berechnung der HMM-Parameter, rein statistisch, nur aufgrund von Schriftproben bestimmt werden.

8.1 Entwicklung des Schriftmodells

Bei der Erkennung gebundener Handschrift mit HMMs wurden schon viele unterschiedliche Modelltopologien untersucht [Ste99]. Die Entscheidung zu Gunsten einer bestimmten Struktur wird meist nicht automatisch, sondern aufgrund von Anschauung getroffen. Die Ursache dafür liegt vermutlich in der direkten visuellen Kontrolle, die man über die Struktur von Schrift besitzt, und die eine manuelle Modellierung überhaupt möglich macht. Im Gegensatz dazu wurde in der Spracherkennung, bei der eine intuitive visuelle Kontrolle nicht direkt möglich ist, eine automatische Clusterung von Aussprachevarianten schon sehr früh eingesetzt [Rab89a].

Auch in dieser Arbeit wird anschauliches Wissen über Schrift genutzt werden und die Struktur der Modellierung nicht vollständig freigegeben. Dies schlägt sich in einer konkreten Vorgabe der Modellstruktur nieder, die aber wesentliche Freiheitsgrade enthält. Sie baut auf dem in Kapitel 5 vorgestellten Schriftmodell auf und wird im folgenden genau beschrieben.

Aus den Freiheitsgraden der Modellstruktur ergeben sich Auswirkungen auf die Strategien zur Optimierung. Der vorgestellte Algorithmus wird zwei Aufgaben zu erfüllen haben: Die Clusterung der Buchstaben in verschiedene Schreibweisen und die Bestimmung der optimalen Topologie für die unterschiedlichen Allographmodelle. Von beiden Aufgaben erhofft man sich Verbesserungen für die Erkennungsleistung des Systems. Neben der schwierigen Aufgabe, bereits handoptimierte Systeme zu verbessern, kann sich ein Verfahren aber auch

dadurch bewähren, Strukturen in noch nicht analysierten Daten zu entdecken, etwa bei unbekannten Schriften oder Schreibweisen. Es kann interessante Einblicke in die Struktur der Schrift selber liefern.

8.2 Modellstruktur

Zunächst erscheint es nicht vorteilhaft, eine automatisch zu bestimmende Struktur bereits apriori einzuschränken, da dies letztlich nur in eine Verschlechterung des Ergebnisses münden kann. Da aber während der Optimierung nur geringe Kontrolle über die Struktur vorhanden ist, kann nicht garantiert werden, dass ein Verfahren das theoretische Optimum überhaupt findet. Zudem ist bei einer vollständig freien Modellierung eine Analyse des Ergebnisses – über die Visualisierung der Modelle oder die Berechnung von Abständen – erschwert.

Man verzichtet außerdem auf Grundwissen, das man über Schrift besitzt, und das man zudem bei der Entscheidung zur Modellierung mit HMMs schon vorausgesetzt hat. Zeichen haben verschiedene Schreibvarianten, die durch eine Abfolge von Zuständen beschrieben werden können. Bereits die Entscheidung für eine Modellierung mit Hidden-Markov-Modellen ist das Ergebnis dieser Annahmen über die Natur der Schrift. Eine Beschränkung auf gewisse Klassen von HMM-Strukturen ist insofern nur eine Variante dieser allgemeinen Entscheidung. Deshalb soll weiterhin von folgenden Grundannahmen über die Schrift ausgegangen werden, welche schon die Grundlage für das in Kapitel 5 vorgestellte Schriftmodell darstellten:

- *Grapheme* sind die kleinste Einheit, in die Wörter sinnvollerweise zerlegt werden können. Sie werden zumeist mit Buchstaben, Ziffern und Sonderzeichen assoziiert.
- Zwischen Graphemen ist ein optionaler Leerraum.
- Grapheme können in unterschiedlichen Gestalten auftreten. Die verschiedenen Schreibweisen bezeichnet man als *Allographen*.
- Jede der Schreibweisen kann durch eine Menge aufeinanderfolgender Zustände charakterisiert werden.

Die genannten Annahmen laufen auf eine Struktur der HMMs heraus, wie sie in Abbildung 8.2 dargestellt ist: Jedes Graphem wird durch ein eigenes Buchstaben-HMM repräsentiert. Das Buchstabenmodell besteht aus einer Anzahl von parallelen Pfaden, die die verschiedenen Schreibweisen repräsentieren. Der Pfad jeder Schreibweise teilt sich weiter in eine Folge von Zuständen auf, die komplett durchlaufen werden müssen. Jeder Pfad für sich ist also ein lineares links-rechts-Modell. Die Schreibweisen sämtlicher Modelle enden mit einem optionalen Pausenzustand, den sich alle Grapheme teilen. Das Schriftmodell lässt sich somit durch die folgenden freien Parameter vollständig spezifizieren:

- Die Anzahl der zu modellierenden Grapheme.

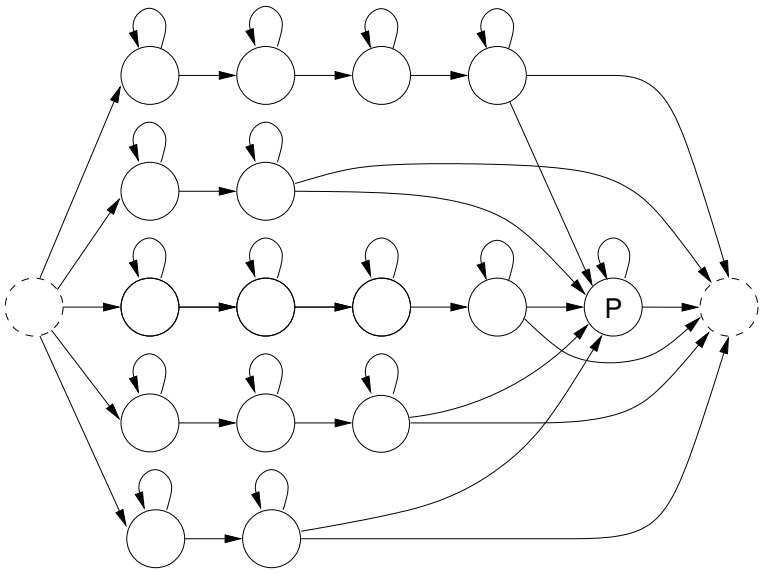

Abbildung 8.2: Variables Buchstabenmodell. Der erste freie Parameter ist die Anzahl der verschiedenen Gestaltklassen und entspricht der Anzahl der parallel liegenden Zustandspfade. Der zweite Freiheitsgrad liegt in der Länge der einzelnen Pfade, die allerdings in ihrer Struktur als lineare links-rechts-Modelle festgelegt sind.

- Für jedes dieser Grapheme die Anzahl der möglichen Schreibvarianten oder Allographen. Sie entspricht der Anzahl der Pfade in jedem Buchstaben-HMM.

- Für jeden der Allographen die Anzahl der Zustände, also die Länge des entsprechenden Pfades.

Es ist nun zu überlegen, auf welche Weise jeder dieser Parameter zu bestimmen ist.

Die Anzahl der zu modellierenden Buchstaben wird hauptsächlich durch die Art der Anwendung bestimmt. Dabei spielen Fragen der folgenden Art eine Rolle: Wie groß ist das Alphabet der zu erkennenden Schrift? Welche Sonderzeichen müssen unterschieden werden? Müssen Ziffern erkannt werden? Es handelt sich also um einen Parameter, der sich aus der geforderten Anwendung ergibt und deshalb direkt vorgegeben werden muss. Eine automatische Bestimmung wäre zwar möglich, indem zum Beispiel sämtliche Zeichen einer Stichprobe mit einer bestimmten Eigenschaft, etwa einer minimalen Auftretenshäufigkeit, berücksichtigt werden. Solche Möglichkeiten sollen hier aber nicht weiter untersucht werden.

Gegenstand der Topologiebestimmung sind allein die Parameter der einzelnen Graphemmodelle. Durch einen Vergleich mit Abbildung 5.3 sieht man, dass ihre Struktur genau durch die im Abschnitt 5.2 beschriebenen, fest vorgegebenen Buchstabenmodelle des Erkennungssystems realisiert ist. Die freien Strukturparameter sind dort mit plausiblen Werten belegt: Für jeden Buchstaben gibt es vier Schreibvarianten, die sämtlich durch drei Zustände modelliert werden. Diese Werte wurden in vielen Test anhand der zu erzielenden Erkennungsrate optimiert, und sind insofern schwierig zu verbessern. Es stellt sich die Frage, ob die in einer

solchen Modellierung enthaltene Vorabinformation über die Schrift nicht tatsächlich genutzt werden sollte, oder welche Argumente gegen eine solche Vorgabe der Struktur sprechen.

Die vier verschiedenen möglichen Schreibvarianten sind durch eine Unterscheidung von Groß- und Kleinschreibung, und jeweils wieder von Schreibschrift und Blockschrift motiviert. Diese Unterscheidung erscheint jedoch willkürlich und ist nicht objektiv messbar: Bei der Generierung von Trainingsdaten ist es häufig gar nicht möglich, eine Schreibweise eindeutig zuzuordnen, insbesondere die Unterscheidung zwischen Schreibschrift und Blockschrift ist auch für Menschen eine schwierige Aufgabe. Darüber hinaus ist die vorgegebene Kategorisierung inkonsistent, da sie nicht auf alle Mengen von Graphemen übertragbar ist. Für die Erkennung von Ziffern macht zum Beispiel weder die Unterscheidung in Groß- und Kleinschrift, noch in Block- und Schreibschrift Sinn. Dennoch können Schreibweisen unterschieden werden, ein prominentes Beispiel ist die deutsche und die amerikanische Schreibweise der Eins. Aus diesen Gründen benötigt jede neue Anwendung eine erneute Vorgabe der Schreibvarianten, was eine Automatisierung dieses Vorgangs notwendig macht.

Die der einheitlichen Pfadlänge drei zugrunde liegende Annahme ist, dass sich bei Buchstaben generell Anfang, Mitte und Ende unterscheiden lassen. Diese Annahme mag auf einige Schriftarten anwendbar sein, ist aber sicherlich keine geeignete Grundlage für eine generelle Modellierung von Schrift. Außerdem spiegelt die Annahme einer einheitlichen Pfadlänge die unterschiedliche Komplexität der verschiedenen Buchstaben nicht wider: Großbuchstaben sind meist breiter und komplexer als Kleinbuchstaben. Und auch die Eigenschaften der Merkmalsbestimmung werden nicht berücksichtigt. Verschiedene Merkmalssätze können unterschiedlich granular sein, etwa durch die Variation der Fensterbreite, was unterschiedlich detaillierte Modellierungen erlaubt. Ohne eine automatische Bestimmung der Pfadlänge ist man also weder von der Vorverarbeitung und Merkmalsextraktion, noch von den Eigenschaften der zu modellierenden Schrift unabhängig. Auch für diesen Parameter ist es also sinnvoll, Kriterien und Methoden zu entwickeln, die eine automatische Bestimmung möglich machen.

8.3 Übersicht

Im nächsten Kapitel werden zunächst einige allgemeine theoretische Überlegungen zur Wahl von Modellen angestellt. Es werden Kriterien hergeleitet, mit denen sich eine optimale Modellierung bestimmen lässt. In Kapitel 10 werden dann Verfahren vorgestellt, mit denen speziell HMM-Topologien optimiert werden. Daran anschließend werden Überlegungen angestellt, was bei der Modellierung des konkreten Handschriftmodells beachtet werden muss.

Es werden zwei von einander unabhängige Verfahren zur Optimierung entwickelt. Zunächst werden in Kapitel 11 die einzelnen Schreibvarianten, modelliert durch Zustandspfade, separat optimiert. Für das zweite Verfahren, die Bestimmung der optimalen Anzahl von Schreibvarianten selber, müssen unter anderem Bewertungsmaße für HMMs entwickelt werden, welche in Kapitel 12 vorgestellt werden. In diesem Kapitel erfolgt auch die Umsetzung

der Modellwahlkriterien auf das konkrete Schriftmodell. Mit den hergeleiteten Bewertungsmaßen erfolgt dann in Kapitel 13 die Bestimmung der Schreibvarianten. Neben einer visuellen Überprüfung der Plausibilität der Ergebnisse wird auch untersucht, wie gut die gemessenen Erkennungsergebnisse mit den theoretischen Modellwahlkriterien korrelieren. Das endgültige Ergebnis der Schriftmodellierung wird abschließend durch eine Kombination der beiden vorgestellten Verfahren bestimmt.

Kapitel 9

Kriterien zur Modellauswahl

Arbeiten zur Topologiebestimmung von HMMs gehen häufig davon aus, dass es eine „richtige“ Topologie gibt. Man nimmt an, dass die Beobachtungen durch einen konkreten Prozess erzeugt werden, der durch die Zustände eines HMM korrekt beschrieben werden kann. Entsprechend werden künstliche Daten, die mit einem „Original“-HMM generiert werden, zur Evaluation von Algorithmen zur Strukturbestimmung verwendet [Sto94,Li00a]. Es wird davon ausgegangen, dass ein Algorithmus dann richtig arbeitet, wenn mit genügend Daten das ursprüngliche HMM rekonstruiert werden kann.

Bei der hier betrachteten Anwendung der Handschrifterkennung liegt eine solche Situation jedoch nicht vor. Die Aufgabe des HMM ist es nicht, ein Modell des Schreibens darzustellen, das den Prozess der Schrifterzeugung als Abfolge erlaubter Zustände beschreibt. HMMs werden stattdessen deshalb eingesetzt, weil sie die räumliche Variabilität realer Schriftdaten erfassen können. Die Güte einer Modelltopologie wird dadurch charakterisiert, wie gut das Modell reale Daten beschreiben kann, und nicht durch den Vergleich mit einer Referenztopologie. Eine solche steht für die Handschrifterkennung nicht zur Verfügung.

In diesem Kapitel werden allgemeine Kriterien entwickelt, die die Eignung von Modellen zur Beschreibung realer Daten zeigen. Sie dienen in den weiteren Kapiteln als Grundlage für die Definition konkreter Kriterien zur Optimierung von HMMs.

9.1 Kriterien für Modelle

Für die Beschreibung von Vorgängen in beliebigen Bereichen werden *Modelle* verwendet, um zugrundeliegende Prozesse zu beschreiben. In der Physik werden zum Beispiel Modelle für die Beschreibung und Vorhersage von Messungen erstellt. Es stellt sich häufig die Frage, wie detailliert ein Modell sein muss, damit Beobachtungen *adäquat* beschrieben werden können.

Es muss ein Gleichgewicht zwischen *Generalisierung* und *Spezialisierung* hergestellt werden. Ist ein Modell zu einfach, so werden wesentliche Eigenschaften der Prozesse nicht erfasst, und das Modell ist dadurch nicht in der Lage, neue Beobachtungen korrekt vor-

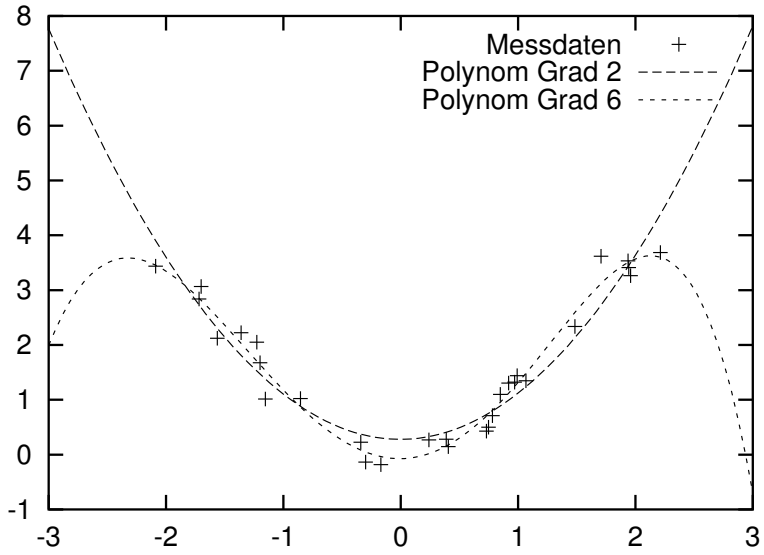

Abbildung 9.1: Näherung von Messkurven durch Polynome: Eine Anzahl von Messpunkten wird durch Polynome verschiedenen Grades approximiert. Auch wenn der mittlere Fehler durch ein detaillierteres Modell, hier ein Polynom vom Grad 6, reduziert werden kann, besitzt ein einfacheres Modell, das Polynom vom Grad 2, die bessere Vorhersagekraft für weitere Messungen, besonders an den Rändern des Messbereiches.

herzusagen. Wird hingegen die Komplexität zu hoch angesetzt, so besteht die Gefahr der Überanpassung an bekannte Beobachtungen, und das Modell verliert aus diesem Grund seine Voraussagekraft. Es ist angestrebt, ein *möglichst* einfaches Modell zur Beschreibung zu verwenden, das die Beobachtungen aber korrekt wiedergibt. Dieses Prinzip bezeichnet man als „*Occam's Razor*“ [Gib97, Ras01].

9.1.1 Occam's Razor

Das Problem lässt sich anhand von *geschachtelten Modellen* verdeutlichen. Das sind Modelle, die sich in der Anzahl der Freiheitsgrade unterscheiden, wobei das komplexere Modell das einfachere als Spezialfall enthält [For00]. Zum Beispiel können Messpunkte in der Ebene durch Polynome verschiedenen Grades beschrieben werden, wie in Abbildung 9.1 gezeigt wird. Für jede Anzahl n von Messpunkten ist es möglich, sie durch Erhöhung der Parameteranzahl beliebig genau zu approximieren, wodurch jedoch die Voraussagekraft des Modells nicht steigt. Es geht nicht nur um die möglichst genaue Beschreibung der Beobachtungen, sondern auch darum, die zugrundeliegenden Prozesse zu verstehen. Ein Beispiel aus der Physik verdeutlicht, wie weit man bei der Vereinfachung eines Modells gehen darf. Die Relativitätstheorie enthält die klassische Mechanik als Sonderfall. Hier unterscheidet der Bereich, in dem die Messungen stattfinden, über die Eignung der verschiedenen Modelle. Im Grenzfall hoher Geschwindigkeiten erweist sich das klassische Modell als zu einfach und damit ungeeignet.

Im Fall verschiedener HMM-Topologien handelt es sich nicht um geschachtelte Modelle, da das Hinzufügen von Zuständen auf die anderen Zustände zurückwirkt und deshalb nicht einfach einem weiteren Freiheitsgrad entspricht. Dennoch kann man auch hier durch beliebig große Modelle die Adaptionsdaten beliebig genau modellieren, wodurch man aber die Fähigkeit für die Erkennung neuer Daten verliert. Auch hier ist es deshalb wichtig, den richtigen Grad der Generalisierung zu finden.

9.1.2 Schätzfehler

Bei der Schriftmodellierung trifft man auf einen weiteren Aspekt bei der Modellwahl, der mit dem endlichen Vorrat an Trainingsdaten zusammenhängt. Man unterscheidet bei der Modellbestimmung neben einem Modellierungs- oder Näherungsfehler (*approximation error*), der angibt, wie gut eine wahre Verteilung angenähert werden kann, auch einen Schätzfehler (*estimation error*), der die Sicherheit bezeichnet, mit der die Modellparameter tatsächlich bestimmt werden können [For00]. Je mehr Parameter man zur Verfügung hat, umso besser *könnte* die wahre Verteilung angenähert werden, umso größer ist aber der Fehler beim Schätzen der Parameter. Im Grenzfall detailliertester Modellierung stehen pro Zustand nur einzelne Adaptionsvektoren zur Verfügung, mit denen die HMM-Parameter nicht mehr in geeigneter Weise trainiert werden können. Bei ausreichend vielen Trainingsdaten verschwindet der Schätzfehler, in einem realen System, bei dem stets nur endlich viele Daten vorhanden sind, muss er aber berücksichtigt werden. Die Suche nach dem geeigneten Kompromiss zwischen systematischem Fehler und Varianz wird prägnant als „*bias-variance trade-off*“ bezeichnet.

Eine weitere Schwierigkeit der Modellwahl bei begrenzten Trainingsdaten rührt von der Qualität der Daten her. Sind die Daten fehlerhaft, so werden bei detaillierter Modellierung gerade die falschen Daten im Modell repräsentiert. Dies ist ein weiteres Argument für die Beschränkung der Modellkomplexität nach oben.

9.2 Bayes'sche Modellierung

Modellparameter werden so bestimmt, dass das Modell vorgegebene Trainingsdaten bestmöglich beschreibt. Ist die Struktur eines Modells frei, dann lässt sich die Beschreibung der Daten durch eine Vergrößerung des Modells beliebig verbessern. Wenn die Struktur selbst zu den zu optimierenden Variablen gehört, muss deshalb ein Ausgleich zu Gunsten einfacher Modelle gegeben sein, damit eine geeignete Generalisierung erreicht wird. Mithilfe des Bayes-Formalismus können verschiedene Ausgleichsfaktoren hergeleitet werden [Sto94].

9.2.1 A-priori-Wahrscheinlichkeiten für Modelle

Ist eine Modellstruktur fest vorgegeben, wie zum Beispiel in Abschnitt 3.4 angenommen, so erfolgt das Training der Modellparameter nach dem *Maximum-Likelihood*-Kriterium. Ge-

sucht ist das Modell $\hat{M}$, dessen Wahrscheinlichkeit bei gegebenen Trainingdaten X maximal ist: $\hat{M} = \text{argmax}\, P(M|X)$. Diese Wahrscheinlichkeit lässt sich mithilfe der Bayes-Formel

$$P(M|X) = \frac{P(X|M) \cdot P(M)}{P(X)} \tag{9.1}$$

berechnen. Standardmäßig werden die einzelnen Terme dieses Ausdrucks folgendermaßen bezeichnet [Mac92]:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidenz}}. \tag{9.2}$$

Da die Trainingsstichprobe fest ist, ist auch die Evidenz $P(X)$ konstant und wird nicht weiter beachtet. Dasselbe wird im Fall einer vorgegebenen Topologie für die a-priori-Wahrscheinlichkeit des Modells $P(M)$ angenommen. Es ist also $P(M|X)$ proportional zur Likelihood $P(X|M)$, weshalb man bei der Adaption mit fester Struktur von *Maximum-Likelihood-*, oder kurz ML-Schätzung spricht.

Ist die Struktur jedoch variabel, so führt dieses Kriterium zur Auswahl von Modellen mit der höchsten Komplexität. Die ML-Schätzung muss erweitert werden, indem die a-priori-Modellwahrscheinlichkeit $P(M)$ als variabler Ausgleichsfaktor in die Suche nach dem Maximum miteinbezogen wird:

$$P(M|X) \propto P(X|M) \cdot P(M). \tag{9.3}$$

Da hierbei das Produkt aus a-priori-Wahrscheinlichkeit $P(M)$ und Likelihood $P(X|M)$ maximiert wird, spricht man von *Maximum-A-Posteriori-*, kurz MAP-Schätzung [Gau94]. Die Modellwahrscheinlichkeit $P(M)$ bezeichnet man einfach als *Prior*.

Einen möglichen Einsatz erfährt die MAP-Schätzung beim Nachtraining von Modellen, wie sie in der Sprecheradaption bei der Spracherkennung [Neu98] oder der Schreiberadaption in Systemen zur Online-Handschrifterkennung [Bra01] notwendig ist. In diesen Fällen ist es Aufgabe der Priors, die Parameter des sprecher- oder schreiberunabhängigen Basissystems widerzuspiegeln. Eine andere Anwendung besteht in der Regularisierung des Modells [Gha01] mit dem Ziel, das Modell so zu trainieren, dass es gut generalisiert. Der Prior dient dann als Gegengewicht zur Likelihood zu Gunsten einfacher Modelle und wird dafür häufig ad-hoc eingeführt. Die Parameter werden durch den Prior beim Training geglättet, wodurch eine Überanpassung an die Trainingsdaten vermieden werden kann [Sto93].

Das Ergebnis der Adaption und somit der Grad der Generalisierung wird von der konkreten Wahl des Priors bestimmt. Die Bedeutung des Priors liegt irgendwo in der Mitte zwischen der allgemeinen Festlegung einer Menge möglicher Strukturen und der konkreten Vorgabe einer Modellstruktur. Mit der Entscheidung für ein bestimmtes Kriterium legt man also zum Teil die zukünftige Modellstruktur fest. Unterschiedliche Kriterien [For00] werden durch verschiedene Herleitungen und Näherungen begründet. Sie werden in den folgenden Abschnitten beschrieben.

9.2.2 A-priori-Wahrscheinlichkeiten für Parameter

Ein Modell M lässt sich in einen Anteil M_s, der die Struktur darstellt, und die freien Parameter θ_M, durch deren Wahl diese Struktur realisiert wird, aufteilen. Beide Anteile gehen in die Modellwahrscheinlichkeit $P(M)$ ein, die als Verbundwahrscheinlichkeit

$$P(M) \equiv P(M_s, \theta_M) = P(\theta_M | M_s) \cdot P(M_s) \tag{9.4}$$

dargestellt werden kann. Diese Trennung in Struktur und Parameter ist nicht eindeutig. Zum Beispiel können erlaubte Zustandsübergänge ein Strukturmerkmal sein, sie können aber auch über die Übergangswahrscheinlichkeiten, die nicht Null sind, zu den Parametern gehören.

Mit der formalen Aufteilung in Parameter und Struktur nimmt die Bayes-Formel für die Schätzung der Parameter θ die Form

$$P(\theta | X, M_s) = \frac{P(X|\theta, M_s) \cdot P(\theta|M_s)}{P(X|M_s)} \tag{9.5}$$

an. Im nächsten Abschnitt wird die Bedeutung der Evidenz $P(X|M_s)$ im Nenner der rechten Seite für die Wahl der Modellstruktur klar werden. Bei der Frage, welches ein geeigneter Prior $P(\theta|M_s)$ für HMM-Parameter ist, wird als natürliche und zweckmäßige Lösung häufig ad-hoc die Dirichlet-Verteilung gewählt [Gha01]. Eine Übersicht über ihre Eigenschaften gibt Anhang B.

9.2.3 Bayes'sche Integration

Mit der Wahl eines Priors und der Bestimmung des Modells über die MAP-Schätzung lassen sich Modelle generieren, die eine geeignete Generalisierung der Trainingsdaten garantieren. Man kann bei dieser Methode verschiedene Modellstrukturen aber nicht direkt, unabhängig von der Wahl der Parameter miteinander vergleichen. Das Hauptproblem liegt in den verborgenen Variablen der Modelle. Die Berechnung der Likelihood in (B.1) setzt zum Beispiel neben der richtigen Wahl der Parameter θ auch die Kenntnis der Ereignisse $c_1, \ldots, c_n$ voraus. Diese Ereignisse beziehen sich auf Zustände des HMM, die zum einen nicht beobachtbar und zudem selber von der Modellierung abhängig sind.

Ist eine reine Bewertung der Modellstruktur gesucht, die unabhängig von der Wahl der Parameter ist, wird wiederum ein Bayes'scher Ansatz gewählt [Che88]:

$$P(M_s|X) = \frac{P(X|M_s) \cdot P(M_s)}{P(X)}. \tag{9.6}$$

Wie üblich geht man davon aus, dass die Modellwahl von der Evidenz der Daten unabhängig ist. Es wird aber auch kein expliziter Prior $P(M_s)$ für die Struktur vorgegeben, da die Bewertung der Struktur implizit über die a-priori-Wahrscheinlichkeiten der Parameter geschehen kann, wie im weiteren gezeigt werden wird. Auch die Wahl der Modellstruktur erfolgt über eine Maximum-Likelihood-Schätzung, $P(M_s|X) \propto P(X|M_s)$.

Der Term $P(X|M_s)$ entspricht dem Nenner in (9.5), die Modelle werden also nach der Evidenz der Daten bei gegebenen Parametern evaluiert [Mac92, Hon01]. Da der Nenner in (9.5) zur Normierung der Wahrscheinlichkeiten dient, erfolgt die Berechnung über die Bildung des Integrals

$$P(X|M_s) = \int_{\theta} P(X|\theta, M_s) P(\theta|M_s) d\theta \tag{9.7}$$

über die Menge aller möglichen Parameter θ. Man bezeichnet diese Berechnung auch als *Bayes'sche Integration* [Gha01].

9.3 Näherungen

Für die Berechnung von (9.7) muss sowohl über die Parameter, als auch über die verborgenen Zustände integriert werden. In den meisten Fällen – auch bei HMMs – kann dafür keine geschlossene Lösung angegeben werden. Die hohe Dimension der Parameter und der verborgenen Zustände macht Näherungen notwendig. Verschiedene Ansätze werden verfolgt [Gha01]: Monte-Carlo-Methoden, Ensemble-Lernen, die Laplace- und die Cheeseman-Stutz-Näherung. Eine Anwendung der Monte-Carlo-Methode durch Gibbs-Sampling wird in [Li00a] beschrieben. Ensemble-Lernen, auch als Variationsmethode bezeichnet, behandelt [Mac97] ausführlich. Im folgenden werden Ansätze vorgestellt, die auf der Laplace- und Cheeseman-Stutz-Näherung beruhen.

9.3.1 Bayes'sches Informations-Kriterium

Das Bayes'sche Informations-Kriterium wird über die Laplace-Näherung hergeleitet.

Als Laplace-Näherung wird im allgemeinen die Approximation eines Integrals $\int f(v)\, dv$ bezeichnet, die darin besteht, dass eine Gaußverteilung am Maximum $\tilde{v}$ der Funktion $f(v)$ eingepasst wird und das Volumen unter der Gaußkurve berechnet wird [Mac98]. Die Kovarianz der Gaußverteilung ist durch die negative Hessematrix A von $f(v)$ bei $\tilde{v}$ gegeben. Die Näherung entspricht einer Taylor-Entwicklung zweiten Grades am Maximum $\tilde{v}$ des Integranden.

Im konkreten Fall soll nach (9.7) die Evidenz $P(X|M_s)$ berechnet werden. Dazu wird für die logarithmierten a-posteriori-Wahrscheinlichkeit eine Taylorentwicklung durchgeführt [Li00a]. Nach dem Einsetzen der Entwicklung in die Exponentialfunktion erhält man

$$\begin{aligned} P(\theta|X, M_s) &\propto P(X|\theta, M_s) P(\theta|M_s) \\ &\approx P(X|\tilde{\theta}, M_s) P(\tilde{\theta}|M_s)\, e^{-\frac{1}{2}(\theta-\tilde{\theta}) A (\theta-\tilde{\theta})^T}. \end{aligned} \tag{9.8}$$

Damit lässt sich die Evidenz in geschlossener Form integrieren,

$$\log P(X|M_s) \approx \log P(X|M_s, \tilde{\theta}) + \log P(\tilde{\theta}|M_s) + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|A|, \tag{9.9}$$

wobei d die Dimension von θ, also die Anzahl der freien Parameter bezeichnet. Die Berechnung dieses Ausdrucks ist wegen der Bestimmung der Hessematrix immer noch zu aufwändig. Es werden deshalb weitere Approximationen durchgeführt.

Für die Berechnung von (9.9) werden nur die Terme betrachtet, die mit der Anzahl T der Daten wachsen. Dies sind zum einen $\log P(X|M_s,\tilde{\theta})$, das linear mit T wächst, und $\log|A|$, das wie $d\log T$ wächst [Sch78, Hec95]. Bei großem T kann außerdem die MAP-Konfiguration $\tilde{\theta}$ durch die ML-Konfiguration $\hat{\theta}$ ersetzt werden; diese maximiert die Wahrscheinlichkeit $P(X|M_s,\theta)$ und ist Ergebnis des Baum-Welsh-Trainings. Man erhält dadurch das Bayes'sche Informations-Kriterium, BIC (*Bayesian Information Criterion*),

$$\log P(M_s|X) \propto \log P(X|M_s) \approx \log P(X|M_s,\hat{\theta}) - \frac{d}{2}\log T. \tag{9.10}$$

Das Ergebnis dieser Näherung ist recht intuitiv. Das BIC ist die Summe zweiter Terme. Der erste Term misst, wie gut das parametrisierte Modell die Daten repräsentiert, der zweite bestraft die Komplexität des Modells. Mit ihm hat man eine a-priori-Wahrscheinlichkeit $P(M_s)$ für die MAP-Schätzung nach Gleichung (9.3) hergeleitet, ohne sie willkürlich explizit vorzugeben.

Diese a-priori-Wahrscheinlichkeit ist das Produkt zweier Größen. Die eine ist die Anzahl d der Parameter, die sehr anschaulich ein Maß für die Komplexität eines Modells darstellt. Der andere Term, $\log T$, spiegelt den Einfluss der Trainingsdaten wider. Er lässt sich folgendermaßen interpretieren: Da die log-Likelihood linear mit der Anzahl T der Daten wächst, müsste dies auch für den Ausgleichsterm gelten, der aber nur logarithmisch wächst. Bei großem T schwindet also der Einfluss des Ausgleichsterms. Dadurch wird bei wenigen Daten eine hohe Modellkomplexität stärker bestraft als wenn viele Daten zur Schätzung des Modells zur Verfügung stehen. In diesem Fall tritt der zu Beginn des Kapitels erwähnte Schätzfehler (*estimation error*) auf. Dieser bezieht sich nicht auf die prinzipielle Generalisierbarkeit der Daten, bewirkt aber über die ungenaue Schätzung der Parameter in der Praxis eine schlechte Generalisierung.

9.3.2 Akaike Informations-Kriterium

Eng mit dem BIC verwandt ist das Akaike Informations-Kriterium, AIC (*Akaike information criterion*), welches durch

$$P(M_s|X) \approx \log P(X|M_s,\hat{\theta}) - d \tag{9.11}$$

gegeben ist. Bei einer einheitlichen Datenmenge von $T = e^2 \approx 7,39$ ist das AIC mit dem BIC (9.10) identisch. Stehen mehr Daten zum Schätzen der Parameter zur Verfügung, dann ist der Bestrafungsterm des AIC im Vergleich zum BIC kleiner, wodurch eine stärkere Tendenz zugunsten komplexer Modelle entsteht. Der Unterschied zwischen den beiden Kriterien liegt also in dem Wert, der dem Schätzfehler beigemessen wird.

Die Idee für die Herleitung des AIC besteht darin, dass bei der Modellwahl die Differenz zwischen der unbekannten *wahren* Verteilung $P(X)$ und dem konkreten Modell $P_M(X)$ minimiert wird [For00]. Dieser Abstand wird mit dem Kullback-Leibler-Abstand

$$\begin{aligned} D &= \int P(X) \log \frac{P(X)}{P_M(X)} dX && (9.12) \\ &= \text{const} - \int P(X) \log P_M(X)\, dX && (9.13) \\ &= \text{const} - E_X\left[\log P_M(X)\right] && (9.14) \end{aligned}$$

quantifiziert. Die Minimierung der KL-Differenz entspricht dem Maximieren des Erwartungswerts $E_X\left[\log P_M(X)\right]$, der bei vielen Trainingsdaten, also großem T, gegen

$$E_X\left[\log P_M(X)\right] \longrightarrow \frac{1}{T}\sum_{t=1}^{T} \log P_M(X_t) \tag{9.15}$$

geht. Durch die Bildung des Maximums werden die Modellparameter $\hat{\theta}$ bestimmt. Es ist aber nicht sinnvoll, mit diesem Kriterium auch unterschiedliche Modelle miteinander zu vergleichen. Bei der Bestimmung von $E_X[\log P_M(X|\hat{\theta})]$ würden dieselben Daten X zweimal verwendet werden, sowohl für die Berechnung der Likelihood, als auch für die Schätzung der Parameter $\hat{\theta}$. Dabei entsteht eine Tendenz zu Ungunsten der korrekten Verteilung. Akaikes Beitrag bestand darin, den Erwartungswert dieses Ungleichgewichts zu bilden [Boz87]:

$$E_\theta\left[E_X\left[\log P_M(X|\hat{\theta})\right] - \frac{1}{T}\sum_{t=1}^{T} \log P_M(X|\hat{\theta})\right] \approx -\frac{d}{T}. \tag{9.16}$$

Aus diesem Ergebnis lässt sich direkt der Korrekturterm des Akaike Informations-Kriteriums (9.11) herauslesen.

9.3.3 Cheeseman-Stutz-Näherung

Das Hauptproblem bei der Lösung des Integrals (9.7) zur Berechnung der Evidenz besteht darin, dass es *verborgene Variablen* gibt: bei HMMs die nicht beobachteten Modellzustände. Wären alle Variablen bekannt, so vereinfachte sich die Auswertung des Integrals beträchtlich [Che96, Chi97]. Man betrachtet deshalb die Gleichung

$$P(X|M_s) = P(X'|M_s) \cdot \frac{\int P(X,\theta|M_s)d\theta}{\int P(X',\theta|M_s)d\theta}, \tag{9.17}$$

in der mit X' die *vollständigen* Daten bezeichnet sind. Die Cheeseman-Stutz-Näherung verwendet diese Gleichung und vereinfacht sie. Damit gelingt es, das Problem der Evidenzberechnung auf den Fall vollständiger Daten $P(X'|M_s)$ zu reduzieren.

Zähler und Nenner im rechten Term der Gleichung werden durch die Laplace-Approximation genähert. Weitere Vereinfachungen sind möglich, wenn beide Wahrscheinlichkeitsverteilungen ihr Maximum bei derselben Parameterkonfiguration $\hat{\theta}$ haben; man kann von

dieser Annahme ausgehen, da sie schon der EM-Schätzung der Parameter zugrunde liegt. Wird zusätzlich noch die BIC-Näherung durchgeführt, so erhält man

$$\begin{aligned}\log P(X|M_s) \approx \log P(X'|M_s) - \log P(X'|\hat{\theta}, M_s) + \frac{d'}{2}\log T \\ + \log P(X|\hat{\theta}, M_s) - \frac{d}{2}\log T.\end{aligned} \tag{9.18}$$

Die Unterschiede in der Dimension der Modelle d und d' werden ebenso vernachlässigt wie die Unterschiede in der Likelihood. Im Ergebnis ersetzt die Cheeseman-Stutz-Näherung die Evidenz unvollständiger durch die Evidenz vollständiger Daten:

$$\log P(X|M_s) \approx \log P(X'|M_s). \tag{9.19}$$

Das Prinzip entspricht dem des EM-Trainings, bei dem zusammen mit den Parametern auch die verborgeneren Variablen geschätzt werden.

Die Näherung vereinfacht die Berechnung der Evidenz, da nicht mehr über die verborgenen Variablen integriert werden muss. Man sieht ihren Wert als gegeben an und rechnet mit den Erwartungswerten, die bei der Bestimmung der ML-Parameter im *Expectation*-Schritt des EM-Training implizit berechnet werden. Im folgenden werden zwei Methoden vorgestellt werden, die $P(X'|M_s)$ nähern oder explizit berechnen.

Maximumsnäherung

In [Li00a] wird von der starken Annahme ausgegangen, dass die Integration über alle Parameterkonfigurationen durch die ML-Konfiguration genähert werden kann:

$$P(X'|M_s) = \int P(X', \theta|M_s)\, d\theta \approx P(X', \hat{\theta}|M_s). \tag{9.20}$$

Die Wahrscheinlichkeitsverteilung auf der rechten Seite lässt sich in zwei Terme aufspalten, und man erhält das Kriterium

$$\log P(X|M_s) \approx \log P(X'|M_s, \hat{\theta}) + \log P(\hat{\theta}|M_s). \tag{9.21}$$

Die zwei Summanden drücken erneut die entgegengesetzten Tendenzen für und wider die Generalisierbarkeit des Modells aus. Der erste Term ist die Likelihood der Trainingsdaten, die während des Baum-Welsh-Trainings ermittelt wird. Er wächst mit zunehmender Modellkomplexität.

Der zweite Term gibt die Wahrscheinlichkeit der berechneten ML-Konfiguration $\hat{\theta}$ an; für ihre Bestimmung wird von einer Dirichlet-Verteilung (B.2) der Modellparameter ausgegangen. Bei der Vorgabe eines gleichförmigen Verteilungsparameters α für alle Verteilungen kann das Modellkriterium mit

$$\log P(X|M_s) \propto \log P(X|M_s, \hat{\theta}) + \sum_i \log \hat{\theta}_i^{\alpha} \tag{9.22}$$

berechnet werden. Komplexität eines Modells wird bestraft, indem jeder freie Parameter der Modellstruktur den Einfluss der Likelihood senkt.

Geschlossene Lösung des Integrals

Die Maximumslösung (9.22) besitzt die unerwünschte Eigenschaft, dass Modelle mit gleichverteilten Parameterwerten am höchsten bewertet werden. Diese Eigenschaft kann mit einem alternativen Ansatz [Sto94] vermieden werden, bei dem das Integral (9.7) geschlossen gelöst wird. Er nutzt die in Anhang B beschriebenen vorteilhaften Eigenschaften der Dirichlet-Verteilung.

Man benötigt dazu zunächst einen Ausdruck für die Likelihood der Trainingsdaten X'. Zu X' gehören neben den unmittelbaren Beoachtungen $X_t \in V$ auch die verborgenen Variablen; bei HMMs sind dies die nicht direkt beobachtbaren Zustände $q_t \in Q$. Es ist also ${X'}_t \in Q \times V$. Bei diskreten und semikontinuierlichen HMMs besteht der Beobachtungsraum V aus einem diskreten Codebuch C, das $k_{\max}$ verschiedene Symbole enthält. Multipliziert mit der Anzahl N der möglichen Zustände in Q gibt es damit $n = N \cdot k_{\max}$ verschiedene Ereignisse, die vom Viterbi- oder Forward-Backward-Algorithmus gezählt werden. Ihre Auftretenshäufigkeiten werden mit $(c_1, \ldots, c_n)$ bezeichnet. Jedem der n Ereignisse ist ein Modellparameter θ_n zugeordnet, der gerade die Wahrscheinlichkeit des Ereignisses angibt. Ist die Summe dieser Wahrscheinlichkeiten 1, so kann eine Multinomialverteilung postuliert werden, mit der sich die Likelihood der Trainingsdaten X' durch

$$P(X'|\theta, M_s) = \prod_{i=1}^{n} \theta_i^{c_i} \tag{9.23}$$

berechnen lässt. Für die Wahrscheinlichkeit der Parameter postuliert man entsprechend eine Dirichlet-Verteilung mit den Parametern $(\alpha_1, \ldots, \alpha_n)$

$$P(\theta|M_s) = \frac{1}{B(\alpha_1, \ldots, \alpha_n)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1}. \tag{9.24}$$

Damit lässt sich das Produkt der Wahrscheinlichkeitsverteilungen im Integral explizit darstellen. Das Integral lässt sich wie in (B.5) geschlossen lösen:

$$\begin{aligned} \int P(X', \theta|M_s)\, d\theta &= \int P(X'|\theta, M_s) \cdot P(\theta|M_s)\, d\theta \\ &\propto \frac{1}{B(\alpha_1, \ldots, \alpha_n)} \cdot \int \prod_{i=1}^{n} \theta_i^{c_i + \alpha_i - 1} d\theta \\ &= \frac{B(c_1 + \alpha_1, \ldots, c_n + \alpha_n)}{B(\alpha_1, \ldots, \alpha_n)}. \end{aligned} \tag{9.25}$$

Durch diesen Ausdruck ist die Evidenz der Daten formal von den konkreten Modellparametern $\hat{\theta}$ unabhängig. Über die Auftretenshäufigkeiten $(c_1, \ldots, c_n)$, die beim EM-Training im *Expectation*-Schritt bestimmt werden, sind die Modellparameter jedoch implizit vertreten. Sie werden im *Maximization*-Schritt aus den Auftretenshäufigkeiten berechnet und enthalten deshalb dieselbe Information.

Die interessanten Eigenschaften des Kriteriums werden in Anhang C diskutiert.

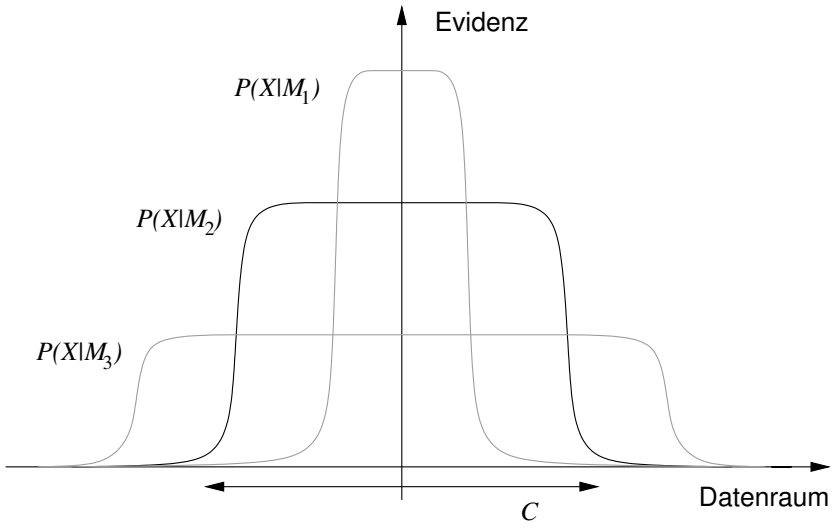

Abbildung 9.2: Das Prinzip des Occam-Faktors. Weniger komplexe Modelle werden bevorzugt, ohne dass komplexe Modelle explizit bestraft werden. Die x-Achse repräsentiert den Raum möglicher Daten, auf der y-Achse ist die Wahrscheinlichkeitsdichte für die einzelnen Beobachtungen aufgetragen. Das einfache Modell M_1 deckt einen kleineren Bereich des Datenraumes ab als die komplexeren Modelle M_2 und M_3. Im Gegenzug dazu ist aufgrund der Normierung die Wahrscheinlichkeitsdichte des Modells M_3 geringer als die der beiden anderen Modelle. Liegen die vom „wahren“ Prozess erzeugten Beobachtungen gerade in C, so ist die Evidenz der Daten für M_2 maximal [Mac92].

9.4 Occam-Faktoren

Gleichung (9.7) enthält keinen expliziten Faktor zur Bestrafung von Komplexität. Es wird kein Prior $P(M_s)$ verwendet, der die Wahrscheinlichkeit komplexer Modelle *a-priori* als niedrig einstuft. Auch Parameterverteilungen $P(\theta|M_s)$ wie die in Anhang B vorgestellte Dirichlet-Verteilung gehen von einer Gleichförmigkeit der Parameter aus.

Dennoch führen alle vorgestellten Näherungen zu einem Gleichgewicht zwischen Spezialisierung und Generalisierung. Sie lassen sich jeweils in zwei Terme zerlegen, von denen einer über die Wahrscheinlichkeit der Beobachtungen die Spezialisierung, also große Modelle begünstigt, während der andere als „Strafterm“ die Modellkomplexität niedrig hält. Das diesem Effekt zugrundeliegende Prinzip kann durch die Betrachtung der „Occam-Faktoren“ der Parameterverteilungen erhellt werden [Mac92].

Als Occam-Faktor bezeichnet man das Verhältnis aus dem prinzipiell erlaubten Volumenbereich der Modellparameter a-priori und dem Volumen der Parameterverteilung a-posteriori. Es ist der Faktor, um den sich – aufgrund der Beobachtungen – der Raum der Modellhypothesen verkleinert. Der Logarithmus des Occam Faktors kann als die Menge an Information angesehen werden, die bei der Beobachtung der Daten über die Parameter gewonnen wird.

Bei der Berechnung der Evidenz (9.7) tritt häufig ein starkes Maximum des Integranden

bei der wahrscheinlichsten Parameterverteilung auf. Man kann die Evidenz durch das Produkt aus der Höhe und der Breite dieses Peaks nähern; dadurch wird sie zu Likelihood und Occam-Faktor proportional. Der Occam-Faktor ist genau dann maximal, wenn die Beobachtungen und die Modellverteilung im Datenraum den gleichen Bereich einnehmen (Abbildung 9.2). Dadurch schafft er den Ausgleich zwischen Anpassbarkeit der Daten und Einfachheit der Modellierung. Explizit wird der Faktor bei der Laplace-Näherung, bei der er durch die Determinante der Gauß'schen Kovarianzmatrix (9.9) gegeben ist.

Kapitel 10

Verfahren zur HMM-Topologiebestimmung

Für das Training der HMM-Parameter bei fest vorgegebener Modellstruktur steht mit dem Baum-Welsh-Algorithmus ein effektives Verfahren zur Verfügung. Soll außerdem auch die Topologie der HMMs bestimmt werden, gibt es hingegen kein Standardverfahren. Im folgenden Abschnitt werden deshalb zunächst einige bekannte Optimierungsverfahren vorgestellt.

Im weiteren geht dieses Kapitel auf die besonderen Umstände ein, die in der Praxis bei der Berechnung der HMM-Strukturen für Buchstabenmodelle auftreten. Sie betreffen neben dem Verhältnis von Clusterung und Modellierung die besondere Rolle der Segmentierung der Trainingsdaten in die einzelnen Grapheme. Sie ist verbunden mit der Schwierigkeit, dass eine Menge vieler Buchstabenmodelle gleichzeitig zu bestimmen ist. Zuletzt muss das allgemeine Vorgehen an die Begebenheiten des konkreten, semikontinuierlichen Erkennungssystems angepasst werden, dessen Codebuch einen wichtigen Faktor für die Erkennungsleistung darstellt.

10.1 Bekannte Verfahren

Bei der Aufgabe, die Topologie in einem System mit Hidden-Markov-Modellen zu bestimmen, werden typischerweise zwei verschiedene Anwendungsziele verfolgt.

- Ein mögliches Ziel ist die unüberwachte *Clusterung* von Daten, deren zeitlicher Charakter mit HMMs modelliert wird. Bei der Handschrifterkennung wird dieser Aspekt durch die Bestimmung verschiedener Schreibvarianten für Buchstaben vertreten.

- Ein anderes Ziel ist die optimale *Anpassung* einer Modellstruktur an Daten, deren Zuordnung bekannt ist, zum Zweck der bestmöglichen Repräsentation und Generalisierung. Bei Schriftmodellen betrifft dies die konkrete Modellierung der einzelnen Allographen.

Die Realisierung beider Ziele kann sich überschneiden, da sich beide Aspekte von Daten – Bedeutung und Struktur – in der Topologie wiederfinden. Dient die HMM-Strukturbestimmung zur Informationsextraktion (zum Beispiel zur Bestimmung wichtiger Felder aus den Headern von wissenschaftlichen Zeitschriftenartikeln, vgl. [Sey99]), dann sind beide Punkte untrennbar miteinander verbunden, da die verschiedenen Bedeutungsklassen über die Struktur selber definiert werden.

Häufig liegt der Schwerpunkt aber entweder auf der Suche nach einer unüberwachten Clustermethode, etwa von medizinischen Daten, die zusätzlich noch eine Interpretation der Cluster zulässt, wofür HMMs eben gut geeignet sind [Li00a]. Oder die Modellierung steht, wie im Fall der Schrifterkennung, im Vordergrund, und die Clusterung in interpretierbare Allographen stellt letztlich nur einen nützlichen Nebeneffekt für die Analyse der Systemleistung dar. Für eine Betrachtung der verschiedenen Herangehensweisen an die Topologiebestimmung ist es deshalb hilfreich, sich die Zielsetzung ins Gedächtnis zu rufen, für die ein System entwickelt wird.

10.1.1 Clusterung mit HMMs

Besteht die Hauptaufgabe in einer unüberwachten Clusterung von Daten mit zeitlicher Struktur, so stellen HMMs zunächst mehr oder weniger nur ein Hilfsmittel für die Beschreibung der Cluster dar. Entsprechend arbeiten fast alle vorgestellten Methoden mit festen Modellstrukturen. Meistens werden lineare Modelle verwendet. Allerdings stellt die Clusterung dann nur einen ersten Teilschritt dar: Sie dient letztlich dazu, die Daten zunächst mit genaueren Kennzeichnungen zu versehen, mit denen in einem zweiten Schritt dann eine verbesserte Modellierung möglich wird.

Ein erstes System zur Clusterung mit HMMs wurde für gesprochene Zahlwörter vorgestellt [Rab89a]. Sein Ziel bestand darin, die Menge der Trainingswörter weiter aufzuteilen, um ein HMM-Spracherkennungssystem mit feiner unterteilten Modellen trainieren zu können und so die Fehlerrate zu senken. Das Prinzip der Clusterung entspricht dem k-means Algorithmus, eine gängige Bezeichnung ist deshalb auch „segmental k-means“ [Jua90]. Iterativ werden die Daten über das ML-Kriterium neuen Clustern zugewiesen und das System mit dieser Zuordnung neu trainiert. Die Anzahl der Modelle wird so lange erhöht, bis eine vorbestimmte Anzahl erreicht ist. Für die Initialisierung der neuen Modelle stehen zwei Möglichkeiten zur Verfügung: Bei der einen werden die bestehenden Modelle aufgespalten und ihre Parameter zufällig verändert, ein Vorgehen, das dem LBG-Algorithmus sehr ähnlich ist [Lin80]. Die andere Möglichkeit besteht darin, die bei der Erkennung nur schlecht bewerteten Modelle als neue Saatpunkte zu verwenden. Mit der zweiten, als *threshold clustering* bezeichneten Methode wird eine Generalisierung des Systems erreicht, die sich in einer Verbesserung der Fehlerrate bei einem unabhängigen Testdeck äußert.

Ein ähnliche Methode, bei der iterativ Modelle trainiert und den Daten neue Label zugewiesen werden, wird auch für Online-Handschrifterkennung eingesetzt [Per00]. Als zu-

sätzliche Aufgabe ist hier die Anzahl der Allographen nicht vorgegeben, sondern wird durch den Algorithmus selber bestimmt. Die vorgestellte Methode kann als „Ausprobieren“ betitelt werden: Bei steigender Clusteranzahl wird einfach nach einer Schulter in der gemittelten Likelihood der Trainingsdaten gesucht. Bei den getesteten Handschriftdaten wird von einer schnellen Konvergenz berichtet, bei der keine leeren Cluster entstehen.

Ein besonderes Augenmerk auf die Anzahl der Cluster wird in Arbeiten gelegt, die ein allgemeines Framework für die Clusterung von Daten von beliebiger, variabler Dimension aufstellen [Smy97, Cad00]. Auch hier werden HMMs zur Modellierung verwendet und ein verallgemeinerter EM-Algorithmus vorgeschlagen, der ähnlich wie die bisher beschriebenen Methoden arbeitet. Die Clusterung erfolgt in Iterationen abwechselnd mit dem Training der HMM-Parameter, ist dabei aber prinzipiell von der Parameterbestimmung unabhängig. Dadurch werden unterschiedliche Kriterien und Clusterstrategien möglich, mit denen die Anzahl der Cluster gesteuert werden kann. Vorgeschlagen werden eine Abstandsbewertung der Modelle, die auf dem Bayes'schen Ansatz der Modellwahl beruht, und eine einfache Realisierung von *Cross Validation*, bei der unabhängige Mengen von Daten für Parametertraining und Clusterung Verwendung finden. Auch hier, wie bei den anderen Verfahren, ist aber die Anzahl der Zustände in den einzelnen Modellbeschreibungen fest vorgegeben.

Bei der Online-Handschrifterkennung wird aber auch der direkte Weg eingeschlagen, zunächst die Trainingsdaten selber zu clustern, und mit ihnen danach die HMMs zu modellieren [Lee00]. Der Abstand der Daten wird mit *Dynamic Time Warping* definiert, die Modellierung der Cluster erfolgt durch Referenzmuster. Interessant ist dieser Weg aus zweierlei Gründen: Zum einen erfolgt die Clusterung so, dass versucht wird, die Daten nacheinander den verschiedenen Clustern zuzuordnen und nur dann, wenn es nötig ist, neue Clusterzentren zu definieren. Es ist dadurch einfach, neue Trainingsdaten einzubringen und damit eventuell neue Allographen in das System einzuführen. Der Ansatz führt die Bestimmung der Clusteranzahl sehr pragmatisch durch und orientiert sich dabei am eingesetzten Clusteralgorithmus selber. Zum anderen ist interessant, dass die Topologie der entstehenden Modelle anhand der repräsentativen Muster bestimmt wird. Im vorliegenden Fall betrifft dies zumindest die Anzahl der HMM-Zustände, die sich an der Länge der Referenzmuster orientiert. Dies ist der erste Schritt weg von der reinen Datenclusterung, hin zu einer direkten Strukturbestimmung der HMMs.

10.1.2 Direkte Bestimmung der HMM-Struktur

Bei den bisher vorgestellten Verfahren stand die Clusterung der Daten im Vordergrund. Entsprechend lag der Schwerpunkt der Betrachtungen auf dem Clusteralgorithmus selber, und die HMMs und das Training ihrer Parameter stellten nur die notwendigen Hilfsmittel zur Verfügung. Jetzt sei das eigentliche Ziel die Bestimmung der freien Parameter der HMMs, zu denen insbesondere die Struktur der Modelle gehört. Es werden zunächst einige einfache und direkte Verfahren vorgestellt.

Implizite Berechnung

Die Topologie eines HMM ist durch die Zustandsübergänge definiert, deren Wahrscheinlichkeit ungleich Null ist. Bei dieser Sichtweise ist es nicht notwendig, die Topologie als eigenständigen Satz von Parametern zu betrachten. Dementsprechend kann sie *implizit* mit dem normalen Baum-Welsh-Training bestimmt werden.

Notwendig ist nur, zunächst ein allgemeines Modell anzugeben, das eine maximale Anzahl von Zuständen und erlaubten Übergängen enthält. Man muss eine Obergrenze für die Anzahl der Zustände festsetzen, und kann eventuell die Struktur noch in anwendungsspezifischer Weise einschränken, etwa auf links-rechts-Modelle. Abgesehen davon ist die spätere Topologie in keiner Weise eingeschränkt. Das Modell wird in gewohnter Weise mit dem Baum-Welsh-Algorithmus trainiert. Die Struktur ergibt sich aus den trainierten Parametern, indem die Übergänge mit geringer Wahrscheinlichkeit gestrichen werden, und mit ihnen die Zustände, zu denen diese Übergänge führen. Das Verfahren ist einfach und schnell, da es nur die eine Iteration des Baum-Welsh-Trainings benötigt. Außer der Tatsache, dass ein größeres Modell trainiert werden muss, entsteht kein wesentlicher Zuwachs an Rechenzeit.

Man muss aber darauf spekulieren, dass sich die Struktur der Daten, also etwa die verschiedenen Schreibvarianten, als Pfade mit hohen Übergangswahrscheinlichkeiten von selber herausbilden. Das kann jedoch nicht unbedingt erwartet werden, insbesondere wenn die Parameter zu Anfang des Trainings nicht geeignet gesetzt sind. Dann können sich die Daten aufgrund der offenen Struktur nicht sinnvoll an das Modell anpassen, und das Risiko ist groß, dass beim Training nur ein schlechtes lokales Maximum erreicht wird. Dieses Risiko ist bei vorgegebenen Strukturen, etwa linearen Modellen, niedriger, da das *alignment* der Daten durch die Struktur erzwungen wird. Eine sinnvolle Vorbelegung der Parameter ist durch die freie Struktur aber gerade nicht möglich, so dass kein Vorwissen in die Modelle eingebracht werden kann. Ist aus diesem Grund die Leistung des Systems nicht befriedigend, so zeigt sich ein weiterer Nachteil der impliziten Strukturberechnung: Eine Analyse der Ergebnisse ist schwierig, weil ein Vergleich mit einer erwarteten Struktur nur dann möglich ist, wenn sich die Strukturen ähneln. Im Fall der Handschrifterkennung ist eine Analyse der Ergebnisse prinzipiell möglich und deshalb auch gewünscht. Deshalb wird in dieser Arbeit auf die Methode der impliziten Modellstrukturbestimmung verzichtet.

Streichen von Übergängen

Eine zur impliziten Strukturbestimmung verwandte Methode ist die *schrittweise* Vereinfachung der Struktur durch das iterative Entfernen von Übergängen zwischen den Zuständen [Vas96]. Es wird ebenfalls zunächst ein allgemeines Modell trainiert und anschließend dessen Übergangswahrscheinlichkeiten analysiert. Dieser Vorgang durchläuft aber mehrere Iterationen. In jedem Schritt wird der Übergang mit der geringsten Wahrscheinlichkeit aus der Struktur entfernt und das so entstandene Modell neu trainiert. Es werden auch ganze Zustände aus dem Modell entfernt, wenn deren Selbstübergang die geringste Wahrscheinlichkeit

aufweist; dadurch bestehen gewisse Ähnlichkeiten mit dem weiter unten vorgestellten *Model Merging*. Die endgültige Topologie gilt als erreicht, wenn bei der Entfernung weiterer Übergänge eine „substantielle“ Verringerung der Likelihood eintritt.

Der Unterschied zur impliziten Bestimmung besteht darin, dass sich das Modell nach dem Streichen eines Übergangs durch das Neutraining an die neue Struktur jeweils optimal anpassen kann. Die Methode erleichtert den expliziten Vergleich der verschiedenen Topologien, da die Likelihood der Daten beim Training jeder Iteration bestimmt wird und als Vergleichskriterium verwendet werden kann. Die Nachteile des Verfahrens – neben dem höheren Aufwand durch die Trainingsiterationen – bestehen aber gerade in diesem Kriterium, das keine objektive Bestimmung der idealen Struktur erlaubt. Evaluiert wurde die Methode auch nur mit künstlichen Daten von geringer Komplexität, einem linearen Modell mit drei Zuständen, dessen Struktur aber korrekt rekonstruiert werden konnte [Vas96].

Bedeutsam an dieser Stelle ist das allgemeine Prinzip des Verfahrens, sich iterativ einer optimalen Struktur zu nähern. Alle im weiteren vorgestellten Verfahren verwenden dieses Prinzip, das einen direkten Vergleich der verschiedener Topologien erlaubt. Sie unterscheiden sich – neben den Optimierungskriterien, die jedoch zum Teil austauschbar sind – in den jeweiligen Strategien, mit denen der Raum aller möglichen Topologien abgesucht wird.

Genetische Algorithmen

Der Raum der möglichen Topologien kann sehr unübersichtlich sein, und entsprechend schwierig ist es, einen geeigneten Startpunkt für die Suche nach der besten Topologie zu finden. Genetische Algorithmen versprechen in diesem Fall eine Lösung für die Optimierungsaufgabe. Sie wurden in der Spracherkennung für die Bestimmung der besten Struktur von Wortmodellen eingesetzt [Tak97]. Die Struktur wurde auf links-rechts-Modelle eingeschränkt; neben den Verbindungen direkt aufeinanderfolgender Zustände wurden zusätzliche Übergänge gesucht, mit denen sich die Modellierung verbessert. Die Optimierung wird bei Genetischen Algorithmen durch eine Fitnessfunktion, in diesem Fall die gemittelte Likelihood der Trainingsdaten, gesteuert Mit diesem Kriterium können auf einer unabhängigen Teststichprobe nicht so starke Verbesserungen der Erkennungsleistung erzielt werden wie bei der Trainingsstichprobe.

Auch in der Handschrifterkennung wurden Genetische Algorithmen zur Bestimmung der Modellstruktur eingesetzt [Sch00b]. Die Modelle repräsentieren hier einzelne Buchstaben und sind auf lineare links-rechts-Modelle festgelegt, deren Länge noch zu bestimmen ist. Da für sämtliche Modelle diese Zustandsanzahl gleichzeitig zu bestimmen ist, entsteht eine Komplexität, die mit Genetischen Algorithmen beherrscht werden soll. Als Fitnessfunktion dient die Erkennungsrate auf einem unabhängigen Testdeck, wodurch für eine geeignete Generalisierung der Ergebnisse gesorgt ist. Die Ergebnisse sind allerdings unbefriedigend, da das Verfahren aufgrund der Suchstrategie schnell lokale Optima findet und dadurch stark von den Anfangswerten abhängig ist. Durch zu kleine Trainings- und Teststichproben wird

gleichzeitig von einer Überadaption an die Daten berichtet, da durch die hohen Rechenzeitanforderungen keine Cross-Validation durchgeführt werden kann. Die besten Ergebnisse werden schließlich mit einem Verfahren erzielt, das die mittlere Merkmalslänge für die Bestimmung der Zustandsanzahl verwendet.

Letztlich verhindern die ausufernden Rechenzeiten der einzelnen Iterationen den Einsatz Genetischer Algorithmen, da für ein gutes Ergebnis die Berechnung vieler Generationen vorausgesetzt wird. Die Vorgehensweise sollte also sein, den Raum der Topologien so weit wie möglich einzuschränken, so dass eine systematischere Suche möglich wird.

10.1.3 Iterative Methoden

Die optimale HMM-Struktur lässt sich nur durch die Bewertung und den Vergleich vieler verschiedener Strukturen bestimmen. Für die Bewertung der Topologie ist in jeder Iteration die Bestimmung der HMM-Parameter notwendig, was die Suche sehr zeitaufwändig macht. Man ist aber auf iterative Methoden angewiesen und muss sich deshalb nun darauf konzentrieren, die Suche möglichst systematisch und zielgerichtet durchzuführen. Der hohe Rechenaufwand kann durch zwei Methoden vermindert werden. Eine Möglichkeit ist, die Anzahl der Iterationen gering zu halten und sie zumindest abschätzbar zu machen. Die andere besteht darin, die Kosten für die Neubewertung der Topologie in jeder Iteration zu minimieren. Sind sich die Strukturen aufeinanderfolgender Iterationen ähnlich, so muss zum Beispiel nicht unbedingt ein vollständiges Parametertraining durchgeführt werden.

Die generelle Vorgehensweise bei den verschiedenen vorgestellten Algorithmen besteht darin, dass zunächst eine Topologie als Startpunkt vorgegeben wird, die dann kontinuierlich verbessert wird. Dies wird durch zwei geschachtelte Iterationen realisiert:

- In der äußeren Schleife wird die Topologie variiert, in ihr findet die eigentliche Suche statt. Die verwendete Suchstrategie wird durch zwei Festlegungen definiert. Es wird spezifiziert, welche Änderungen in jeder Iteration an der Struktur durchgeführt werden, und zudem wird ein *Bewertungskriterium* definiert, das den Abbruch der Schleife steuert und so die endgültige Topologie definiert.

- Die innere Schleife hat die Aufgabe, die verschiedenen Änderungsalternativen zu bewerten, also die konkreten Zustände zu bestimmen, bei denen eine Änderung der Struktur den höchsten Gewinn bringt. Dazu muss ein geeignetes *Auswahlkriterium* definiert sein. Zum Teil ist für dessen Berechnung eine Neubestimmung der Parameter und die Bewertung der Trainingsdaten erforderlich.

Möglicherweise können in beiden Schleifen unterschiedliche Kriterien für die Auswahl der Modifikation und die Bewertung der Topologie gewählt werden. Mit dem Auswahlkriterium wird die Suche gesteuert, seine Wahl beeinflusst die Geschwindigkeit der Konvergenz. Eine gute Wahl dieses Kriteriums sorgt für eine schnelle Konvergenz, es muss aber sichergestellt sein, dass das Verfahren überhaupt konvergiert, also zumindest ein lokales Maximum des

Bewertungskriteriums gefunden werden kann. Die Auswahlkriterien müssen somit an die Bewertungskriterien angepasst sein.

Bei der Wahl der Ablaufsteuerung der äußeren Schleife kann man prinzipiell drei verschiedene Vorgehensweisen unterscheiden:

- Es wird mit einem minimalen Modell begonnen, das eventuell aus nur einem einzigen Zustand besteht. In jeder Iteration erhöht sich die Anzahl der Zustände um eins.
- Der Start erfolgt mit einem Modell, das sämtliche Trainingsdaten modelliert, also eine maximale Anzahl von Zuständen besitzt. Diese wird in den nachfolgenden Iterationen kontinuierlich verringert.
- Als Kompromiss dieser Verfahren startet man mit einem *plausiblen* Modell. Dieses wird dann modifiziert, indem entweder Zustände entfernt oder neue Zustände hinzugefügt werden.

In jedem der Fälle besteht die Abbruchbedingung darin, dass im Vergleich zum System der letzten Iteration keine Verbesserung des gegebenen Bewertungskriteriums erreicht werden kann. Dieses Vorgehen kann noch verfeinert werden, so dass auch mehrere Schritte ohne Verbesserung der Bewertung erlaubt sind.

Das iterative Vorgehen hat verschiedene Vorteile. Zum einen ist in jedem Schritt ein Vergleich mit der vorherigen Struktur möglich, was eine Analyse der Ergebnisse erleichtert. Durch die Information über die Strukturänderungen erhält man Einblicke in das Wesen der gegebenen Trainingsdaten. Zudem kann man durch die Vorgabe der Auswahlkriterien in einfacher Weise auf die Ausbildung der Struktur Einfluss nehmen.

Fortlaufende Zustandsteilung

Für die Clusterung von *Allophonen* in der Spracherkennung wurde ein Algorithmus vorgestellt, der ausgehend von einem primitiven Modell die Zustände schrittweise aufspaltet [Tak92]. Die Methode wird als *Successive State Splitting* bezeichnet. Zur Modellierung werden HMMs mit verbundenen Zuständen verwendet. Die Clusterung der Allophone erfolgt parallel zur Berechnung der verbundenen Zustände und zur Bestimmung der Parameter.

Interessant ist, welches Kriterium für die Auswahl der aufzuspaltenden Zustände gewählt wird, und welche Bedingung an die endgültige Topologie gestellt wird: Der aufzuspaltende Zustand wird anhand der Varianz seiner Verteilung ausgewählt. Dazu werden beim Training der Parameter alle Zustands-Emissionen durch zwei Gaußverteilungen modelliert und deren Abstand ausgewertet. Die Entscheidung, ob der Zustand in Richtung der Zeitachse aufgespaltet wird oder kontextuell – als paralleler Pfad – wird anhand des Likelihoodgewinns getroffen. Die Suche ist abgeschlossen und die endgültige Topologie definiert, sobald eine vorgegebene Anzahl von Zuständen erreicht ist. Für diese Topologie erfolgt ein abschließendes Training mit nur einer Gaußverteilung.

Problematisch an diesem Vorgehen ist zum einen, dass die Likelihood nur lokal maximiert wird. Zum anderen ist die Breite der Verteilung nicht unbedingt ein gutes Maß für unterschiedliche Aussprachevarianten, sondern gibt eher die natürliche Varianz bei verschiedenen Sprechern wieder. Deshalb ist es sinnvoll, die Breite der Verteilung durch ein Maximum-Likelihood-Kriterium als Auswahlkriterium zu ersetzen [Sin96, Ost97]. Dieses Kriterium wird sowohl für die Wahl des aufzuspaltenden Zustands, als auch für die Art der Aufspaltung (zeitlich oder kontextuell) verwendet. Im Anschluss an jede Aufspaltung erfolgt ein Neutraining der betroffenen Parameter. Wieder wird eine feste Anzahl von Zuständen vorgegeben.

Modell-Zusammenlegung

Zum Bilden geeigneter Modelle in der Spracherkennung wird aber auch der entgegengesetzte Ansatz verwendet [Sto94]: Der Start erfolgt mit einem Modell, das sämtliche Merkmalsvektoren der Trainingsdaten mit einzelnen Zuständen repräsentiert und damit eine bestmögliche Beschreibung der Trainingsmenge bietet, aber keinerlei Generalisierung zulässt. Dann werden in Schritten die Zustände des Modells paarweise zusammengefasst, eine Methode, die als *Model Merging* bezeichnet wird. Das Kriterium für die Wahl der zusammenzufassenden Zustandspaare ist identisch zur Bewertung der Topologie und ergibt sich aus dem in Kapitel 9 beschriebenen Ansatz der Bayes'schen Modellwahl. Es werden diejenigen Zustände ausgewählt, bei denen sich die Bewertung am meisten erhöht. Diese Berechnung kann effizient durchgeführt werden, wenn davon ausgegangen wird, dass sich der Viterbi-Pfad bei der Zusammenlegung zweier Zustände nicht ändert.

Eine weitere Anwendung erfährt diese Methode bei der Entwicklung von Modellen zur Kennzeichnung natürlicher Sprache, dem *part-of-speech tagging* [Bra98]: Werden zum Beispiel n-Gramm Sprachmodelle durch HMMs dargestellt, dann entspricht die Topologiebestimmung der Festlegung der Kontextlänge. Die Zustände für eine Zusammenlegung werden danach ausgewählt, bei welcher Kombination die Likelihood am wenigsten verringert wird. Für die Wahl der Topologie werden verschiedene Kriterien genannt: neben der Likelihood auch eine feste Anzahl von Zuständen oder Bayes'sche Ansätze.

Die Hauptanwendung des *Model Merging* liegt bei diskreten HMMs. Bei kontinuierlichen Daten führt sie zu Schwierigkeiten, wenn für das anfängliche Modell Zustände gebildet werden sollen, die einzelne Datenpunkte modellieren [Li00a]. Das größere Problem der Methode ist aber, dass mit der Verfügbarkeit vieler Trainingsdaten das ursprüngliche Modell beliebig groß wird. Ist die Anzahl der Trainingsdaten l, so wächst der Rechenaufwand aufgrund sämtlicher zu prüfender Kombinationen und dem Neutraining der Parameter wie $O(l^4)$. In dieser Hinsicht ist die Methode des Aufspaltens der Modelle klar im Vorteil, da die Modelle im Verhältnis zu der Anzahl der Daten immer viel kleiner sind. Eine Lösung des Problems besteht in einer inkrementellen Verarbeitung der Trainingsdaten [Sto94], eine Methode, die sich auch für online-Nachadaptionen eignet. Es wird nur eine vorgegebene Anzahl von Trainingsdaten in das anfängliche Modell aufgenommen, bevor die ersten Zustände

zusammengelegt werden, erst anschließend werden weitere Daten zum Modell hinzugefügt. Im Extremfall wird das Modell durch das kontinuierliche Einfügen einzelner Beispiele aufgebaut [Tho86]. Es entsteht eine gewisse Ähnlichkeit zur Aufspaltung der Zustände, bei der ebenfalls ein einfaches Modell immer komplexer wird. Der Unterschied besteht darin, dass sich die Komplexität hier nur erhöht, wenn neue Daten hinzugefügt werden.

Ein etwas anderer Ansatz und ein weiteres Kriterium für die Wahl der Zustände findet sich in einer Anwendung, bei der die Modelllänge einzelner Silbenmodelle für die Spracherkennung bestimmt werden soll [Ham99]: Hier wird mit einem einzigen Modell von ausreichender Länge gestartet, das durch Zusammenlegen der Zustände so lange verkürzt wird, bis das Optimum erreicht ist. Die Zusammenlegung von Zuständen wird jetzt über die Ähnlichkeit der modellierten Gaußverteilungen gesteuert, die über den Bhattacharyya-Abstand

$$D_{\text{bhat}} = \frac{1}{8}(\mu_1 - \mu_2)^T \left(\frac{K_1 + K_2}{2}\right)^{-1} (\mu_1 - \mu_2) + \frac{1}{2}\log \frac{|\frac{K_1+K_2}{2}|}{\sqrt{|K_1| \cdot |K_2|}} \tag{10.1}$$

berechnet wird. Dieser ist ein Maß für die Trennbarkeit von Klassen und außerdem eine obere Grenze für den optimalen Bayes'schen Klassifizierfehler [Mak96]. Als Abbruchkriterium für die Optimierung wird die Verbesserung der Topologiebewertung nach dem Bayes'schen Informationskriterium (9.10) herangezogen [Ham98].

Kombination von Teilung und Zusammenlegung

Es wird auch die dritte der genannten Vorgehensweisen, die Kombination von Teilungen und Zusammenlegungen der Zustände, vorgestellt [Bra96]. Anwendung ist auch hier die Kennzeichnung von gesprochener Sprache.

Für die Wahl, welche Methode in einem Schritt konkret angewendet werden soll, wird folgende Überlegung angestellt: Eine Generalisierung der Daten durch die Verbindung von Zuständen ist dann sinnvoll und notwendig, wenn nur wenige Daten vorhanden sind. Auf der anderen Seite erlaubt das Vorhandensein vieler Daten eine Spezialisierung des Modells durch das Aufspalten von Zuständen. Der *Schätzfehler* des Modells wird gering gehalten, wenn das Kriterium für die Wahl der Modellmodifikation durch die Anzahl der Trainingsdaten für die einzelnen Zustände gegeben ist. Das Kriterium ist leicht zu implementieren, da diese Zahl im Training der HMM-Parameter ermittelt wird. Die Topologie selber wird über eine Cross-Validation der Erkennungsleistung bewertet.

Der große Vorteil der Kombination von *Merging* und *Splitting* besteht vor allem in der schnellen Konvergenz des Verfahrens, wenn die Starttopologie sinnvoll vorgegeben ist. Von einer sinnvollen Vorgabe kann immer dann ausgegangen werden, wenn die Anwendung der Topologieoptimierung darin liegt, ein gegebenes, funktionstüchtiges System zu verbessern. Gleichzeitig ergibt sich die Garantie, dass die Leistung des optimierten Systems gleich hoch oder besser ist.

Auswahlkriterien

Sind bei der Suche nach der besten Topologie sowohl Teilungen als auch Verschmelzungen von Zuständen zulässig, so wird die Suche nach einem geeigneten Auswahlkriterium schwieriger als im Fall von nur einer Modifikationsmöglichkeit. Bevor die an der Topologieänderung betroffenen Zustände überhaupt bestimmt werden können, muss man zunächst die Art der Änderung festlegen. Dazu ist es notwendig, ein einheitliches und vergleichbares Kriterium für Teilung und Verbindung von Zuständen zu besitzen. Die bisher gezeigten Kriterien lassen sich in drei Kategorien aufteilen:

- *Globale Maße* messen die Auswirkungen der Topologieänderung auf die Gesamtheit des Systems. Beispiele sind die Likelihood der Trainingsdaten oder andere Wahrscheinlichkeiten, die über die Bayes'sche Modellwahl definiert wurden.
- Angenommene *Modellierungsfehler* werden lokal identifiziert. Indizien für falsche Modellierung sind die Varianz der Daten innerhalb der einzelnen Zustände, die Entropie der Symbolemissionen oder der Abstand zwischen verschiedenen Zuständen des Modells.
- *Schätzfehler* der Modellparameter können durch die Anzahl der zum Training zur Verfügung stehenden Daten bestimmt werden.

Wichtig für jedes zu entwickelnde Verfahren für die Topologieoptimierung ist, dass das Auswahlkriterium auf das Optimierungskriterium abgestimmt ist, so dass der Algorithmus konvergiert. Unmittelbar ist dies nur durch die erstgenannten globalen Maße gegeben.

10.2 Bestimmung der Struktur der Buchstaben-HMMs

In diesem Abschnitt wird beschrieben, welche Besonderheiten bei der Schriftmodellierung auftreten, wie die in Abschnitt 8.2 vorgestellten Buchstabenmodelle also konkret optimiert werden.

10.2.1 Clusterung und Modellierung

Die Aufgabe bei der Bestimmung der genannten Parameter scheint zunächst einmal darin zu bestehen, die unbekannten Schreibweisen zu identifizieren, also in einer unüberwachten Clusterung der Grapheme in Allographen. Es gibt aber einen entscheidenden Unterschied zwischen dieser Clusterung und den in Abschnitt 10.1.1 erwähnten bekannten Clusterverfahren. Die Variabilität der Modelllänge der Allographen ist ein wesentliches Merkmal für die Repräsentation der Cluster und muss bei der Clusterung berücksichtigt werden. Die Bestimmung der Pfadanzahl und deren Länge sind gleichberechtigt, wie es die Repräsentation in einem gemeinsamen Buchstabenmodell auch nahe legt. Es können zwar Clusterung und

Modellierung als zwei Teilaspekte bei der Optimierung identifiziert werden, sie müssen aber beide gleichzeitig berücksichtigt werden.

Der direkte Ansatz besteht darin, die Optimierung beider Freiheitsgrade in zwei ineinander geschachtelten Schleifen durchzuführen [Li00b, Li00a]. Sowohl für die Anzahl der Cluster, als auch für die Anzahl der HMM-Zustände werden Kriterien nach der Bayes'schen Modellwahl herangezogen. In der äußeren Schleife wird die Anzahl der Cluster kontinuierlich erhöht, in der inneren die Anzahl der Zustände, jeweils so lange, bis die durch das Kriterium gewählte Modellwahrscheinlichkeit wieder sinkt. Dabei erfolgt in jedem Schritt jeweils eine Zuordnung der Trainingsdaten zu den Clustern sowie ein vollständiges Training der HMM-Parameter. Die Anzahl der Cluster und der jeweiligen Zustände werden bei diesem Algorithmus also gleichzeitig optimiert. Dadurch ist garantiert, dass die gegenseitige Abhängigkeit der beiden Parameter berücksichtigt wird.

Der vorgeschlagene Algorithmus ist jedoch mit so hohem Rechenaufwand verbunden, dass er für eine Anwendung hier nicht praktikabel ist. Die Komplexität der Berechnung lässt sich aber dadurch senken, dass die beiden freien Parameter voneinander unabhängig optimiert werden. Bei der Berechnung der Pfadlänge sorgt man sich nicht um die Zuordnung zu den verschiedenen Pfaden, umgekehrt geht man bei der Bestimmung der Anzahl der Pfade davon aus, dass die Länge der bestehenden Pfade optimal ist. Eine globale Optimierung beider Parameter ist trotzdem möglich, indem iterativ beide Optimierungsschritte wechselweise durchgeführt werden. Man betrachtet beide Probleme unabhängig voneinander und entwickelt für jedes eine optimale Strategie, wodurch erzielte Verbesserungen den einzelnen Verfahren auch zugeordnet werden können.

Ein weiterer Grund für die getrennte Behandlung der Parameter ist neben der Rechenzeit auch die Verschiedenartigkeit der Probleme. Es wird sich zeigen, dass es sinnvoll ist, für beide Optimierungen unterschiedliche Kriterien anzulegen. Für die Pfadlänge ist die Likelihood der Daten als Kriterium geeignet, da durch ihre Maximierung in der konkreten Anwendung weder ganz kurze noch ganz lange Pfade favorisiert werden. Anders sieht es bei der Anzahl der Pfade aus; hier würde die Anzahl der Modelle beliebig wachsen, was den Einsatz von Kriterien der Bayes'schen Modellwahl notwendig macht.

10.2.2 Segmentierung der Grapheme

In der Handschrifterkennung geht das Problem der Modellbestimmung über die Aufgabe der Clusterung und Modelllängenschätzung hinaus: Es müssen die Modelle *sämtlicher* Grapheme optimiert werden, wobei die Trainingsdaten jedoch keine explizite Segmentierung dieser Grapheme bereitstellen. Stattdessen erfolgt diese Segmentierung in der Regel implizit, sowohl im Training, als auch während der Erkennung – eine Eigenschaft, die gerade die Stärke der Modellierung mit HMMs ausmacht. Für die Optimierung der Modellstruktur stellt diese implizite Segmentierung jedoch ein Problem dar, weil sich die Segmentierung mit jeder Änderung der Modellierung ändern kann.

Es ist deshalb wichtig, dass alle Grapheme *gleichzeitig* modelliert werden, damit bei der Optimierung der Modelle auch die Qualität der Segmentierung steigt, weil diese auf die Modelle zurückwirkt. Bei einem iterativen Vorgehen können diese Voraussetzungen erfüllt werden. Wichtig ist aber vor allem, dass ein funktionsfähiges System am Ausgangspunkt steht, damit die Suche nach den Optimalitätskriterien nicht in einem lokalen Maximum endet, das weit vom globalen Maximum entfernt ist und einer schlechten Segmentierung entspricht. Es ist zum Beispiel nicht möglich, mit einem minimalen System zu beginnen und die Anzahl der Cluster und Zustände immer nur zu erhöhen. Der Start sollte mit einer zumindest plausiblen Modellierung erfolgen, und die Schritte der Iteration sollten sowohl Vergrößerungen als auch Verkleinerungen des Schriftmodells zulassen.

10.2.3 Bestimmung des Codebuchs

Das konkret betrachtete Handschrifterkennungssystem modelliert die Beobachtungen mit semi-kontinuierlichen HMMs. Die Besonderheit ist, dass die Normalverteilungen der Klassen aus den Zuständen selber gebildet werden und somit jedem Zustand direkt eine Klasse zugeordnet ist (vgl. Abschnitt 5.4). Diese Zuordnung geht bei jeder Änderung der Modelltopologie verloren. Man muss deshalb nach jeder Topologieänderung ein neues Codebuch berechnen, um die Trennschärfe und die Güte des neuen Schriftmodells auch mit dem geeigneten Codebuch messen zu können.

Kapitel 11

Optimierung der Zustandspfade

Für alle Allographen soll die optimale *Länge der Zustandspfade* der linearen Buchstaben-HMMs bestimmt werden. Es wird angenommen, dass die verschiedenen Schreibweisen der Grapheme bestimmt und festgelegt sind. Die Optimierung erfolgt unabhängig von der Clusterung der Allographen und lässt sich einfach für eine Verbesserung des im ersten Teil dieser Arbeit beschriebenen Erkennungssystems einsetzen.

Eine separate Behandlung der einzelnen Allographen wie in [Li00a] ist wegen der Wechselwirkungen der Buchstabenmodelle nicht möglich. Da immer nur ganze Wortbilder behandelt werden, wirkt auch im Training die Modellierung der Länge eines Graphems auf die benachbarten Buchstaben zurück. Diese Besonderheit der Anwendung wird bei der Entwicklung des konkreten Optimierungskriteriums und des Algorithmus berücksichtigt.

11.1 Maximum-Likelihood-Kriterium

Bei der Bayes'sche Modellwahl geht es darum, einen Ausgleich zwischen einer guten Beschreibung der Daten und der Einfachheit des Modells zu erreichen (vgl. Kapitel 9). Durch die Beschränkung der Modellgröße soll die Generalisierbarkeit des Modells gewährleistet sein. Sie ist durch eine feste Zuordnung der Schreibweisen von der Pfadlängenoptimierung aber gar nicht betroffen. Durch die vorgegebene *1-Pfad-Struktur* der HMMs und die Wechselwirkung der Modelle untereinander nehmen die Modelllängen von selber ein Gleichgewicht ein.

Zum einen ist durch die Anzahl der Zustände die Mindestbreite des modellierten Buchstabens gegeben. Eine zu hohe Modellkomplexität wird automatisch schon dadurch bestraft, dass die Zustände gar nicht mehr den richtigen Beobachtungen zugewiesen werden können. Zum anderen bewirkt der Einfluss der benachbarten Buchstaben, dass die Anzahl der HMM-Zustände die mittlere Breite der Grapheme von selber widerspiegeln wird. Aus diesen Gründen ist es nicht notwendig, explizit Kriterien der Bayes'schen Modellwahl anzuwenden.

Ein geeignetes Kriterium ist also ausschließlich eine möglichst gute Anpassung der vorgegebenen Struktur an die Beobachtungen. Das entspricht der Zielsetzung beim normalen

HMM-Training, nur dass neben den Parametern jetzt auch die Struktur zu bestimmen ist. Auch in diesem Fall kann also die maximale gemittelte Likelihood aller Trainingswörter als Kriterium herangezogen werden. Damit wird letztlich die Grundstruktur des Trainingsverfahrens übernommen: Die Parameter verschiedener Topologien werden mit dem Baum-Welsh-Training geschätzt, anschließend wird diejenige Topologie als optimal ausgewählt, deren mittlere Likelihood maximal ist.

11.2 Optimierung mit parallelen Pfaden

Die Komplexität der Optimierungsaufgabe ist wegen der kombinatorischen Vielfalt der möglichen Topologien sehr hoch. Wenn man für jeden Pfad nur eine endliche Anzahl m möglicher Pfadlängen zulässt, gibt es bei p verschiedenen Graphemmodellen bereits m^p verschiedene Topologien. Da für typische Alphabete $p \geq 26$ gilt, ist an eine Bewertung sämtlicher Kombinationen nicht zu denken. Es bietet sich deshalb an, die Länge der einzelnen Pfade unabhängig voneinander zu optimieren [Sch00b]. Die Anzahl der notwendigen Berechnungen verringert sich bei dieser Methode auf $m \times p$. Allerdings wird die Wechselwirkung der Allographmodelle nicht berücksichtigt.

Die Optimierung muss deshalb zumindest in mehreren Iterationen durchgeführt werden. Der Nachteil besteht dabei nicht im erhöhten Berechnungsaufwand. Viel größer ist die Gefahr, dass nur ein schlechtes lokales Optimum erreicht wird, das weit vom gewünschten Ergebnis entfernt ist. Denn durch den Einfluss der Modelle auf die Segmentierung können keine Annahmen über das Verhalten der Likelihood gemacht werden, wenn die Struktur der anderen Modelle weit von einer guten Schriftmodellierung entfernt ist. Man umgeht dieses Problem, indem sämtliche Pfadlängen in einer Iteration gleichzeitig bewertet und angepasst werden. Das gelingt durch die Verwendung paralleler Pfade.

11.2.1 Algorithmus

Der hier vorgestellte Algorithmus arbeitet iterativ. Er berechnet in jeder Iteration gleichzeitig verschiedene Alternativen für jedes Modell und wählt anschließend zwischen ihnen aus. Er ist in Abbildung 11.1 skizziert.

Der Ausgangspunkt jeder Iteration ist ein funktionsfähiges Erkennungssystem mit trainierten Parametern. Mit diesem System werden den Trainingsdaten eindeutig die Schreibvarianten der einzelnen Buchstaben zugewiesen; das ist durch die Auswertung des Viterbi-Pfades bei der Erkennung möglich. Nach der Zuordnung und einer entsprechenden Markierung der Trainingsdaten werden die Buchstabenmodelle in mehrere 1-Pfad-Modelle aufgespalten, die jeweils nur die entsprechende Schreibvariante modellieren. Nur diese Modelle werden im weiteren betrachtet.

Bei den neuen Allographmodellen wird jetzt der einzige Pfad der Länge s erneut in mehrere Pfade aufgespalten, die nun aber von verschiedener Länge sind und $s-1$, s und $s+1$

Initialisierung

Start mit plausiblen Pfadlängen, Training der HMM-Parameter.

Iteration

Feste Zuordnung der Schreibweisen.
Einfügen von zusätzlichen, parallelen Pfaden mit jeweils ± 1 Zuständen.
Baum-Welsh-Training der erweiterten Modelle.
Auswahl der besten Pfadlängen.
Nachtraining der HMM-Parameter ohne feste Zuordnung.
Endgültige Auswahl der Veränderungen

Abschluss

Abbruch bei geringer Verbesserung oder keiner Veränderung der Längen.

Abbildung 11.1: Algorithmus zur Optimierung der Zustandspfadlängen

Zustände besitzen. Jeder der neuen Pfade repräsentiert denselben Allographen, nur mit unterschiedlicher Modellierung. Man darf die hier besprochenen parallelen Pfade also nicht mit denen der Buchstabenmodelle verwechseln, die unterschiedliche Gestalten der Buchstaben repräsentieren.

Die Parameter der so erweiterten Modelle werden durch ein EM-Training mit der neu markierten Trainingsstichprobe belehrt. Durch eine Analyse der berechneten Parameter kann danach derjenige Pfad bestimmt werden, der die höchste Likelihood der Trainingsdaten erzeugt. Es ist der Pfad mit der höchsten Eingangswahrscheinlichkeit, da sie als die Häufigkeit interpretiert werden kann, mit der dieser Pfad die Trainingsdaten mit der höheren Wahrscheinlichkeit emittiert hat. Seine Länge ist das Optimum für das Allographmodell in der aktuellen Iteration. Er wird als neues Modell ausgewählt und ersetzt in der nächsten Iteration den alten Modellpfad. Das Grundprinzip des beschriebenen Optimierungsschritts ist in Abbildung 11.2 dargestellt.

Nach der Auswahl der neuen Modelllängen werden die Allographmodelle wieder zu vollständigen Graphemmodellen zusammengesetzt und diese Modelle mit dem Baum-Welsh-Training erneut belehrt. Das ist notwendig, da sich mit der Modellierung auch die Zuordnung der Schreibweisen ändern kann; sie wird deshalb in jeder Iteration neu erstellt. Eine weitere Notwendigkeit für das Training der vollständigen Buchstabenmodelle ist das Abbruchkriterium der Iteration. Es ist entweder dadurch gegeben, dass sich keine Veränderung an der Modellierung mehr ereignet, oder durch die Bewertung der Likelihood der Trainingsdaten. Verbessert sich diese nicht mehr ausreichend, so ist die Optimierung abgeschlossen. Es lassen sich aber nur die Modellwahrscheinlichkeiten der *vollständigen* Buchstabenmodelle sinnvoll und konsistent vergleichen.

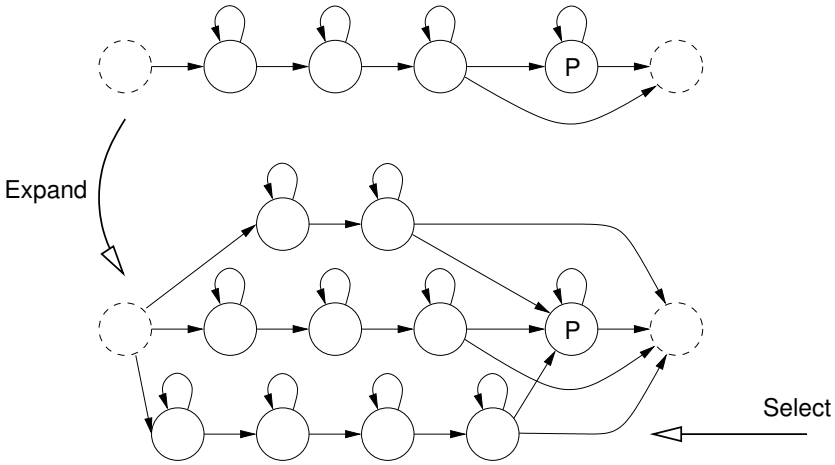

Abbildung 11.2: Lernen der Modelllänge. Zunächst wird ein Allographmodell expandiert, indem parallele Pfade hinzugefügt werden, die alternative Modelllängen realisieren. Anschließend erfolgt ein Neutraining der Parameter, der beste Pfad wird ausgewählt und stellt das neue Modell dar. Der Vorgang wird iterativ durchgeführt und optimiert alle Buchstabenmodelle zur selben Zeit.

11.2.2 Transformation der Pfade

Bei der Erweiterung der Allographmodelle müssen die neu generierten parallelen Pfade mit geeigneten Werten initialisiert werden. Die richtige Wahl der HMM-Parameter für den verlängerten und den verkürzten Pfad hat starken Einfluss auf die Geschwindigkeit der Konvergenz des Baum-Welsh-Trainings, das einen hohen Anteil des Zeitaufwands bei der Optimierung beansprucht. Außerdem ist es wichtig, in den parallelen Pfaden vergleichbare lokale Optima zu finden; das wird durch eine geeignete Initialisierung der Parameter begünstigt.

Zwei Verfahren werden im weiteren verfolgt: Beim ersten werden die Parameter benachbarter Zustände gemischt, beim zweiten werden ausgewählte Zustände zusammengelegt bzw. dupliziert.

Mischen

Der hier verfolgte Ansatz besteht darin, die Emissions- und Übergangswahrscheinlichkeiten der neuen Zustände aus denen des ursprünglichen Modells zu mischen. Der Vorgang ist in Abbildung 11.3 dargestellt. Man nimmt an, dass jeder Zustand den gleichen Beitrag zum vollständigen Modell liefert und damit eindeutig bestimmt ist, wie groß die korrespondierenden Teilbereiche der Zustände sind. Damit ergibt sich eine einfache Berechnungsvorschrift. In einem Pfad mit N Zuständen bezeichne P_n entweder ein Emissionsgewicht $b_n(k)$ des n-ten Zustands oder eine Übergangswahrscheinlichkeit a_{nm} aus diesem Zustand. Dann lassen sich die entsprechenden Wahrscheinlichkeiten P_n^- der Zustände in dem um einen Zustand verkürzten Pfad mit

$$P_n^- = \frac{N-n}{N} \cdot P_n + \frac{n}{N} \cdot P_{n+1}, \tag{11.1}$$

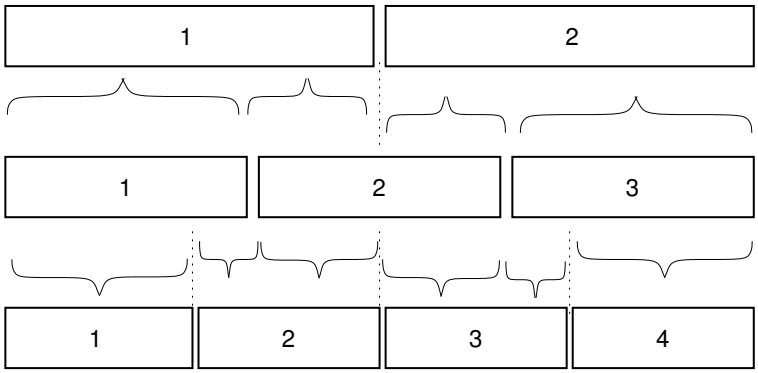

Abbildung 11.3: Initialisierung der Pfade durch Mischen. Es wird postuliert, dass jeder Zustand den gleichen Anteil am Gesamtmodell repräsentiert. Die Parameter der neuen Modellpfade lassen sich dann durch Mischen der Wahrscheinlichkeiten des Ursprungsmodells bestimmen.

und die P_n^+ der Zustände im verlängerten Pfad mit

$$P_n^+ = \frac{n-1}{N} \cdot P_{n-1} + \frac{N-n+1}{N} \cdot P_n \tag{11.2}$$

berechnen.

Für die Vorgabe der Eingangswahrscheinlichkeiten muss die Besonderheit der Aufgabenstellung beachtet werden. Gerade in der ersten Trainingsiteration ist zu erwarten, dass die erzeugten neuen Pfade schlechtere Ergebnisse erzielen als der Ursprungspfad. Dies darf keinen Einfluss auf die Wahl der Pfadlänge haben, deshalb muss sichergestellt werden, dass im gesamten Training der ursprüngliche Pfad nicht häufiger trainiert wird als die neu generierten Modellvarianten. Während des gesamten Trainings besitzen die Eingangswahrscheinlichkeiten also den konstanten Wert

$$\pi_i = \frac{1}{3}, \qquad i = 1 \ldots 3. \tag{11.3}$$

Die Eingangswahrscheinlichkeiten werden also nicht mittrainiert. Erst nach der Auswahl der Pfadlänge werden sie für die Auswertung der neuen Modellgüte geschätzt.

Verschmelzen und Duplizieren

Im zweiten Verfahren werden nicht sämtliche alten Zustände in die Generierung der neuen Zustände miteinbezogen, sondern es werden vorher jene Zustände bestimmt, bei denen eine Entfernung oder Duplikation schon ohne Nachtraining eine Verbesserung der Modellierung verspricht. Diese Zustände werden auf folgende Weise ausgewählt:

- Für die Verkürzung des Pfades werden zwei benachbarte Zustände verschmolzen. Es werden die im Pfad aufeinander folgenden Zustände gewählt, deren Abstand am geringsten ist. [1] Für die Kombination der Zustände wird deren Länge berücksichtigt. Die

[1] Der Abstand von Zuständen wird im folgenden Kapitel 12 definiert.

Länge l_i eines Zustands s_i ist durch

$$l_i = \frac{a_{ii}}{1 - a_{ii}} \tag{11.4}$$

definiert, wobei a_{ii} die Wahrscheinlichkeit eines Selbstübergangs bezeichnet. Sie gibt die durchschnittliche Anzahl an Beobachtungen aus diesem Zustand an. Die Emissionsgewichte werden proportional zu den Längen verteilt, die Übergangswahrscheinlichkeiten berechnen sich entsprechend der Summe der Länge der Ursprungszustände s_i, s_j zu

$$a_{ii} = \frac{l_i + l_j}{1 + l_i + l_j}. \tag{11.5}$$

- Der Pfad wird verlängert, indem ein Zustand ausgewählt und einfach dupliziert wird. Die Wahl fällt auf den Zustand mit der höchsten Länge l_i. Die Emissionsgewichte werden einfach kopiert, der Selbstübergang der neuen Zustände ergibt sich aus der halben Länge des alten Zustands, die übrigen Parameter ergeben sich entsprechend.

Für die Eingangswahrscheinlichkeiten gilt dasselbe wie beim Mischen der Wahrscheinlichkeiten. Sie erhalten den konstanten Wert $\pi_i = \frac{1}{3}$.

11.3 Ergebnisse

Der Algorithmus zur Längenanpassung wurde auf alle acht Erkennungssysteme, die in Abschnitt 6.4.1 beschrieben wurden, angewendet. Im folgenden werden zunächst allgemein die Eigenschaften des Algorithmus untersucht, anschließend wird die konkrete Auswirkung auf die Erkennungsleistung der Systeme gemessen. Die Plausibilität der resultierenden Modelle wird anhand der Visualisierung der Ergebnisse überprüft [Sch03b].

11.3.1 Entwicklung der Modellierungsgüte

Abbildung 11.4 zeigt die iterative Entwicklung von Likelihood und Erkennungsrate am Beispiel des Erkennungssystems arabischer Schrift. Ausgangspunkt ist eine Modellierung sämtlicher Zeichen mit drei Zuständen; im Lauf der Iterationen wird die Modelllänge aller Allographen dann individuell bestimmt. Anhand der Kurven lassen sich unterschiedliche Eigenschaften des Modelllängen-Adaptionsprozesses zeigen:

- Zwischen Likelihood und Erkennungsrate besteht keine eindeutige Korrelation. Die Schwankungen in der Erkennungsleistung auf den Testdaten lässt sich mit der Entwicklung der Likelihood der Trainingsdaten nicht vorhersagen, die Erkennungsrate auf den Trainingsdaten ist dafür der bessere Indikator. Im folgenden Abschnitt wird sich zeigen, dass die Erkennungsrate auf den Trainingsdaten auch für die Wahl des besten Systems am günstigsten ist.

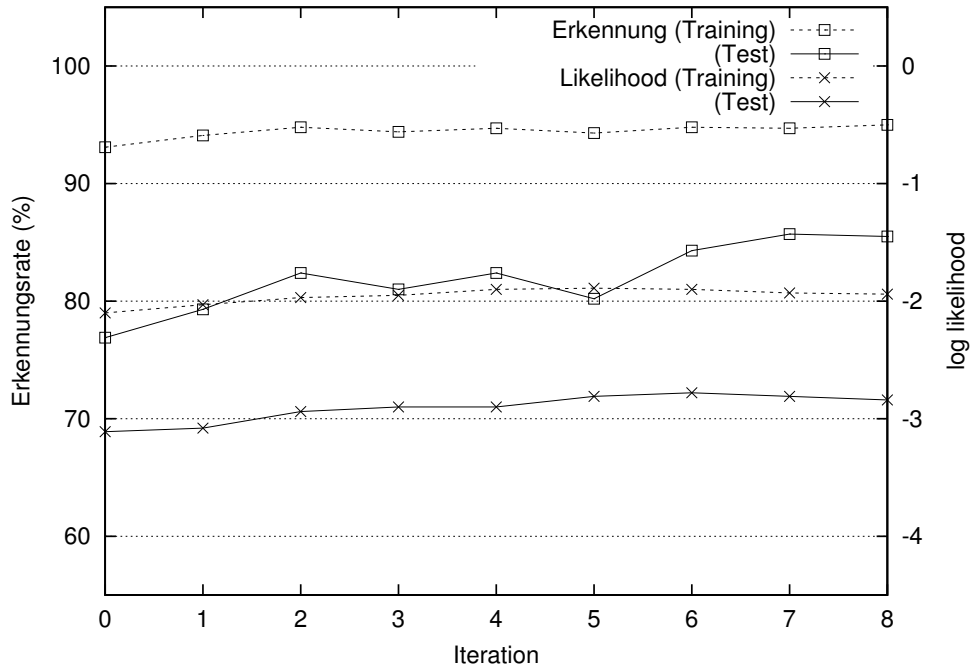

Abbildung 11.4: Entwicklung von Likelihood und Erkennungsrate während der Iterationen der Modelllängenanpassung am Beispiel des Systems zur Erkennung arabischer Schrift.

- Likelihood und Erkennungsrate nehmen lokal während der Iterationen auch wieder ab. Deshalb ist es sinnvoll, die Iterationen solange fortzusetzen, wie Änderungen an der Modelltopologie durchgeführt werden, und die optimale Topologie dann im Anschluss zu bestimmen.

- Die Fähigkeit des Systems zur Generalisierung verbessert sich im Lauf der Iterationen. Dies sieht man daran, dass der Abstand zwischen Trainings- und Testdaten geringer wird, sowohl bei der Erkennungsrate, als auch bei der Likelihood. [2]

Der vorgestellte Algorithmus konvergiert *nicht* gegen ein lokales Optimum, was sich im Experiment an der Abnahme der Likelihood auch der Trainingsdaten während der Iterationen zeigt. Die Veränderung der Topologie von jedem Modell wirkt auf die anderen Modelle zurück und beeinflusst die Segmentierung der Trainingsdaten. Verbesserungen einzelner Modelle können so insgesamt zu einer Verschlechterung des Gesamtmodells führen. Die Verbesserung der Generalisierung zeigt jedoch die prinzipielle Fähigkeit des Algorithmus, bessere Modelle zu erzeugen.

[2] Der Abstand der Werte zwischen Training und Test ist beim arabischen Erkennungssystem im Vergleich zu den anderen Systemen sehr hoch, das heißt es generalisiert relativ schlecht. Das liegt zum einen daran, dass die Trainingsdaten synthetisch und die Testdaten real sind, zum anderen an der Vorverarbeitung, die nicht speziell an die Eigenschaften arabischer Schrift angepasst ist.

Projekt	Startpunkt	Mischen		Verschmelzen & Duplizieren	
		Likelihood	Erkennung	Likelihood	Erkennung
Arab Emirate (Emirate)	76,9 %	85,5 %	85,5 %	83,2 %	82,2 %
Kanada (Adressen)	93,4 %	93,1 %	93,4 %	92,7 %	93,4 %
Deutschland (Adressen)	92,8 %	91,1 %	93,3 %	91,5 %	93,1 %
Deutschland (PLZ)	69,1 %	72,4 %	72,1 %	71,9 %	71,9 %
USA (Adressen)	81,2 %	81,6 %	82,7 %	81,5 %	82,6 %
USA (ZIP)	60,8 %	63,1 %	62,9 %	63,5 %	62,9 %
USA-CEDAR (Städte)	84,2 %	83,9 %	84,2 %	82,0 %	84,1 %
USA-CEDAR (ZIP)	58,2 %	59,3 %	59,5 %	61,8 %	61,8 %
Mittlere Erkennungsrate	77,1 %	78,8 %	79,2 %	78,5 %	79,0 %
Relative Verbesserung		+2,21 %	+2,72 %	+1,82 %	+2,46 %

Tabelle 11.1: Erkennungsraten nach Längenadaption für verschiedenen Adaptionsmethoden (Mischen, Verschmelzen und Duplizieren) und Auswahlkriterien (Auswahl nach Likelihood oder Erkennungsrate auf den Trainingsdaten).

11.3.2 Erkennungsraten

Für beide Initialisierungsmethoden der Pfadtransformation, *„Mischen“* und *„Verschmelzen und Duplizieren“*, wurde die Erkennungsleistung der Systeme gemessen. Die Erkennungsraten der Tests sind in Tabelle 11.1 aufgelistet. Die beste Topologie wurde auf zwei Arten aus den Iterationen ausgewählt: entweder nach der höchsten Likelihood, oder nach der höchsten Erkennungsrate, die auf den Trainingsdaten erzielt wird.

In nahezu allen Fällen ist es besser, die Topologie nach der Trainings-Erkennungsrate zu wählen, da sie den besten Indikator für die Erkennungsleistung des Systems darstellt. Mit dieser Methode kann eine mittlere Verbesserung der Erkennungsrate von 2,72 Prozent erreicht werden. Verlässt man sich hingegen auf die Likelihood, muss zum Teil sogar mit einer Verschlechterung des Systems gerechnet werden.

Die Systeme, bei denen die größten Verbesserungen erzielt werden, sind die Erkennungssysteme für arabischer Schrift und für Ziffern. Für arabische Schrift sind Merkmalsgenerierung und Modellierung des Ausgangssystems noch nicht manuell aneinander angepasst, so dass es den größten Spielraum für Verbesserungen des Modells bietet. Bei den Ziffernerkennern hingegen erlaubt die vergleichsweise einfache Segmentierung eine detailliertere Modellierung der Ziffernmodelle.

Die zwei verschiedenen Initialisierungsmethoden erzeugen vergleichbare Erkennungsraten, obwohl sie einen großen Einfluss auf die resultierenden Modelle haben, wie im nächsten Abschnitt gezeigt wird. Im weiteren wird wegen der im Schnitt besseren Ergebnisse ausschließlich die Methode *„Mischen“* verwendet.

11.3.3 Modelllänge

Einen auffälligen Unterschied zwischen den Initialisierungsmethoden gibt es bei der durchschnittlichen Länge der erzeugten Zustandspfade. Sie beträgt bei der Transformation durch „Verschmelzen und Duplizieren“ im Durchschnitt 3,46 Zustände und ist damit deutlich länger als die 3,27 Zustände bei der „Mischen“-Methode. Durch das Aufspalten der breitesten Zustände beim Duplizieren werden eher Modelle erzeugt, bei denen die einzelnen Zustände gleich große Anteile des Gesamtmodells erzeugen. Damit entstehen an breiteren Zeichenabschnitten leichter Folgen von ähnlichen Zuständen. Diese Redundanz in der Modellbeschreibung führt zu der schlechteren Erkennung im Vergleich zur „Mischen“-Methode, bei der sich die Zustandsmodellierung ausschließlich am Aussehen der Zeichensegmente orientiert.

Um einen tieferen Einblick in den Adaptionsalgorithmus zu erhalten, wurde ein weiteres Experiment durchgeführt. Neben den Standard-Erkennungssystemen wurden solche Systeme als Startpunkt der Iterationen gewählt, die nur zwei Zustände pro Modell besitzen. Es sollte getestet werden, ob in beiden Konfigurationen ähnliche Modelltopologien gefunden werden können. Die abschließenden Pfadlängen betragen 2,99 (Verschmelzen und Duplizieren) und 2,48 (Mischen). Sie sind deutlich geringer als beim Start mit drei Zuständen. Auch die Erkennungsraten liegen mit 74,4 % und 76,0 % deutlich unter den bisherigen Ergebnissen.

Es offenbart sich eine Schwäche des vorgestellten Algorithmus. Durch die übermäßig vereinfachte Modellierung sind Segmentierung und Erkennung der Trainingsdaten in den ersten Iterationen offensichtlich so schlecht, dass in den nachfolgenden Iterationen mit mehr Zuständen die Adaption in einem schwachen lokalen Maximum stecken bleibt. Der Startpunkt der Modellierung sollte deshalb zumindest so gut gewählt sein, dass die Qualität der ersten Segmentierung im Training zur Verfeinerung der Modellierung ausreicht.

11.3.4 Visualisierung

Die Visualisierung der erzeugten Modelle gibt einen guten Eindruck von den Effekten der Modelllängenadaption. In Abbildung 11.5 werden die adaptierten Zahlen des deutschen Postleitzahlen-Erkennungssystem gezeigt. Man sieht, dass die Zeichen viel detaillierter modelliert werden als in Abbildung 5.5 auf Seite 54. Auf den ersten Blick überraschend ist die Tatsache, dass nicht die „1“, sondern die „2“ am schmalsten modelliert ist. Es ist jedoch bei allen Zeichen deutlich, wie sich die benachbarten Zustände unterscheiden; die Komplexität der Zeichen ist gut abgebildet.

Dasselbe gilt für die Modelle arabischer Schrift, die in Abbildung 11.6 gezeigt werden. Man sieht eine Auswahl von drei Zeichen, links die Ursprungsmodelle mit drei Zuständen, rechts die längenadaptierten Modelle mit variabler Anzahl an Zuständen. Durch die Längenadaption werden deutlich mehr Details in den Modellbildern sichtbar, was zu der oben gezeigten starken Verbesserung der Erkennungsleistung führt. [3] Durch die feinere Model-

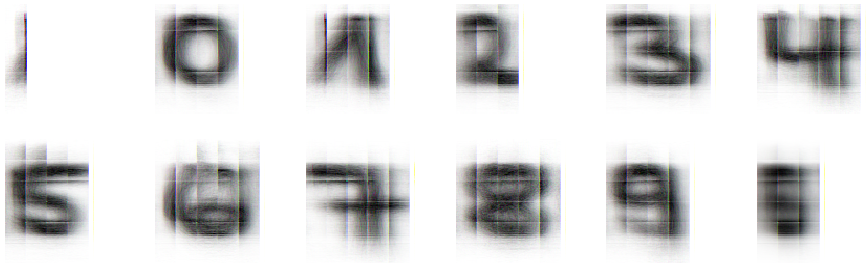

Abbildung 11.5: Modellbilder deutscher Ziffern nach Längenadaption der Modelle. Das Pause- (links oben) und das Joker-Modell (rechts unten) wurden nicht adaptiert. (Vgl. Abbildung 5.5 auf Seite 54.)

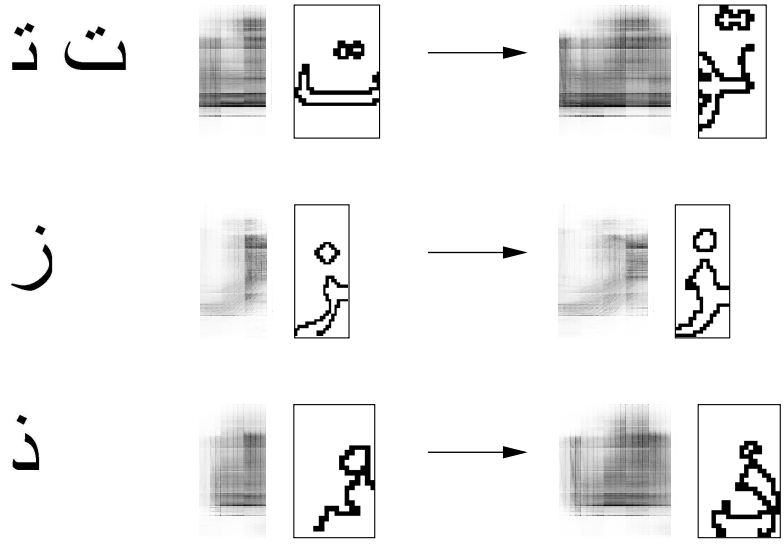

Abbildung 11.6: Längenadaption für eine Auswahl von arabischen Zeichen. Die Visualisierung der Modelle ist jeweils deutlich detaillierter, was teilweise zu einer Änderung der typischen Form der modellierten Zeichen führt.

lierung ändern die Modelle zum Teil ihr Aussehen, wie die prototypischen Beispielzeichen rechts von den Modellbildern zeigen. Sie sind die jeweils am höchsten bewerteten Einzelzeichen der Trainingsstichprobe und repräsentieren dadurch den „typischen“ Vertreter des Modells. In der ersten und dritten Zeile sind die Änderungen der Zeichen sehr deutlich.

Die resultierende Modellstruktur für Modelle mit mehreren Zustandspfaden sieht man schön in Abbildung 11.7 für die Modelle von US-Adressdaten. Am Beispiel der Buchstaben „B“ bis „E“ kann man sehen, wie breite Großbuchstaben detailliert modelliert werden. Es kann aber auch ein komplexer Kleinbuchstabe von der größten Anzahl an Zuständen repräsentiert sein, wie der Buchstabe „F“ zeigt. Insgesamt kann das Ergebnis der Modelllängenschätzung als plausibel bezeichnet werden.

[3] Die Streifen innerhalb der Modellbilder sind auf Quantisierungseffekte zurückzuführen, die aufgrund der niedrigen Auflösung der gescannten Wortbilder entstehen.

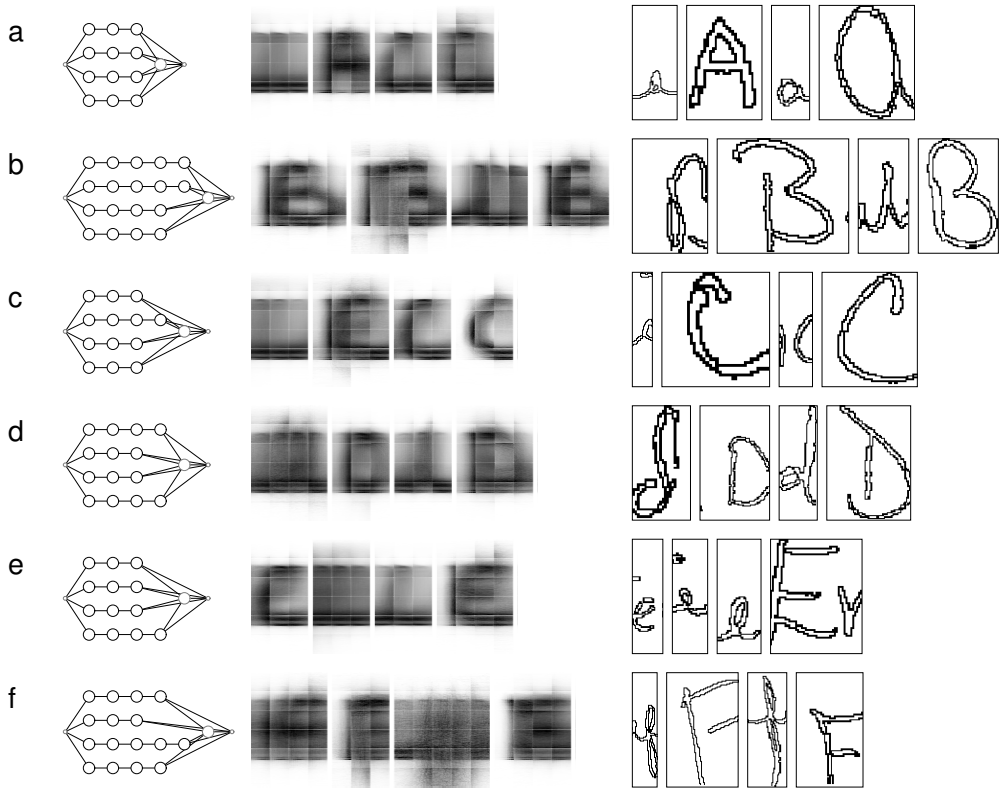

Abbildung 11.7: Modellstruktur und -bilder nach einer Längenadaption der Buchstabenmodelle für US-Adressdaten.

11.3.5 Fazit

Auch wenn der Algorithmus zur Optimierung der Pfadlänge Einschränkungen hat, indem er nicht konvergiert und stark von der Initialisierung abhängig ist, stellt er in der Praxis ein äußerst nützliches Werkzeug zur Verbesserung der Modellierung dar. Die resultierenden Modelle sind plausibel; sie zeichnen sich aber zusätzlich dadurch aus, dass sie sich zum Teil *nicht* mit der intuitiven Vorstellung decken. Hier zeigt sich ein Vorteil der automatischen gegenüber der manuellen Modellierung: Sie wird nicht von Vorstellungen über das Aussehen der Schrift geleitet, sondern beschreibt die real auftretenden Daten der Trainingsstichprobe.

Die Erkennungsleistung der untersuchten Projekte konnte durch die automatische Adaption erhöht werden. Es wird sich zeigen, wie sich diese Ergebnisse mit der in den folgenden Abschnitten behandelten Adaption der Allographen kombinieren lassen.

Kapitel 12

Bewertung von HMMs

Für die Bestimmung der Allographmodelle müssen Maße entwickelt werden, die es erlauben, die einzelnen Buchstaben-HMMs zu bewerten, unabhängig von den Auswirkungen auf das gesamte Schrifterkennungssystem. Es gibt dafür verschiedene Anwendungsmöglichkeiten:

- Die Bewertung kann für die *Analyse* der Ergebnisse der Strukturbestimmung oder des Parametertrainings dienen. Mit den Eigenschaften der Modelle kann die Qualität der Algorithmen bewertet werden. Es ist auch möglich, Informationen über die zugrundeliegenden Daten zu sammeln.
- Eine Bewertung kann den *Algorithmus* zur Strukturbestimmung steuern. Dazu muss ein Maß nicht unbedingt selbst in den Kriterien für die Modellwahl enthalten sein, es kann stattdessen als geeignete Heuristik für die Maximierung eines unabhängigen Kriterium eingesetzt werden. Mit ihm kann zum Beispiel entschieden werden, welche Schreibvarianten oder Modellzustände zusammengelegt oder geclustert werden sollen.

Es werden zwei Maße vorgestellt: Eins zum Vergleich zweier verschiedener HMMs, und eins zur direkten Bewertung einzelner Modelle.

Des weiteren werden konkrete Umsetzungen der Bayes'schen Modellwahl aus Kapitel 9 für die spezielle Topologien der HMMs aus Abschnitt 8.1 hergeleitet. Es wird untersucht, wie das Bayes'sche Informations-Kriterium, das Akaike Informations-Kriterium und die beiden Näherungen nach Cheeseman-Stutz angepasst werden können.

12.1 Distanzmaße bei HMMs

Ein Beispiel für den Einsatz von Distanzmaßen bei HMMs ist das diskriminative Training der Modellparameter [Kwo98]. Im Gegensatz zum Maximum-Likelihood-Training ist das Kriterium für die Parameterwahl der maximale Modellabstand, MMD (*Maximum model distance*). Parameter, die für die Unterscheidung von ähnlichen Beobachtungen relevanter sind, werden bei dieser Methode intensiver nachtrainiert, wodurch die Fehlerrate bei der Erkennung gesenkt werden kann.

12.1.1 Abstand zwischen Zuständen

Bevor über den Abstand zwischen ganzen Hidden-Markov-Modellen gesprochen werden kann, muss zunächst der Abstand zwischen einzelnen Zuständen definiert werden. Dabei werden sich die gegenseitigen Beziehungen zwischen Beobachtungen und Modell für die Abstandsberechnung zeigen.

Als Vereinfachung beschränken sich die Überlegungen auf diskrete und semikontinuierliche HMMs mit festem Codebuch. Spezielle Eigenschaften postulierter Modellverteilungen – etwa für die Berechnung der Divergenz von Mischdichten – müssen dadurch nicht beachtet werden. Im Fall von semikontinuierlichen HMMs erfolgt eine Fuzzy-Vektorquantisierung. Die Beobachtungen werden durch Vektoren der Klassenwahrscheinlichkeiten repräsentiert, deren Elemente sich aufgrund ihrer Definition zu eins addieren.

Die Emissionsparameter der HMM-Zustände geben die Wahrscheinlichkeit für die Beobachtung der Codebuchklassen an. Im Training werden sie als Erwartungswerte der Klassenwahrscheinlichkeiten gebildet. Bei der Erkennung dienen sie als Gewichte für die Klassenwahrscheinlichkeiten; die Wahrscheinlichkeit der Beobachtung berechnet sich über das Skalarprodukt aus Emissionsgewichten und Klassenwahrscheinlichkeiten:

$$P(O_t|s_i) = \sum_k P(y_k|s_i) \cdot P(O_t|y_k, s_i) = \sum_k b_i(y_k) \cdot \varphi_k(O_t). \tag{12.1}$$

Da diese Wahrscheinlichkeit ein Maß für die Ähnlichkeit der Beobachtung zum Erwartungswert darstellt, kann durch $(1 - P(O_t|s))$ ein Abstand zwischen Zustand s und Beobachtung O_t definiert werden.

Auf dieselbe Weise kann man auch zwei Erwartungswerte miteinander vergleichen und damit einen Abstand zwischen Zuständen definieren. Der Abstand zweier Zustände ist dann über das Skalarprodukt der Emissionsgewichte definiert:

$$D(s_i, s_j) = 1 - \sum_k b_i(y_k) \cdot b_j(y_k). \tag{12.2}$$

Diese Abstandsdefinition ist nicht mit dem euklidischen Abstand identisch, bewahrt aber die Relationen des euklidischen Abstands von einem festen Zustand zu den anderen Zuständen. Ihr Vorteil liegt in der Interpretation der Emissionswahrscheinlichkeit: Ein Zustand ist dem anderen ähnlich, wenn die Wahrscheinlichkeit hoch ist, dass er die erwartete Klassenverteilung des anderen emittiert. Diese Dualität von Beobachtung und Zustand erweist sich für die Definition und Berechnung der Abstände vollständiger HMMs als Vorteil.

12.1.2 Distanzmaße für beliebige HMMs

Es werden in der Literatur verschiedene Maße für den Abstand beliebiger HMMs vorgestellt, die im folgenden kurz skizziert werden.

Summe der Zustandsabstände

Ein möglicher Ansatz für die Definition des Abstands zweier HMMs besteht darin, einfach den Abstand der einzelnen Zustände aufzusummieren [Jua85]:

$$D_{\text{state}}(\lambda_1,\lambda_2) = \sqrt{\frac{1}{N}\sum_i D\left(s_i^{(1)}, s_i^{(2)}\right)^2}. \tag{12.3}$$

Dafür kann sowohl der euklidische Abstand, als auch das Skalarprodukt der Emissionswahrscheinlichkeiten (12.2) verwendet werden. Die Definition setzt identische Topologien der zu vergleichenden HMMs voraus. Übergangswahrscheinlichkeiten, und damit die zeitliche Struktur der HMMs, werden nicht berücksichtigt. Zwei HMMs mit beliebig kleinem Abstand können dieselbe Beobachtung mit sehr unterschiedlichen Wahrscheinlichkeiten erzeugen.

Dies weist auf ein Kriterium hin, das für eine zielgerichtete HMM-Abstandsdefinition gelten sollte. Gesucht ist nicht der direkte Vergleich der internen Zustände, sondern ein Vergleich der Ausgaben der beiden HMMs: Ähnlich sind solche HMMs, die ähnliche Beobachtungen generieren.

Kullback-Leibler-Divergenz

Ein probabilistischer Ansatz für die Definition eines Abstands basiert auf der Kullback-Leibler-Divergenz [Jua85], (vgl. Gleichung (5.11) in Abschnitt 5.4.4):

$$D_{\text{KL}}(\lambda_1,\lambda_2) = \int_{\{O\}} \frac{1}{T}\log\frac{P_{\lambda_2}(O)}{P_{\lambda_1}(O)} P_\lambda(O)\, dO. \tag{12.4}$$

Es ist ein Maß für die Ähnlichkeit der Wahrscheinlichkeitsverteilungen, mit der die beiden Modelle λ_1 und λ_2 die Menge aller Beobachtungen erzeugen. Da die Menge der betrachteten Sequenzen vom Modell λ_1 vorgegeben ist, ist das Maß asymmetrisch, kann aber wie in Gleichung (5.23) einfach symmetrisiert werden.

Eine Möglichkeit der Berechnung besteht in dem Einsatz von Monte-Carlo-Methoden. Es werden mit λ_1 ausreichend lange Zustandssequenzen O generiert und so das Integral durch eine Summe genähert. Diese Methode ist sehr rechenaufwändig. Eine Berechnungsmethode, die ohne Zufallsfolgen auskommt, wird in [Fal95] vorgestellt. Sie macht die Voraussetzung, dass die betrachteten Modelle so ähnlich sind, dass man von gleichen Viterbi-Pfaden für die Generierung der Beobachtung ausgehen kann. Ohne explizite Modellierung der Zustandsdauer kann diese Annahme aber nicht gemacht werden, da die Länge der realen Beobachtungen stark von den generierten Sequenzen abweicht.

Co-Emissionen

Eine andere Abstandsdefinition basiert auf Co-Emissionswahrscheinlichkeiten [Lyn99]. Die Ähnlichkeit der Modelle ist durch die Wahrscheinlichkeit gegeben, dass die beiden Modelle

λ_1 und λ_2 voneinander unabhängig die identischen Sequenzen erzeugen:

$$P_{\text{co}}(\lambda_1,\lambda_2) = \sum_{\{O\}} P_{\lambda_1}(O) \cdot P_{\lambda_2}(O). \tag{12.5}$$

Auch hier gibt es eine Berechnungsmethode, die ohne Monte-Carlo-Simulation auskommt. Ein Problem besteht aber darin, dass die Co-Emissionswahrscheinlichkeit zu λ_1 nicht unbedingt für $\lambda_1 = \lambda_2$ am höchsten ist. Es ist nämlich entscheidend, wie konzentriert die Wahrscheinlichkeiten im Raum der Beobachtungsfolgen sind.

Keines der hier vorgestellten Maße ist für die vorliegende Anwendung perfekt. In Abschnitt 8.2 werden die Allographen allerdings auf lineare Modelle eingeschränkt; unter dieser Bedingung für die Struktur der HMMs können weitere Definitionen des Abstands gegeben werden.

12.1.3 Vergleich linearer Modelle

Im Fall linearer HMMs kann ein Verfahren zur Abstandsbestimmung verwendet werden, das der Berechnung der Wahrscheinlichkeit der Beobachtung über den Forward- oder Viterbi-Algorithmus sehr ähnlich ist. Die Berechnung ist damit sehr effektiv durchzuführen. Es nutzt die weiter oben gezeigte Beziehung von Modellzustand und Beobachtung, und behandelt die HMM-Zustände wie Beobachtungen. Das Prinzip der Dynamischen Programmierung, das sowohl für den Vergleich von Sequenzen mit *Dynamic Time Warping* (DTW), als auch ihre Klassifikation mit HMMs verwendet wird, kann so auch für die Berechnung eines HMM-Abstands genutzt werden.

Dynamic Time Warping dient zum Vergleich zweier Beobachtungsfolgen. Es kann zur Klassifikation von Zeitreihen verwendet werden, wenn die Klassen durch ein oder mehrere Referenzmodelle repräsentiert sind. Die Zuordnung einer Klasse erfolgt über den minimalen, mit DTW bestimmten Abstand zu den Prototypen [Sak78, Hua01]. Seien X_t und Y_t die beiden zu vergleichenden Beobachtungsfolgen. Dann gibt es verschiedene Pfade Φ der Länge N, die eine zeitliche Angleichung der Beobachtungen darstellen, deren Elemente also Zuordnungen der Einzelbeobachtungen $\Phi_i \equiv (X_{\Phi_{Xi}}, Y_{\Phi_{Yi}})$ sind. Bei jedem Schritt $i \rightarrow (i+1)$ geht dabei mindestens eine der Folgen X und Y zur nächsten Beobachtung über. Über solch ein *alignment* lässt sich durch die Summe der Abstände der Einzelbeobachtungen ein Abstand der vollständigen Beobachtungsfolgen definieren:

$$D_\Phi(X,Y) = \frac{1}{N} \sum_{i=1}^{N} D(X_{\Phi_{Xi}}, Y_{\Phi_{Yi}}). \tag{12.6}$$

Die Definition wird unabhängig von der Wahl eines konkreten Pfades, wenn man als Abstand der Beobachtungen den kleinsten Abstand aller Pfade wählt

$$D(X,Y) = \min_{\Phi} D_\Phi(X,Y). \tag{12.7}$$

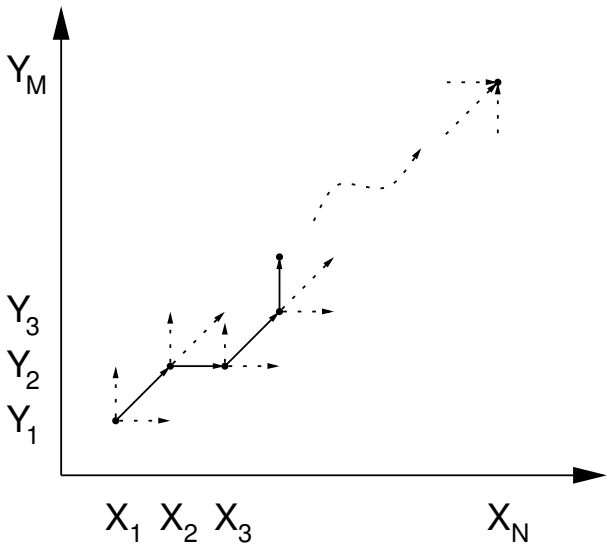

Abbildung 12.1: Statistical Dynamic Time Warping (SDTW). Anhand einer einzigen Skizze kann man sich das gemeinsame Prinzip von Dynamic Time Warping, linearen HMMs und der Definition eines Abstands zweier HMMs verdeutlichen. X und Y stehen entweder für zwei Zeitreihen, eine Zeitreihe und ein statistisches Modell, oder zwei Modelle.

Den Pfad, der den Abstand minimiert, bezeichnet man als „kürzesten“ oder Viterbi-Pfad. Er ist in Abbildung 12.1 skizziert. Seine Berechnung erfolgt mit dynamischer Programmierung, sie ähnelt dem in Abschnitt 3.3.2 beschriebenen Viterbi-Algorithmus für HMMs. Der Unterschied im Ablauf besteht darin, dass bei einem Zeitschritt des Pfades Φ_i jede der beiden Beobachtungsfolgen in der aktuellen Beobachtung verharren kann. Die Berechnung des Trellis muss deshalb diagonal zu beiden Beobachtungsreihen erfolgen.

Die Analogie zur Erkennung mit Hidden-Markov-Modellen besteht darin, dass eine der bisher betrachteten Beobachtungsfolgen durch eine statistische Beschreibung, eben ein lineares HMM, ersetzt wird. Die HMM-Erkennung wird deshalb auch als *Statistical DTW* (SDTW) bezeichnet [Bah01]. Es wird gewissermaßen der Abstand zwischen einem HMM und der Beobachtungsfolge gemessen. In Abschnitt 12.1.1 wurde verdeutlicht, dass – durch die Beschreibung der Emissionen als Erwartungswerte – auch der Abstand zwischen Zustand und Beobachtung auf die gleiche Weise wie der zweier Beobachtungen berechnet werden kann. Die Einführung von Übergangswahrscheinlichkeiten ändert am Prinzip der Berechnung nichts Grundlegendes. Ein Unterschied besteht nur darin, dass nicht mehrere Zustände dieselbe Beobachtung emittieren können, der Pfad also in jedem Schritt zur nächsten Beobachtung übergeht. Deshalb erfolgt die Berechnung des Viterbi nicht diagonal, sondern in Richtung der fortschreitenden Beobachtungsfolge.

Man geht nun noch einen Schritt weiter und berechnet auch den Abstand zweier HMMs mit derselben Methode der beiden anderen Fälle [Bah01]. Auch die zweite Zeitreihe wird durch ein HMM als statistische Beschreibung ersetzt, und der Abstand der beiden Beobachtungen in (12.6) wird zum Abstand der Zustände. Die Beachtung der Übergangswahrschein-

lichkeiten stellt auch jetzt keine Schwierigkeit dar. Sie lassen sich leicht in die Berechnung des Abstands der Modelle $\lambda^{(X)}$ und $\lambda^{(Y)}$ integrieren:

$$\begin{aligned} & D_\Phi\left(\lambda^{(X)}, \lambda^{(Y)}\right) \\ & = \frac{1}{N}\sum_{i=1}^{N} D\left(s^{(X)}_{\Phi_{Xi}}, s^{(Y)}_{\Phi_{Yi}}\right) + \left(1 - a^{(X)}_{\Phi_{Xi}\Phi_{X(i-1)}}\right) + \left(1 - a^{(Y)}_{\Phi_{Yi}\Phi_{Y(i-1)}}\right). \end{aligned} \tag{12.8}$$

Die Analogie zum DTW ist wieder vollständig, da die Zustände beider Modelle Selbstübergänge besitzen und so beide Modelle bei einem Schritt des Pfades im aktuellen Zustand bleiben können. Die Viterbi-Berechnung erfolgt damit diagonal zu den beiden Zustandsfolgen.

Die Wahl des Abstandsmaßes der Zustände ist bei dieser Definition der Modellähnlichkeit noch frei. Bei kontinuierlichen Modellen werden unter anderem die Kullback-Leibler-Divergenz, die Bayes-Fehlerwahrscheinlichkeit [Bah01] oder der Bhattacharyya-Abstand [Mak96] vorgeschlagen. In dieser Arbeit wird, wegen der Einschränkung auf diskrete und semikontinuierliche Modelle, der in Abschnitt 12.1.1 definierte Abstand, das Skalarprodukt der Erwartungswerte, verwendet.

Das hier vorgestellte Abstandsmaß für HMMs kann bei gleicher Anzahl von Zuständen in den eingangs vorgestellten Abstand (12.3) übergehen. Trotzdem besitzt es nicht die genannten Nachteile: Man kann Modelle verschiedener Länge vergleichen, die Übergangswahrscheinlichkeiten werden beachtet, und auch bei identischer Länge der Modelle kann aufgrund der Art der Berechnung ein unterschiedliches *alignment* der Zustände gewählt werden.

12.1.4 Eigenschaften

Über den Modellabstand (12.8) können Ähnlichkeits- oder Verwandtschaftsverhältnisse von Ziffern und Buchstaben definiert werden. Eine gute Methode, die Qualität des Abstandsmaßes zu untersuchen, ist der visuelle Vergleich. Über den Zusammenhang zwischen Klassifikationsergebnissen und Modellabstand kann das Abstandsmaß auch quantitativ analysiert werden.

Visueller Vergleich

Die Modelle werden mit der in Abschnitt 5.5 beschriebenen Methode visualisiert. Abbildung 12.2 zeigt einige Beispiele der Ziffern mit kleinstem bzw. größtem Abstand in einem Erkennungssystem für US-amerikanische ZIP-Codes. Die Ergebnisse stimmen gut mit der menschlichen Vorstellung von Ähnlichkeit überein. Die Ziffern „3“, „5“ und „8“ sind einander am ähnlichsten, während die Paare „2“ und „4“, sowie „1“ und „2“ die größten Unterschiede in der Gestalt aufweisen.

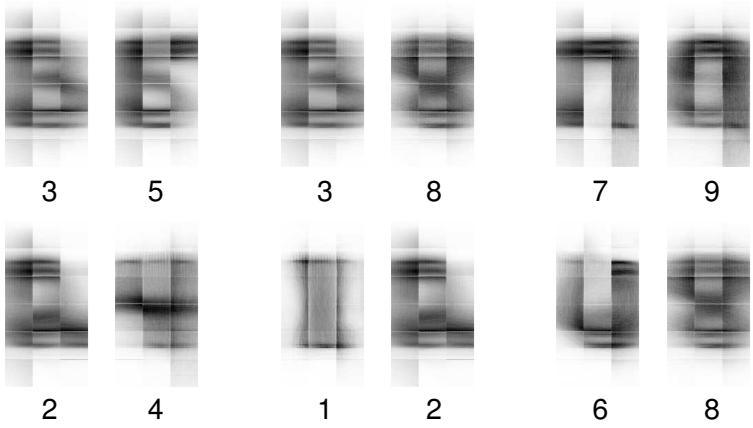

Abbildung 12.2: Beispiele für Paare von Modellen mit geringem (obere Reihe) und hohem Abstand (untere Reihe) in einem Erkennungssystem für US-Ziffern.

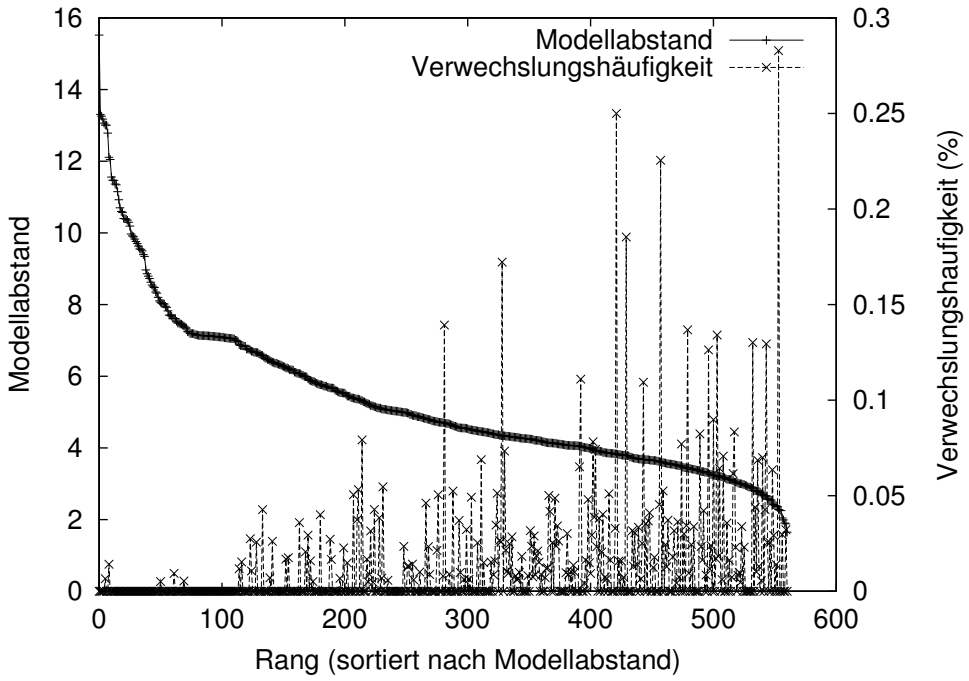

Abbildung 12.3: Modellabstand und Fehlklassifikation. Sämtliche Zeichenpaare eines deutschen Worterkennungssystems (34 Zeichen) sind nach Modellabstand sortiert. Daneben ist aufgetragen, wie häufig diese zwei Zeichen bei der Erkennung verwechselt wurden.

Modellabstand und Fehlklassifikation

Um die Nützlichkeit des Distanzmaßes zu beweisen, wird eine Korrelation zwischen Modellabstand und Fehlklassifikation erwartet [Bah01]. Ein Beispiel mit deutschen Adressdaten, das 34 Buchstabenmodelle umfasst, wird in den Abbildungen 12.3 und 12.4 gezeigt. In Abbildung 12.3 sieht man, wie die Wahrscheinlichkeit einer Fehlklassifikation, also der Verwechslung von Zeichen innerhalb eines Worts, mit abnehmendem Abstand zwischen den HMMs wächst. Abbildung 12.4 zeigt deutlich, dass die höchsten Verwechslungsraten bei

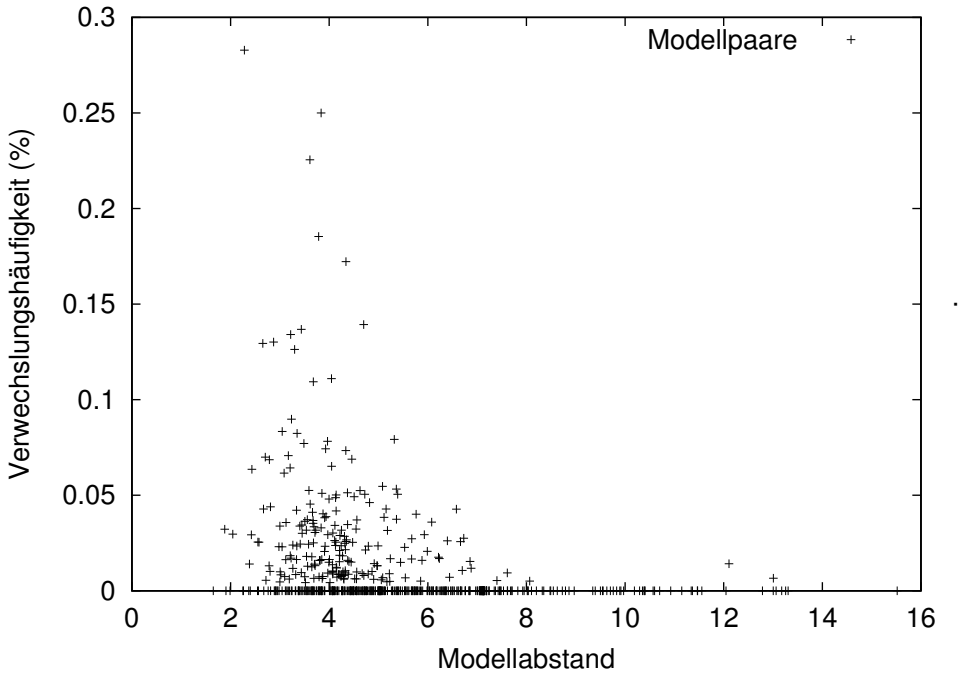

Abbildung 12.4: Modellabstand und Fehlklassifikation. Hohe Verwechslungsraten treten hauptsächlich bei Modellen mit geringem Abstand auf.

den Modellpaaren auftreten, die den kleinsten Abstand haben. Diese Korrelationen werden in allen acht untersuchten Erkennungssystemen (vgl. Abschnitt 6.4.1) beobachtet.

12.2 Entropie linearer HMMs

Für die Auswahl einzelner Zeichenmodelle zur Modifikation der Modellstruktur wird ein Maß für die Modellgüte benötigt, das unabhängig von den anderen HMMs die Qualität der Modellierung angibt. Ein einfaches Maß für semikontinuierliche Systeme ist die Entropie der Emissionsgewichte $b_i(k)$ der Klassen k im Zustand s_i,

$$H(s_i) = -\sum_k b_i(k) \log b_i(k). \tag{12.9}$$

Die Eigenschaften der Klassen selbst werden nicht betrachtet. Im Fall gut separierbarer Klassen ist der Wert aber ein Indikator, wie gut die Modellzustände definiert sind. Die Qualität des gesamten Modells erhält man über die Bildung des Mittelwerts der Entropie aller Modellzustände.

Der Einfluss auf die Erkennungsgüte kann in Abbildung 12.5 am Beispiel von US-Ziffern gesehen werden. Auch wenn Wörterbucheffekte die Erkennungsrate einzelner Zeichen beeinflussen, kann die erwartete Korrelation erkannt werden. Mit abnehmender Entropie steigt die Erkennungsrate der einzelnen Ziffern.

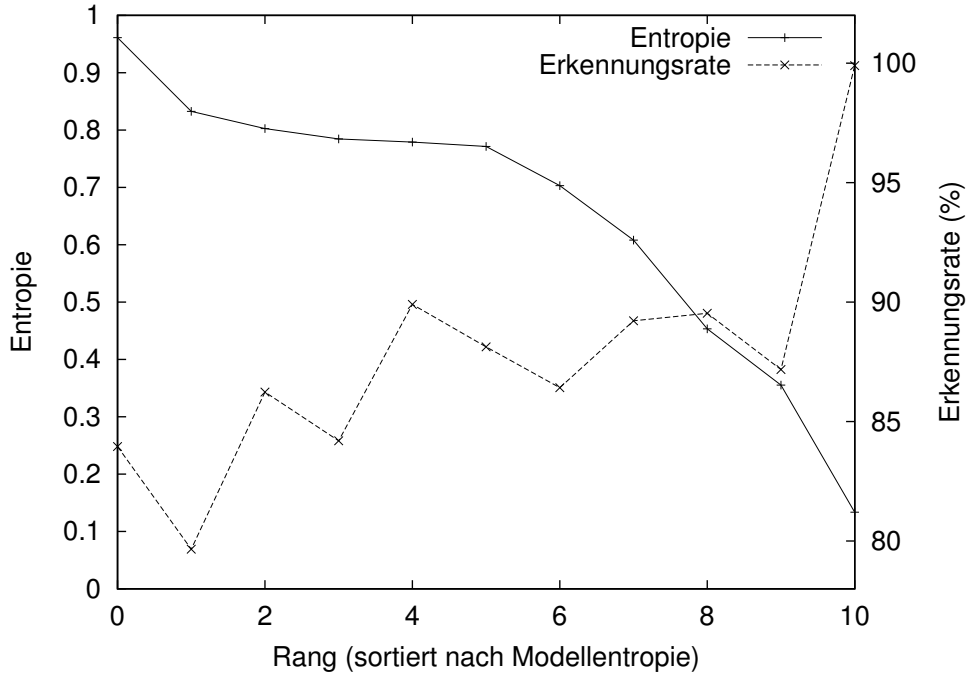

Abbildung 12.5: Modellentropie und Zeichenerkennungsrate. Die Zeichen eines US-Ziffernerkennungssystems (CEDAR) sind nach absteigender Entropie sortiert. Mit abnehmender Entropie steigt die Erkennungsrate der Ziffern.

12.3 Modellwahl für Buchstaben-HMMs

In Kapitel 9 sind mit verschiedenen Ansätzen und Näherungen mehrere Kriterien zur Modellwahl hergeleitet worden: das Akaike Informationskriterium, das Bayes'sche Informationskriterium und die Cheeseman-Stutz-Näherungen. In diesem Abschnitt wird untersucht, wie sich diese Kriterien auf die konkret behandelten Daten und die Struktur der Buchstabenmodelle anwenden lassen.

12.3.1 Anwendung des AIC

Da für das gesamte Schriftmodell eine Vielzahl von Parametern bestimmt werden muss, was in mehreren, in Abschnitt 5.4 ausführlich beschriebenen Trainingsschritten geschieht, ist es keine triviale Aufgabe, die Modellwahlkriterien an die konkrete Situation anzupassen. Deshalb wird mit dem AIC zunächst das formal einfachste Kriterium untersucht. Es lautet nach Gleichung (9.11):

$$P(M_s|X) \approx \log P(X|M_s, \hat{\theta}) - d. \tag{12.10}$$

Zur Berechnung benötigt man nur die Anzahl der freien Modellparameter d. Sie setzt sich aus der Anzahl der Parameter des Klassifikators d_{Class} und der Zustandsparameter d_{HMM} zusammen.

Jede Klasse ist durch die Matrix der LDA-Transformation und die Beschreibung der

Gaußverteilung bestimmt. Die Dimension der Vektoren ist für alle zu vergleichenden Modelle konstant und beträgt vor der Transformation 100, danach 32. Variabel ist bei den Modellen aber die Anzahl der Klassen $k_{\max}$. Die Anzahl der Klassifikator-Parameter beträgt damit

$$d_{\text{Class}} = k_{\max} \cdot \left(100 \cdot 32 + \frac{32 \cdot (32+1)}{2} \right) = 3728 \cdot k_{\max}. \tag{12.11}$$

Für die Abschätzung der Anzahl der HMM-Parameter werden die drei Bestandteile der Modellbeschreibung $\lambda = (\pi, A, B)$ jedes Buchstabenmodells addiert. Für etwa jeden dritten Zustand wird eine Eingangswahrscheinlichkeit π_i bestimmt, und jeder Zustand besitzt etwa zwei echte Übergangswahrscheinlichkeiten a_{ii} und $a_{i(i+1)}$ und so viele Emissionsgewichte $b_j(k)$ wie Klassen. Die Anzahl der Zustände beträgt N, dann ist

$$d_{\text{HMM}} = N \cdot \left(\frac{1}{3} + 2 + k_{\max} \right) \approx N \cdot k_{\max}. \tag{12.12}$$

Die gesamte Anzahl freier Parameter d beträgt also etwa

$$d = (3728 + N) \cdot k_{\max}. \tag{12.13}$$

Das AIC berechnet sich damit durch

$$G_{\text{AIC}}(M_s) = \log P(X|M_s, \hat{\theta}) - (3728 + N) \cdot k_{\max}. \tag{12.14}$$

Dies ist eine grobe Abschätzung der Größenordnung. Es wird zum Beispiel nicht berücksichtigt, dass die Bedeutung der einzelnen Modellparameter für die Qualität des Modells sehr unterschiedlich ist, wenn zum Beispiel Übergangs- und Emissionswahrscheinlichkeiten miteinander verglichen werden. Dennoch wird es im folgenden Kapitel möglich sein, mit diesem Kriterium verschiedene Modellstrukturen miteinander zu vergleichen.

12.3.2 Anwendung der anderen Kriterien

Schon die Anpassung des einfachsten Kriteriums an das konkrete Schriftmodell ist mit einigen Unsicherheiten behaftet. Im folgenden werden Überlegungen angestellt, wie die anderen Näherungen sich für die konkrete Situation anwenden lassen.

Bayes'sches Informations-Kriterium

Das BIC in Gleichung (9.10) unterscheidet sich vom AIC nur durch den Faktor $\frac{1}{2} \log T$ im zweiten Term, durch den Modellkomplexität bestraft wird. T bezeichnet die Anzahl der Trainingsdaten. Die Größenordnung des Faktors lässt sich wie folgt abschätzen:

Bei durchschnittlich 50 Vektoren in 10000 bis 20000 Trainingsbildern beträgt die Anzahl der Trainingsvektoren zwischen 500000 und einer Million. Da jeder Vektor nur Informationen für einen HMM-Zustand trägt, reduziert sich diese Zahl bei im Mittel etwa 100 Zuständen noch um diesen Faktor. AIC und BIC unterscheiden sich im „Bestrafungsterm“ also um einen Faktor der Größenordnung $\frac{1}{2} \cdot \log 7500 \approx 4,5$.

Angesichts der Unsicherheit bei der Bestimmung der freien Parameter bei der Definition des AIC wird dieser Faktor vernachlässigt. Es erfolgt neben (12.14) keine spezielle Modellbewertung über das BIC.

Cheeseman-Stutz-Näherungen

Ein ähnliches Argument gilt auch für den Verzicht einer speziellen Berücksichtigung der Cheeseman-Stutz-Näherungen.

Der Bestrafungsterm der CS-Maximumsnäherung in Gleichung (9.22) beträgt $-\sum_i \log \hat{\theta}_i^{\alpha}$. Der Ausdruck summiert über alle freien Parameter und ist dadurch proportional zur Parameteranzahl d und zum durchschnittlichen Wert von $-\log \hat{\theta}_i^{\alpha}$. Der zweite Wert wird über alle zu vergleichenden Modellstrukturen als konstant angenommen; seine Größenordnung befindet sich für die behandelten Parameter in einem Bereich, dass auch die CS-Maximumsnäherung als separates Kriterium vernachlässigt werden kann.

Auch bei der Herleitung der geschlossenen Lösung der CS-Näherung in Kapitel 9 sieht man in Gleichung (9.24) die Proportionalität des Bestrafungsterms zur Anzahl der freien Parameter. Aus der geschlossenen Lösbarkeit des Integrals kann wegen der hohen Komplexität des betrachteten Schriftmodells leider kein Gewinn erzielt werden.

12.3.3 Fazit

Die vier verschiedenen Näherungen der Modellbewertung, die in Kapitel 9 theoretisch hergeleitet worden sind, erweisen sich bei der Anwendung auf das Schriftmodell als sehr ähnlich. Die Likelihood der Trainingsdaten wird gegen die Komplexität des Modells abgewogen, welche durch die Anzahl der freien Parameter gegeben ist. Mit Gleichung (12.14) wurde ein einziges, einfaches Modellierungskriterium für die konkrete Schriftmodellierung hergeleitet. Im nächsten Kapitel wird sich zeigen, wie gut sich damit die reale Modellierungsgüte der Erkennungssysteme vorhersagen lässt.

Kapitel 13

Clusterung der Allographen

Der folgende Schritt bei der Modellierung, die Festlegung der optimalen Anzahl von Pfaden im Graphemmodell, ist besonders für die Analyse der behandelten Schrift interessant. Es werden die Grapheme in Allographen geclustert, und damit sowohl die Anzahl der unterschiedlichen Schreibweisen bestimmt, als auch ihre Gestalt offenbart. Durch die Clusterung erhält man Einblick in das Wesen der Trainingsdaten.

Die Möglichkeit zur Bestimmung der Pfadlänge, die in Kapitel 11 vorgestellt wurde, wird für diese Optimierung als Grundlage herangezogen. Sie wird als eine erweiterte Form der Parameterbestimmung angesehen, mit der verschiedene generierte Modelle, die die Schreibvarianten unterschiedlich granular beschreiben, trainiert und damit auch bewertet werden können. Das Problem der Pfadlängenbestimmung wird als gelöst angesehen, und die Allograph-Clusterung unabhängig von dieser Aufgabe angegangen. In der Herangehensweise an das Problem bestehen Ähnlichkeiten. Auch bei der Clusterung wird iterativ vorgegangen. Neue Modellpfade werden generiert, bewertet und ausgewählt, wobei immer gewährleistet sein muss, dass die Segmentgrenzen richtig bestimmt werden können.

13.1 Algorithmus

Ein häufiges Vorgehen für die Bestimmung einer optimalen Anzahl von Clustern besteht darin, dass nacheinander verschiedene Größen vorgegeben werden, mit denen eine Clusterung durchgeführt wird, bis die optimale Clusterung gefunden ist. Man beginnt zum Beispiel mit einem einzelnen Cluster und erhöht die Anzahl der Cluster so lange, bis keine Verbesserung des gewählten Kriteriums mehr erfolgt [Li00a].

Bei der Schriftmodellierung stellt sich das zusätzliche Problem, dass viele Grapheme gleichzeitig geclustert werden müssen. In jeder Iteration muss eine Entscheidung getroffen werden, bei welchem Graphem eine Änderung der Modellierung vorgenommen werden soll. Mit diesem zusätzlichen Auswahlschritt stellt sich der generelle Aufbau des Algorithmus wie in Abbildung 13.1 dar: Nach der Initialisierung eines Startmodells werden in mehreren Iterationen Grapheme ausgewählt und modifiziert. Die Bewertung der Modellstrukturen

Initialisierung

Start mit plausiblem Modell.

Training der Längen und Parameter.

Iteration

Auswahl der Grapheme zur Modifikation.

Modifikation der Allographen.

Neutraining der HMM-Parameter.

Abschluss

Bewertung.

Abbruch bei Verschlechterung des gewählten Kriteriums.

Abbildung 13.1: Algorithmus zur Clusterung der Allographen

jeder Iteration erfolgt nach einem Neutraining der HMM-Parameter.

Das Startsystem muss in der Lage sein, die Trainingswörter sinnvoll in Grapheme zu segmentieren, da diese die zu clusternde Einheit sind. Startet man mit einem primitiven Modell, sollte zumindest ein Minimum sich stark unterscheidender Schreibweisen vorgesehen sein. Es wird dann ein vollständiges Training der HMM-Parameter, eventuell zusammen mit der Bestimmung der Pfadlängen, durchgeführt. Am Ausgangspunkt steht immer ein funktionsfähiges Erkennungssystem.

Eine schnellere und eventuell bessere Konvergenz erreicht man, wenn statt eines einfachen schon ein plausibles Modell zur Verfügung steht, und dieses sowohl durch Hinzunahme, als auch durch Entfernung von Clustern variiert wird. Man hat dann zum Beispiel die Möglichkeit, mit einem sehr komplexen Modell zu starten, und dadurch die Qualität der Optimierung zu verbessern. Es lassen sich auch schon bestehende Systeme einfach verbessern, ohne dass von vornherein klar ist, ob sie zu einfach oder zu komplex sind. Für die Auswahl und die Modifikation der Graphemmodelle müssen dann auch Verfahren zur Verfügung stehen, die zu entfernende Pfade behandeln.

13.1.1 Strategien zur Modellwahl

Die Auswahl der Grapheme, die verändert werden sollen, und die Art der Modifikation, die an ihnen durchgeführt wird, bestimmen die Strategie der Clusterung. Sie beeinflusst maßgeblich die Leistung des Algorithmus. Sie hat die Aufgabe, die Modelle einzeln zu bewerten und damit jene Modelle zu finden, bei denen eine Modifikation die Modellierung insgesamt am meisten verbessert.

Die Grapheme können nach globalen oder lokalen Kriterien ausgewählt werden. Ein *glo-*

bales Kriterium misst die Auswirkung des einzelnen Buchstabenmodells auf die Güte der gesamten Modellierung. Es ist durch den Anteil der einzelnen Modelle an der Bewertung der Trainingsdaten gegeben. Es handelt sich aber um kein Bayes'sches Kriterium, da die Modellkomplexität bei der Auswahl der möglichen Modifikationen keine Rolle spielt. *Lokale* Kriterien berücksichtigen nur die Eigenschaften der Modelle selbst. Sie sind dadurch effizient zu berechnen, unterscheiden sich aber von dem Kriterium, das letztendlich maximiert werden soll.

Eine ähnliche Unterscheidung der Strategien erfolgt danach, ob die Modellierung direkt oder indirekt optimiert wird. Eine *direkte* Optimierung wählt die Modifikation nach ihrer Wirkung aus, wählt im konkreten Fall also die Änderung der Modellierung des Graphems, bei der die Likelihood am stärksten wächst. Eine solche Strategie ist die Implementierung eines *greedy* Algorithmus. Die Auswahl über lokale Kriterien stellt immer eine *indirekte* Optimierung dar und kann als Heuristik bezeichnet werden.

Es werden drei Auswahlmethoden vorgestellt, eine globale und zwei lokale Methoden. Alle drei Methoden optimieren die Modellierung aus Effizienzgründen indirekt. Für die Implementierung werden bei jeder Strategie mehrere Modifikationen zusammengefasst, so dass in jeder Iteration jeweils ein bestimmter Prozentsatz der Modelle modifiziert wird. Als viertes wird noch eine direkte Optimierung der Modellierung vorgestellt, die aber nicht implementiert wird.

Güte der Modellierung

Es wird jenes Graphem ausgewählt, dessen Beitrag zur Modellierung der Trainingsdaten am geringsten ist. Das Maß ist die mittlere Graphem-Likelihood. Ausreißer nach unten weisen auf eine ungenügende Modellierung hin, die durch eine detaillierte Modellierung, also das Hinzufügen zusätzlicher Pfade, verbessert werden kann. Dieser Vorgang ähnelt dem Hinzufügen von Trainingsdaten bei der inkrementellen Variante des *Model Merging* [Sto94].

Während des Trainings wird die mittlere Graphem-Likelihood und ihre Varianz bestimmt. Sie lässt sich über den Viterbi-Pfad definieren. Sei in diesem t_a die erste Beobachtung des ersten Zustands s_a des Modells und t_e die letzte Beobachtung des letzten Zustands s_e, dann wird die normierte logarithmierte Graphem-Likelihood durch die Bildung der Differenz der Eintrage im Viterbi-Trellis $\delta_t(\cdot)$ berechnet:

$$p_{\text{graphem}} = \frac{1}{t_e - t_a} \cdot \left(\log \delta_{t_e}(s_e) - \log \delta_{t_a}(s_a)\right). \tag{13.1}$$

Im Rahmen des Baum-Welsh-Training kann der Viterbi-Pfad auch durch die Maxima des Forward-Backward-Trellis genähert werden. Die Aufgabe des Viterbi-Trellis übernimmt dann der Forward-Trellis.

Modellbewertung durch Abstand und Entropie

Die Auswahl der Grapheme erfolgt über eine lokale Analyse der Modellierung. Das entscheidende Maß für das Entfernen von Pfaden ist deren Ähnlichkeit, also der Abstand paralleler Modellpfade (Abschnitt 12.1.3). Es wird jenes Modell verkleinert, das die Pfade enthält, die den geringsten paarweisen Abstand (12.8) besitzen. Diese Pfade werden zusammengelegt.

Ein Modell wird dagegen erweitert, wenn die aktuelle Modellierung ungenau oder unsicher ist. Das kann durch eine hohe Varianz der Wahrscheinlichkeitsverteilungen angezeigt werden. Im konkreten Fall semikontinuierlicher Modelle wird eine unbestimmte, wenig diskriminative Modellierung aber anhand der Entropie der Emissionsgewichte (Abschnitt 12.2) ermittelt. Sie wird über alle Zustände des Allographmodells gemittelt:

$$H_{\text{graphem}} = -\frac{1}{N}\sum_i \sum_k b_i(k) \cdot \log b_i(k). \tag{13.2}$$

Die Generierung des neuen Pfades erfolgt mit den im nächsten Abschnitt beschriebenen Methoden.

Anzahl der repräsentierten Trainingsdaten

Auch die Häufigkeit der Repräsentation kann als Auswahlkriterium herangezogen werden, ob ein Graphem mehr oder weniger detailliert modelliert werden soll. Werden viele Daten von einem Modell repräsentiert, so motiviert dies eine genauere Modellierung, da sich die Qualität dieses Graphemmodells stärker auf das Gesamtergebnis der Schriftmodellierung auswirkt als bei selten auftretenden Allographen. Außerdem kann durch die Beachtung der Häufigkeit der Trainingsdaten der Schätzfehler der Parameter optimiert werden. [1]

Eigentlich handelt sich um ein globales Kriterium. Die Anzahl der repräsentierten Trainingsdaten kann aber lokal über die statische Häufigkeitsverteilung der Grapheme und die Eingangswahrscheinlichkeiten der Allographmodelle berechnet werden. Ein hoher Wert erhöht die Anzahl der Allographmodelle, bei einem niedrigen wird ein Pfad des Modells gelöscht.

Auswirkung auf die Güte des Gesamtmodells

Die direkte Methode zur Optimierung der Modellierung besteht darin, mögliche Änderungen der Modellierung nach ihrer Wirkung zu bewerten und auszuwählen. Es handelt sich um einen *greedy* Suchalgorithmus, da in jeder Iteration die Änderung gewählt wird, die gerade die größte Verbesserung des Modells bewirkt. Die beiden Schritte Auswahl und Modifikation der Graphemmodelle werden zusammengelegt. Sämtliche Modelle werden modifiziert und

[1] Es gibt auch das Argument, dass seltene Buchstaben höhere Diskriminanzeigenschaften haben und deshalb genauer modelliert werden sollten, während häufige Buchstaben wenig Bedeutung tragen und nur entsprechend ungenau erkannt werden müssen. Im Kontext dieser Arbeit soll es aber nicht weiter beachtet werden. Vgl. [Wil97a].

bewertet, aber nur diejenige Modifikation tatsächlich ausgewählt, bei der die mit der Häufigkeit der Trainingsdaten gewichtete Graphem-Likelihood (13.1) am meisten steigt. Folgende Schritte werden durchgeführt:

- Bestimmung der Likelihood der alten Graphemmodelle,
- Einfügen eines neuen Pfads bei jedem Graphemmodell,
- Training der neuen Modelle, und
- Bestimmung der Likelihood der neuen Graphemmodelle.

Die Komplexität kann nicht nur erhöht werden, in einer Iteration können auch sämtliche Modelle um einen Pfad vereinfacht werden. Die Auswahl fällt dann auf das Modell mit der geringsten Abnahme der Graphem-Likelihood.

Ein großer Nachteil der direkten Optimierung besteht im hohen Aufwand an Rechenzeit. Sie wurde deshalb nicht implementiert.

13.1.2 Initialisierung neuer Allographmodelle

Nachdem Grapheme zur Modifikation ausgewählt worden sind, wird ein neues Buchstabenmodell generiert, indem ein neuer Pfad erzeugt oder ein alter Pfad entfernt wird. Damit nach dem Neutraining der Parameter ein aussagekräftiger Vergleich mit der Vorgängermodellierung angestellt werden kann, muss das neue Buchstaben-HMM sinnvoll initialisiert werden.

Als verhältnismäßig einfach stellt sich die Verringerung der Pfadanzahl dar. Es gibt zwei Möglichkeiten:

- Ein Pfad wird einfach entfernt. Das ist dann sinnvoll, wenn die Modellierung des speziellen Allographen als fehlerhaft oder unwichtig angesehen wird. Die Methode wird deshalb bei der Auswahl nach Graphem-Likelihood oder Häufigkeit verwendet.
- Zwei Pfade werden zusammengefasst, wenn ihre *Ähnlichkeit* der Grund für die Vereinfachung der Modellierung ist. Das ist bei der Auswahl nach Modellabstand der Fall. Sie werden „gemischt“, indem die Mittelwerte der Emissionswahrscheinlichkeiten gebildet werden, gewichtet mit der jeweiligen Eingangswahrscheinlichkeit. Bei unterschiedlichen Längen der beiden Pfade wird zunächst die Länge des zu generierenden Pfades bestimmt, indem wieder das gewichtete Mittel der Ursprungslängen genommen und gerundet wird. Anschließend werden beide Pfade durch Mischen (vgl. Abschnitt 11.2.2) auf die entsprechende Länge gebracht und gemittelt.

Das Hinzufügen neuer Pfade ist schwieriger, da ein Modell für eine bisher nicht repräsentierte Schreibweise erstellt werden muss. Auch hier können zwei Vorgehensweisen unterschieden werden.

- Der neue Zustandspfad wird zufällig initialisiert. Die Parameter können entweder vollständig zufällig bestimmt werden, oder es kann ein bestehender Pfad ausgewählt werden, der kopiert und um einen Zufallswert verschoben wird, was letztendlich einer Aufteilung des ausgewählten Pfades entspricht. Der Nachteil einer zufallsgesteuerten Initialisierung ist die zu erwartende langsame Konvergenz, ein Vorteil des Verfahrens besteht dagegen in der einfachen Realisierung.

- Schlecht bewertete Trainingsdaten werden als Saatpunkt für das neue Modell verwendet [Sto94, Li00a]. Alternative Schreibweisen können auf diese Weise schnell gefunden werden. Der entscheidende Nachteil ist aber, dass Fehler in den Trainingsdaten, ob falsche Label oder Bildstörungen, die Ergebnisse negativ beeinflussen. Ganz seltene Schreibweisen würden zudem überrepräsentiert sein, da die Häufigkeit der Schreibweisen nicht in die Initialisierung mit eingeht.

Bei allen Methoden zur Modellauswahl wird der neue Zustandspfad durch die Aufspaltung eines bestehenden Pfades erzeugt. Bei der Auswahl nach Entropie und Häufigkeit wird dazu der bestbewertete und häufigste Pfad verwendet. Erfolgt die Auswahl nach Graphem-Likelihood, wird der Pfad gewählt, der die größte Variabilität in den Trainingsdaten aufweist, gemessen anhand der Varianz der Likelihood.

13.1.3 Neutraining der modifizierten Modelle

Für das Training neuer Modelle könnte es sinnvoll sein, mit dem Viterbi-Pfad die Segmentgrenzen zu bestimmen und zu fixieren, so dass man separate Trainingsmengen für die einzelnen Grapheme hat. Für dieses Vorgehen spricht, dass bei der Berechnung der Differenz der Graphem-Likelihood beide Werte auf derselben Grundlage berechnet werden und dadurch exakt vergleichbar sind. Bei der Einführung neuer Allographmodelle kann nämlich nicht vorausgesetzt werden, dass die Segmentierung beim Trainieren erhalten bleibt.

Hier wird jedoch davon ausgegangen, dass sich bei einer modifizierten Modellierung die Segmentierung auch verbessert. Auf die Fixierung der Segmentgrenzen beim Neutraining wird deshalb verzichtet.

13.2 Vergleich der Auswahlstrategien

Es werden zunächst Untersuchungen zum Vergleich der im vorigen Abschnitt vorgestellten Auswahl- und Modifikationsstrategien angestellt. Das Ziel besteht nicht primär in der Optimierung der Modellstrukturen, sondern in der Verdeutlichung der Unterschiede der verschiedenen Verfahren. Es werden deshalb Experimente durchgeführt, bei denen mit den einzelnen Strategien die Anzahl der Allographen ausschließlich erhöht oder vermindert wird. Die Ergebnisse wurden in [Sch03a] veröffentlicht.

Für jedes der acht getesteten Projekte aus Abschnitt 6.4.1 wurden alle sechs in Abschnitt 13.1.1 vorgestellten Verfahren getestet. Gestartet wird beim Hinzufügen von Schreibvarianten mit einem System mit jeweils einem Allographenmodell pro Graphem, beim Entfernen modelliert das Basissystem jeweils sechs oder acht Allographen: Sechs für die Visualisierung der Modelle, acht für die Bestimmung der Erkennungsraten. In jedem Iterationsschritt des Algorithmus (vgl. Abbildung 13.1) werden jeweils 30 Prozent aller Modelle verändert. Es werden maximal 30 Iterationen durchgeführt.

Um die besten Verfahren zu bestimmen, werden die Plausibilität und die quantitative Auswertung der Ergebnisse bewertet. Die Plausibilität wird durch die Visualisierung der generierten Modellstrukturen geprüft, die quantitative Auswertung erfolgt über die Erkennungsleistung. Anhand der Ergebnisse werden jeweils die besten Strategien für das Hinzufügen und Entfernen von Allographmodellen bestimmt. Durch die visuelle Bewertung hat diese Auswahl auch einen subjektiven Charakter.

13.2.1 Visueller Vergleich

Eine Analyse der Bilder der generierten Graphemmodelle gibt interessante Einblicke in die Eigenschaften der unterschiedlichen Auswahlstrategien. Am Beispiel des Erkennungssystems für die CEDAR-Adressdaten, die ausschließlich aus Buchstaben bestehen, werden die Iterationen, die die besten Erkennungsergebnisse auf den Testdaten liefern, ausgewählt und die resultierenden Modelle diskutiert. Die Abbildungen 13.2 bis 13.7 zeigen die Visualisierungen der Modelle. Links ist jeweils das Ausgangssystem, rechts das Ergebnis der Modellanpassung zu sehen. [2]

Hinzufügen von Allographen

Beim Hinzufügen von Allographen erhält man bei Auswahl nach Entropie einen sehr guten visuellen Eindruck, wie man in Abbildung 13.2 sieht. „Einfache“ Zeichen wie das „c“ sind nur zweimal modelliert, während „a“ und „d“ ein Maximum von zehn unterschiedlichen Varianten haben, was ihr häufiges und variables Auftreten widerspiegelt. Im Vergleich zu den anderen Systemen weisen die Modellbilder der Schreibvarianten untereinander recht deutliche Unterschiede auf, wenn man berücksichtigt, dass aufgrund der Merkmalsgenerierung die Lage der Zeichen zu den Schreiblinien einen großen Einfluss auf die vom System „gesehenen“ Zeichen hat. Ansonsten ähnliche Modellbilder unterscheiden sich in der vertikalen Position der Zeichen, was zum Beispiel bei „a“, „c“ oder „e“ recht deutlich zu sehen ist. Insgesamt ist das Ergebnis der Modellierung nach Entropie plausibel.

Im nächsten Abschnitt wird gezeigt, dass durch Hinzufügen von Allographen aufgrund ihrer Häufigkeit die besten Erkennungsleistungen erzielt werden. Die Modellierung in Ab-

[2] Leere schmale Rahmen zeigen modellierte Schreibvarianten an, für die in der begrenzten Teststichprobe keine Entsprechung gefunden wurde. In den Abbildungen betrifft dies nur den verhältnismäßig seltenen Buchstaben „j“.

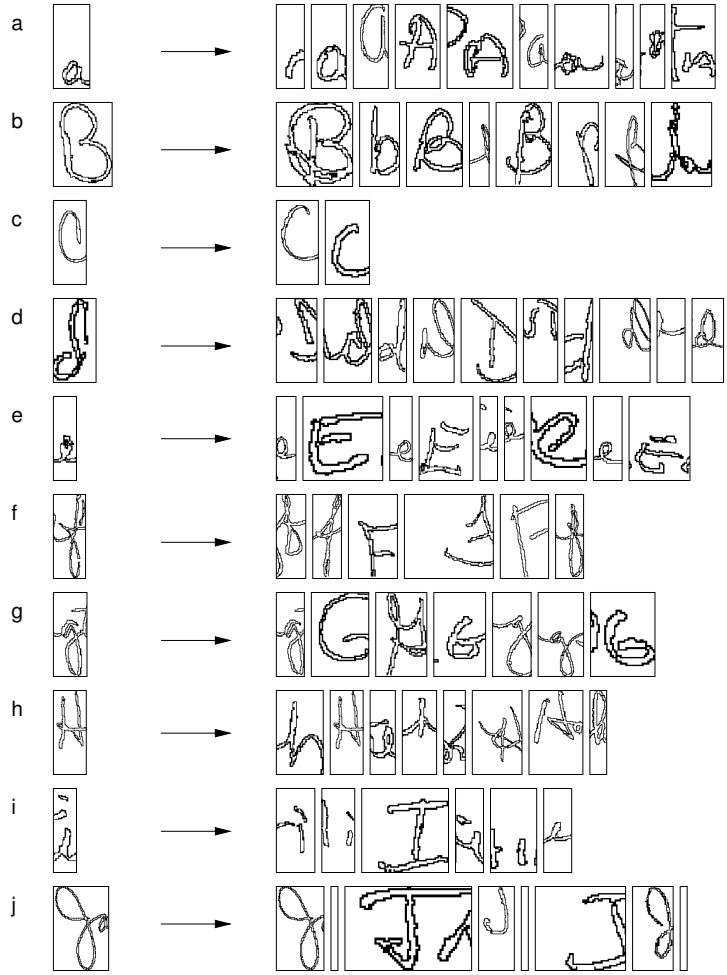

Abbildung 13.2: Hinzufügen von Allographen nach Entropie: Die Zeichenkomplexität wird plausibel abgebildet, die paarweisen Ähnlichkeiten innerhalb der Modelle sind gering.

bildung 13.3 ist aber vergleichsweise wenig intuitiv: Häufige Zeichen, wie „a“, „e“ und „i“, werden bis zu 16 mal modelliert und produzieren ähnliche Allographen, während andere, wie „f“, „g“ und „j“, übermäßig vereinfacht sind, wie der Vergleich mit dem vorigen System in Abbildung 13.2. zeigt. Die Komplexität der Zeichen wird nicht richtig widergespiegelt, wie man besonders deutlich am Buchstaben „j“ sieht.

Ausgewogene Ergebnisse erhält man beim Hinzufügen nach Likelihood, zu sehen in Abbildung 13.4. Jedes Modell wird durch eine Mindestanzahl von Allographen repräsentiert, die die wichtigsten Schreibweisen abdecken. Auch hier werden komplexe Zeichen wie das „a“ bis zu 15 mal modelliert, allerdings geschieht dies nur bei Zeichen, die tatsächlich eine hohe optische Variabilität in der Gestalt aufweisen, wie zum Beispiel auch die in der Abbildung nicht gezeigten Buchstaben „x“ und „y“. Die Ergebnisse entsprechen am ehesten der Modellierung von Hand: In frühen Iterationen weisen die meisten Buchstabenmodelle zwei bis vier Allographen auf, was die manuelle Strukturvorgabe des Projekts abbildet. Die gute Qualität dieser Ergebnisse lässt sich unter anderem damit erklären, dass mit der Like-

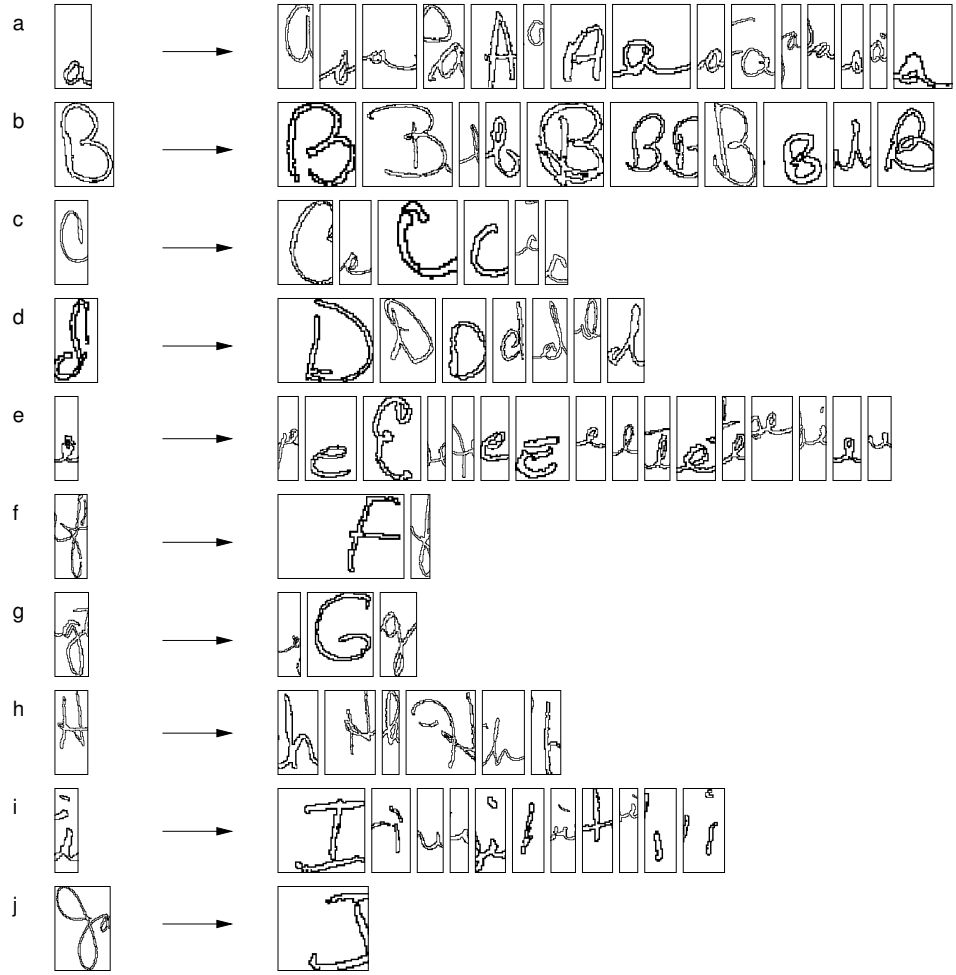

Abbildung 13.3: Hinzufügen von Allographen nach Häufigkeit: Die Modellkomplexität wird nicht beachtet, die Detaillierung der Modelle ist unausgewogen.

lihood dasselbe Kriterium für die Strukturwahl verwendet wird wie für die Bestimmung der HMM-Parameter.

Entfernen von Allographen

Betrachtet man die Iterationen, in denen Allographen entfernt werden, erzeugt die Auswahl nach Likelihood die schlechtesten Ergebnisse in allen Projekten, wie man in Abbildung 13.5 sieht. Modelle mit niedriger Likelihood werden durch das Entfernen von Allographen noch weiter verschlechtert, so dass einige Modelle bis auf eine einzige Schreibvariante reduziert werden. Wichtige und komplexe Zeichen wie „a“ und „e“ bestehen schließlich nur noch aus wenigen oder einer einzigen Variante. Das Ergebnis der Modellierung ist nicht akzeptabel. Die dem Auswahlkriterium zugrundeliegende Annahme, dass sich eine schlechte Modellierung durch das Entfernen einzelner Allographen auch verbessern kann, ist bei der geringen Anzahl von Allographen, die hier modelliert werden, nicht haltbar.

Die Auswahl nach Häufigkeit in Abbildung 13.6 zeigt ähnliche Ergebnisse wie das Hinzufügen nach Häufigkeit. Die häufigsten Zeichen, wie „a“, „b“ und „e“, werden ein Ma-

Abbildung 13.4: Hinzufügen von Allographen nach Graphem-Likelihood: Die wichtigen Schreibvarianten aller Zeichen sind repräsentiert.

ximum von fünf mal modelliert, ein seltenes Zeichen wie „j“ wird auf eine Alternative reduziert. Da das Ausgangssystem maximal nur sechs Allographen modelliert, ist die Unausgewogenheit der Modellierung nicht so stark wie beim Hinzufügen nach Häufigkeit. Als pragmatischer Ansatz zur Reduktion der Modellkomplexität bei bestmöglicher Erhaltung der Erkennungsleistung scheint die Methode geeignet zu sein, auch wenn sie die Komplexität der Zeichen nicht beachtet und die Modellierung deshalb nicht sehr plausibel ist.

Einen sehr guten visuellen Eindruck hinterlässt das Entfernen nach Abstand, gezeigt in Abbildung 13.7. Wichtige Schreibweisen sind vertreten, indem komplexe Zeichen wie „a“, „g“ und „h“ feinstmöglich modelliert sind, das einfache Zeichen „c“ hingegen auf drei Varianten reduziert wird. Es gibt keine doppelten Repräsentationen durch paarweise Ähnlichkeit. Angesichts der offensichtlichen Nachteile der beiden anderen vorgeschlagenen Strategien zur Allographreduktion erzeugt die Reduktion nach Modellabstand eindeutig die plausibelsten Ergebnisse.

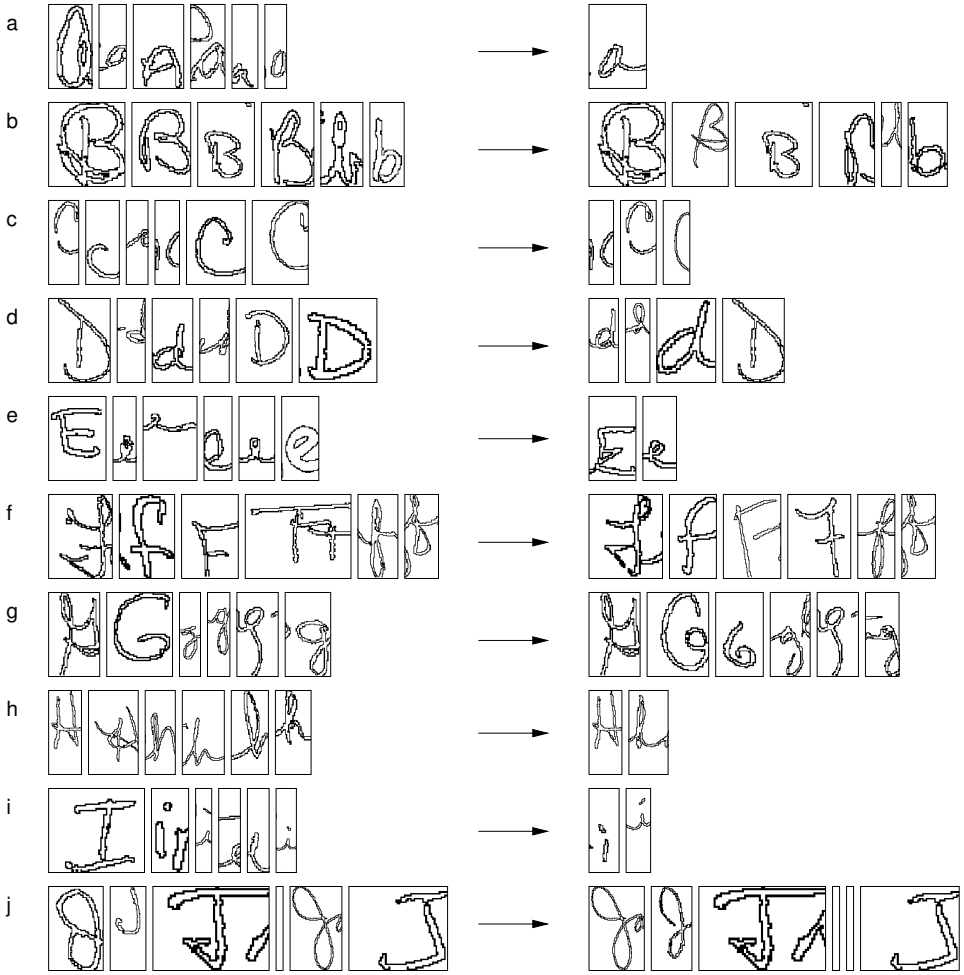

Abbildung 13.5: Entfernen von Allographen nach Likelihood: Schlecht bewertete Zeichen werden bis auf eine Variante reduziert, da sich eine niedrige Zeichenbewertung durch Entfernen von Varianten verstärkt.

13.2.2 Erkennungsraten

Für die quantitative Auswertung der Auswahlstrategien werden die Erkennungsraten bestimmt, die mit den sechs verschiedenen Strategien erreicht werden. Sie werden in den Tabellen 13.1 und 13.2 gezeigt. Der Startpunkt für das Entfernen von Allographen sind jetzt Modelle mit acht Schreibvarianten; insgesamt werden jeweils 30 Iterationen durchgeführt. Die Erkennungsraten der Tabellen beziehen sich auf von den Trainingsdaten unabhängige Testdaten.

Wie bei der Längenadaption in Kapitel 11 werden zwei verschiedene Kriterien für die Auswahl der besten Systeme getestet. Das beste Iteration wird entweder aufgrund maximaler Likelihood oder maximaler Erkennungsrate auf den Trainingsdaten bestimmt. Erneut werden in allen Fällen die besten Erkennungsraten erreicht, wenn die Auswahl aufgrund der Trainings-Erkennungsrate erfolgt. Nur sie werden im folgenden weiter betrachtet. Wählte man direkt die Systeme mit den absolut besten Ergebnissen auf den Testdaten, dann könnte mit 83,2 Prozent Erkennungsrate noch ein geringfügig besseres Ergebnis erzielt werden. Im

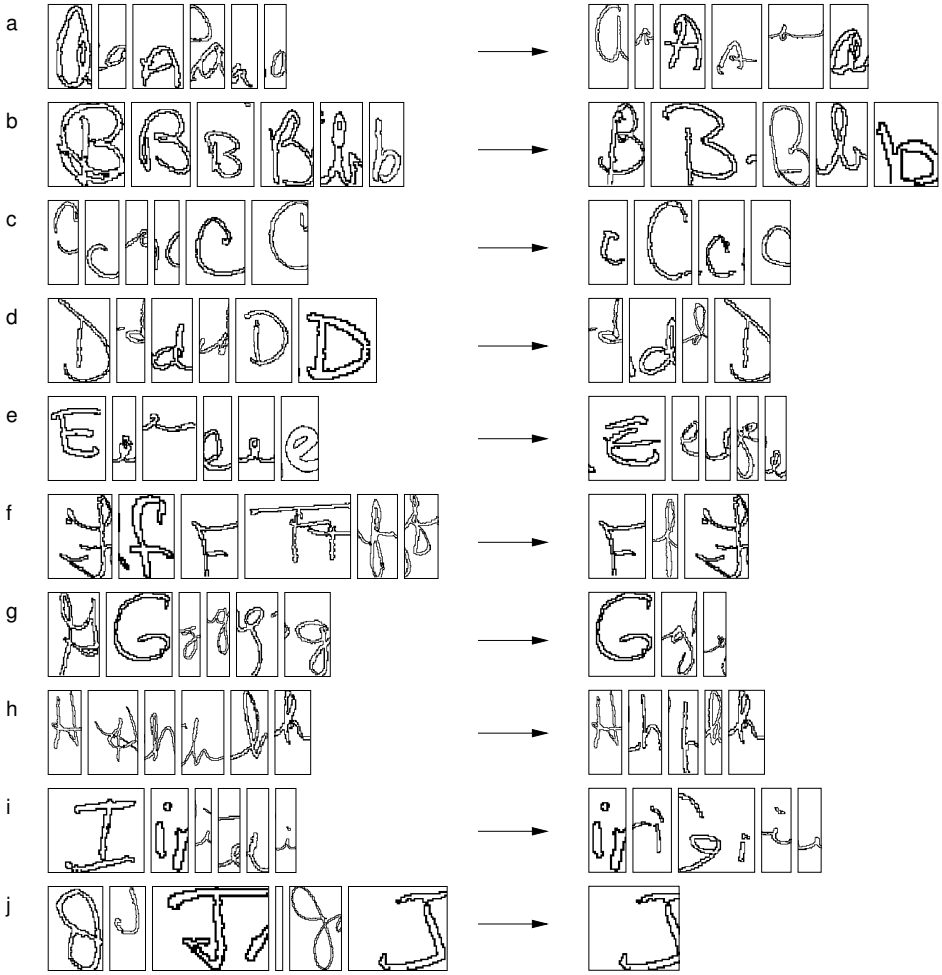

Abbildung 13.6: Entfernen von Allographen nach Häufigkeit: Wie beim Hinzufügen nach Häufigkeit wird die Zeichenkomplexität nicht beachtet. Durch den gewählten Startpunkt ist das Ungleichgewicht in den Modellen jedoch nicht so stark.

folgenden wird untersucht werden, ob mit den Bayes'schen Kriterien zur Modellwahl dieses Potential ausgeschöpft werden kann.

Beim schrittweisen Entfernen werden bessere Ergebnisse erzielt als beim Hinzufügen von Schreibweisen. Meist werden bei der Reduktion des zu Beginn sehr detaillierten Modells mit gleichförmiger Allographenverteilung nur wenige Iterationen durchgeführt. Die besten Ergebnisse werden zwischen der 3. und 6. Iteration erzielt. Im Gegensatz dazu müssen bei der schrittweisen Erweiterung des primitiven Modells viele Iterationen durchlaufen werden, bis eine gute Modellierung erreicht ist. Dabei besteht die Gefahr, dass sich eine schlechte Aufteilung und Segmentierung der Trainingsdaten aufgrund der anfänglich übermäßig einfachen Modelle bis zu den späteren Iterationen fortpflanzt.

Die Ergebnisse der schrittweisen Verfeinerung der Systeme sind in Tabelle 13.1 dargestellt. Eine maximale durchschnittliche Erkennungsrate von 81,5 Prozent wird erreicht, was einer relativen Verbesserung von 5,7 Prozent im Vergleich zur Auswertung der Ursprungssysteme in Abschnitt 6.4 entspricht. Die Kriterien Likelihood und Häufigkeit unterscheiden

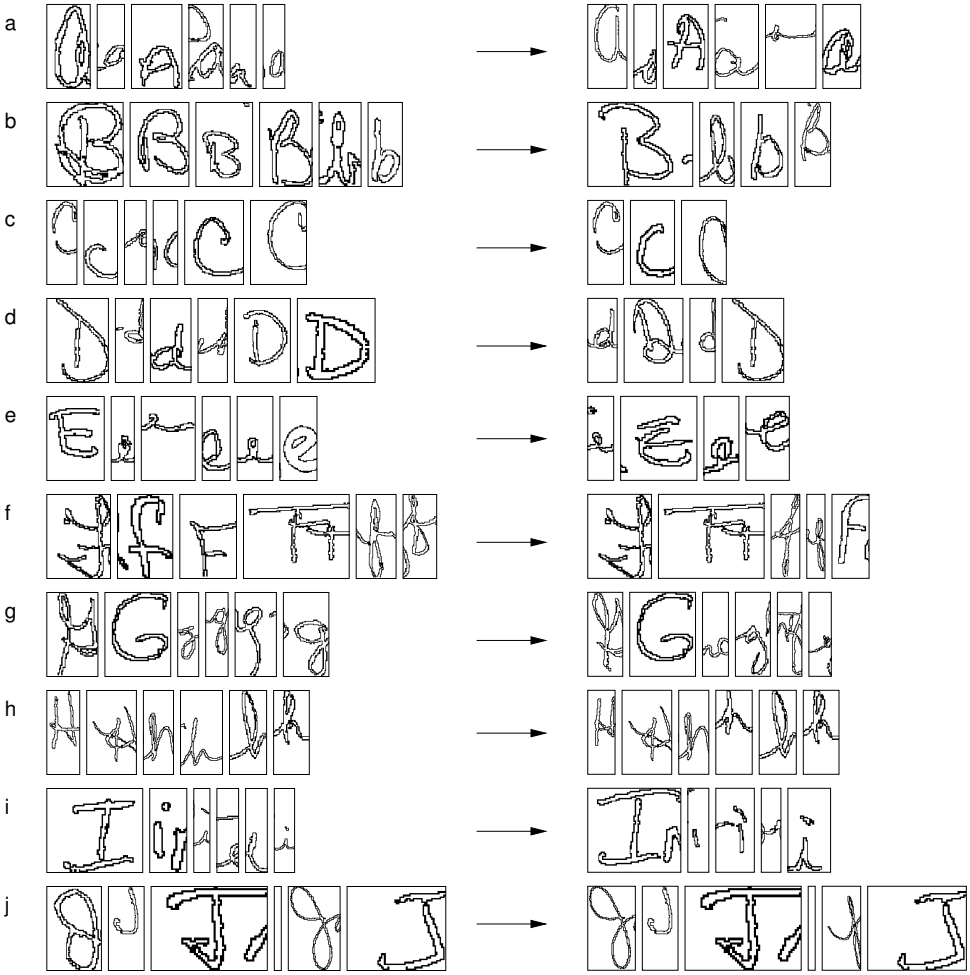

Abbildung 13.7: Entfernen von Allographen nach Abstand: Die Modellierung ist plausibel; die wichtigsten Zeichenvarianten sind vertreten.

sich in den Ergebnissen kaum, nur die Entropie fällt etwas ab. Die Leistung der Systeme, bei denen Zustandspfade entfernt wurden, ist in Tabelle 13.2 zu sehen. Mit 82,5 Prozent ist die Erkennungsrate etwas höher, die durchschnittliche Verbesserung beträgt jetzt 7,5 Prozent. Die Unterschiede zwischen den verschiedenen Auswahlstrategien sind minimal, da die besten Systeme nach wenigen Iterationen bestimmt sind und sich deshalb auch in der Struktur nicht stark voneinander unterscheiden.

Für einen weiteren Vergleich der verschiedenen Strategien zur Auswahl und Modifikation beim Hinzufügen und Entfernen von Allographen, sind in Abbildung 13.8 Erkennungsraten gegen die Anzahl der Allographen beispielhaft für das Erkennungssystem deutscher Postleitzahlen aufgetragen. Die Strategien zum Hinzufügen von Varianten sind mit durchgezogenen Linien gezeichnet und starten von einem gemeinsamen Punkt links, die Strategien zum Entfernen sind gepunktet gezeichnet und starten rechts. Aus diesem Diagramm kann die beste Anzahl an Modellen leicht abgelesen werden; im Beispiel beträgt sie etwa 60 Modelle. Interessanterweise erreicht die Erkennungsrate beim Entfernen von Modellen bei etwa 25 Allographen ein Minimum, während beim Hinzufügen bessere Konfigurationen für diese

		Hinzufügen von Pfaden nach ...		
Projekt	Startpunkt	Entropie	Likelihood	Häufigkeit
Arab Emirate (Emirate)	76,9	77,0 / 77,0	55,7 / 82,4	60,5 / 82,4
Kanada (Adressen)	93,4	93,7 / 94,1	94,0 / 94,4	94,2 / 94,9
Deutschland (Adressen)	92,8	93,6 / 93,5	93,6 / 93,9	92,6 / 92,6
Deutschland (PLZ)	69,1	71,2 / 72,2	74,5 / 75,9	76,2 / 77,4
USA (Adressen)	81,2	83,8 / 83,4	83,6 / 83,6	80,3 / 81,5
USA (ZIP)	60,8	63,5 / 64,4	69,2 / 69,2	69,6 / 71,0
USA-CEDAR (Städte)	84,2	84,2 / 84,2	85,7 / 84,9	84,5 / 84,5
USA-CEDAR (ZIP)	58,2	62,8 / 63,0	64,1 / 66,4	67,8 / 67,8
Mittlere Erkennungsrate	77,1	78,7 / 79,0	77,6 / 81,3	78,2 / 81,5
Relative Verbesserung	—	2,08 / 2,46	0,64 / 5,44	1,42 / 5,70

Tabelle 13.1: Worterkennungsraten (in Prozent), ausgewählt nach Likelihood / Erkennungsraten der Trainingsdaten. Ausgehend von einem System mit einer Schreibvariante pro Zeichen werden in 30 Iterationen Allographen hinzugefügt.

		Entfernen von Pfaden nach ...		
Projekt	Startpunkt	Abstand	Likelihood	Häufigkeit
Arab Emirate (Emirate)	76,9	86,4 / 86,4	86,4 / 86,4	86,4 / 86,4
Kanada (Adressen)	93,4	95,2 / 95,4	95,7 / 95,4	95,2 / 95,4
Deutschland (Adressen)	92,8	94,4 / 94,5	94,4 / 94,4	94,6 / 94,4
Deutschland (PLZ)	69,1	78,3 / 78,3	76,0 / 76,9	75,3 / 77,6
USA (Adressen)	81,2	84,3 / 84,6	84,6 / 84,6	84,1 / 84,3
USA (ZIP)	60,8	69,9 / 70,7	69,4 / 70,0	69,7 / 70,5
USA-CEDAR (Städte)	84,2	85,1 / 85,1	85,0 / 84,8	84,2 / 84,9
USA-CEDAR (ZIP)	58,2	68,3 / 68,3	68,0 / 70,6	69,0 / 68,5
Mittlere Erkennungsrate	77,1	82,7 / 82,9	82,4 / 82,9	82,3 / 82,8
Relative Verbesserung	—	7,26 / 7,52	6,87 / 7,52	6,74 / 7,39

Tabelle 13.2: Worterkennungsraten (in Prozent), ausgewählt nach Likelihood / Erkennungsraten der Trainingsdaten. Ausgehend von einem System mit acht Schreibvarianten pro Zeichen werden in 30 Iterationen Allographen entfernt.

Anzahl von Modellen gefunden werden. Die kann damit erklärt werden, dass bei der Modellreduktion nach vielen Iterationen unplausible Modellierungen entstehen, bei denen der Detaillierungsgrad der Modelle stark unterschiedlich ist, was die Erkennung negativ beeinflusst.

Bei den folgenden Experimenten wird unter anderem aus diesem Grund von einem Startsystem ausgegangen, das eine feste, mittelgroße Anzahl Allographen modelliert, die anschließend in wenigen Iterationen modifiziert wird, indem Modelle sowohl hinzugefügt als auch entfernt werden. Für die Steuerung dieser Schritte wird jeweils nur eine der hier untersuchten Strategien verwendet. Es sind die Methoden, mit denen sowohl bei der Erkennung,

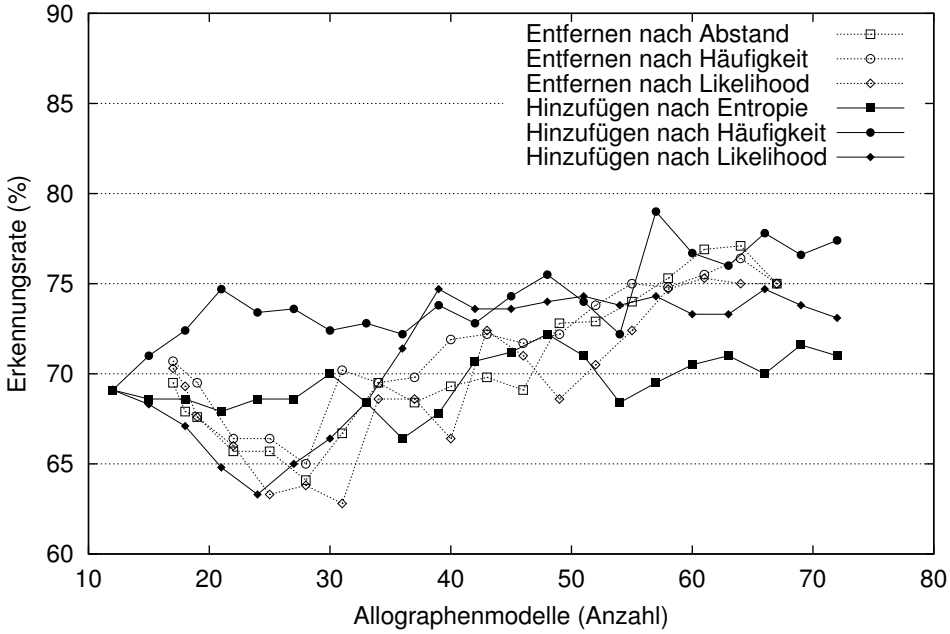

Abbildung 13.8: Erkennungsraten für verschiedene Anzahl von Allographen während der Iterationen zur Modelladaption des Systems für deutsche Ziffern.

als auch bei der Prüfung der Plausibilität der generierten Modellstrukturen gute Ergebnisse erzielt werden:

- Hinzugefügt werden Modelle nach dem Likelihood-Kriterium. Die Entropie erzeugt zwar eine sehr plausible Modellierung, die Erkennungsraten fallen im Vergleich zu Likelihood und Häufigkeit aber leicht ab. Deren Ergebnisse sind vergleichbar, die Modellierung nach Likelihood ist aber deutlich plausibler.

- Die Reduktion der Modellkomplexität erfolgt nach dem Modellabstand. Die Erkennungsleistung der generierten Systeme aller Kriterien sind vergleichbar, allein über den Abstand wird aber auch nach mehreren Iterationen noch eine plausible Modellierung erzeugt.

13.3 Generalisierung

Eine gute Modellierung zeichnet sich dadurch aus, dass mit einer Menge von repräsentativen Trainingsdaten die Modellparameter so trainiert werden können, dass auch unbekannte Testdaten durch das Modell richtig repräsentiert werden. Das Modell soll in der Lage sein, von den Trainingsdaten zu abstrahieren. Diese Fähigkeit zur Generalisierung ist Gegenstand der Überlegungen zur Modellwahl in Kapitel 9. Sie wird im folgenden anhand der iterativen Anpassung der Allographmodelle untersucht.

Zunächst wird an künstlichen Beispielen nachgewiesen, dass durch übermäßig detaillier-

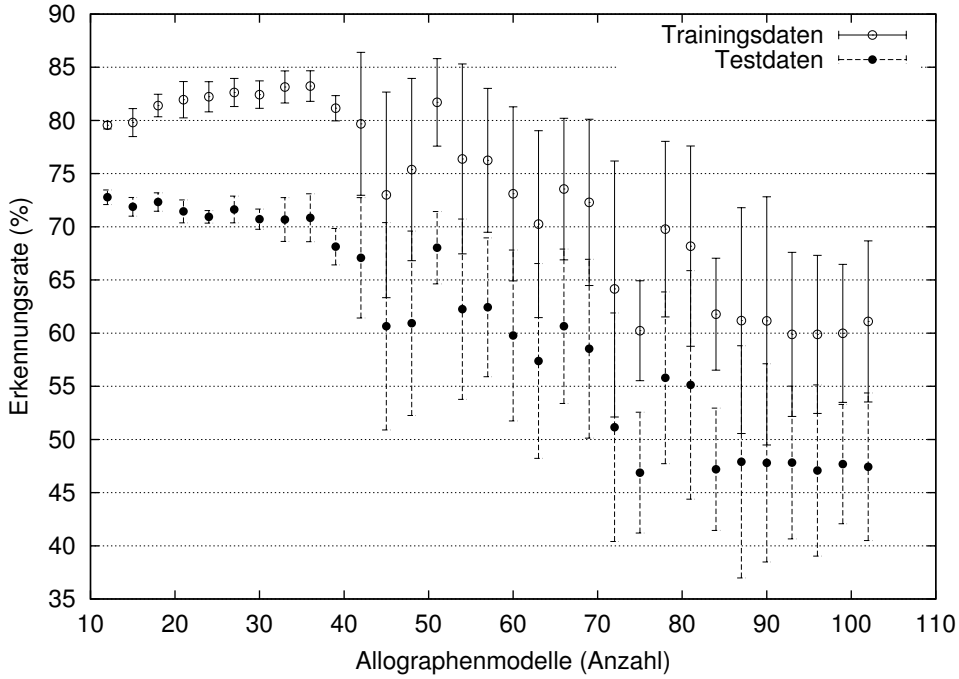

Abbildung 13.9: Überadaption durch Schätzfehler: Entwicklung der Leistung eines Erkennungssystems für deutsche Ziffern, trainiert mit nur 1000 deutschen Postleitzahlen. Die Werte wurden mit einer zehnfachen Kreuzvalidation bestimmt.

te Modellierungen die Güte der Modelle tatsächlich sinken kann. Anhand von realen Beispielen wird dann die graduelle Abhängigkeit der Generalisierung von der Modellkomplexität demonstriert. Schließlich wird untersucht, ob durch die Anwendung des Modellwahlkriteriums aus Abschnitt 12.3 das „optimale“ Modell gefunden werden kann.

13.3.1 Schätzfehler

In Abbildung 13.9 wird demonstriert, wie die Güte eines Erkennungssystems sinkt, wenn das zugrundeliegende Modell zu detailliert ist. In einem Experiment wird die Anzahl der Trainingsdaten für ein System deutscher Postleitzahlen künstlich auf 1000 reduziert. Nach dem Likelihood-Kriterium wird die Anzahl der Allographenmodelle schrittweise erhöht. Zur statistischen Absicherung werden die Ergebnisse zehnfach kreuzvalidiert, das Experiment wird also zehn mal mit unterschiedlichen Aufteilungen von Test- und Trainingsdaten durchgeführt.

Werden insgesamt mehr als 40 Allographenmodelle für die Repräsentation der zehn Ziffern verwendet, sinkt die Erkennungsleistung des Systems deutlich, und zwar sowohl auf den Testdaten, als auch auf den Trainingsdaten. Dies liegt an dem in Abschnitt 9.1.2 beschriebenen Effekt des Schätzfehlers. Für die Bestimmung der hohen Anzahl an Modellparametern sind nicht mehr genügend Daten vorhanden. Die Anzahl der Trainingsdaten bestimmt die Obergrenze der Modellkomplexität.

Die hohe Schwankung der gemessenen Erkennungsraten lässt sich mit der Art der Klas-

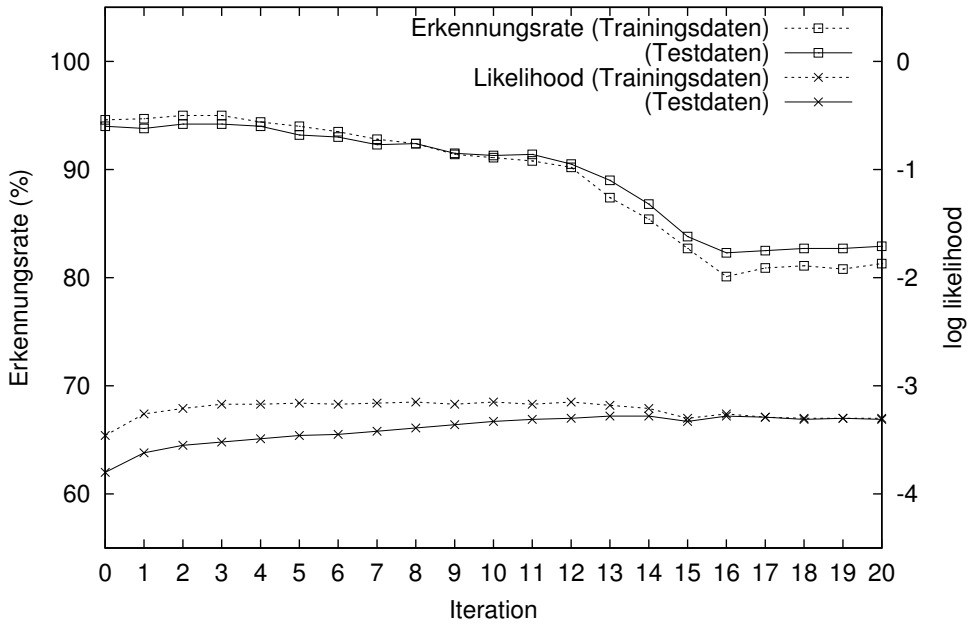

Abbildung 13.10: Entwicklung von Likelihood und Erkennungsrate während der iterativen Vereinfachung des Modells. Allographen des Modells für deutsche Adressdaten werden nach dem Abstandskriterium entfernt.

senaufteilung und -clusterung des Systems erklären, die in Abschnitt 5.4 beschrieben ist. Es hängt von der Aufteilung der Schreibvarianten in den Trainingsdaten ab, ob für die den Zuständen zugeordneten Klassen genügend Vektoren für die Bestimmung einer invertierbaren Kovarianzmatrix vorhanden sind. Nicht invertierbare Klassen werden geclustert; die Erkennungsleistung hängt davon ab, wie viele Klassen, die zu unterschiedlichen Ziffernmodellen gehören, zusammengelegt werden. Bei wenigen Trainingsdaten hat deshalb die konkrete Auswahl große Auswirkungen auf die Erkennungsleistung des Systems.

13.3.2 Überanpassung

Das eigentliche Problem bei den Fragen zur Modellwahl in Kapitel 9 ist die *Überanpassung* des Modells an die Trainingsdaten. Wann ist die Schriftmodellierung so detailliert, dass das Schätzen der Parameter einem „Auswendiglernen“ der Trainingsdaten entspricht? Anhand eines Beispiels lässt sich der graduelle Charakter der Generalisierung zeigen. Anschließend wird untersucht, wie für ein Modell bei steigender Komplexität eine Sättigung in der Generalisierbarkeit gefunden werden kann.

Abbildung 13.10 zeigt die Entwicklung von Likelihood und Erkennungsrate des Systems für deutsche Adressdaten, aus dessen Schriftmodell iterativ Allographen entfernt werden. Die Kurven sind jeweils sowohl für Trainingsdaten als auch für Testdaten eingezeichnet. Besonders deutlich ist der Generalisierungseffekt bei den Likelihood-Kurven: Die Differenz zwischen der Likelihood der Trainings- und Testdaten wird immer geringer, je weniger kom-

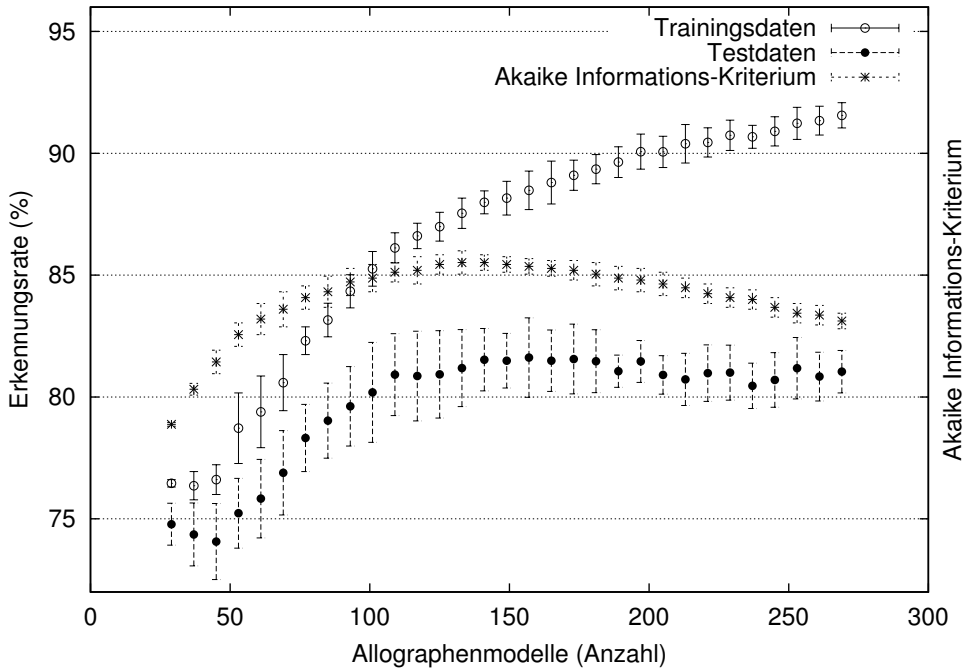

Abbildung 13.11: Sättigung und Modellwahlkriterien für US-Adresslesesystem (CEDAR), Allographen werden nach dem Likelihood-Kriterium hinzugefügt.

plex das Modell ist. Ungefähr ab Iteration 15 ist kein Unterschied zwischen Training und Test mehr zu sehen; das Modell generalisiert perfekt. Bei der Erkennungsrate ist der Effekt ähnlich. Mit abnehmender Modellgröße nimmt die Erkennungsleistung zwar ab, das Verhältnis von Test und Training verbessert sich aber zugunsten der Testdaten. [3]

Lässt sich quantitativ bestimmen, wann die Komplexität eines Modells den Punkt erreicht hat, an dem es optimal generalisiert? Zur Beantwortung dieser Frage wird die Entwicklung der Modelle der beiden CEDAR-Erkennungssysteme (Adress- und ZIP-Erkennung) bei schrittweiser Verfeinerung genauer untersucht. Dazu wird eine vierfache Kreuzvalidierung auf den bisher verwendeten Trainings- und Testdaten durchgeführt. Ausgehend von einem primitiven Modell werden Allographen nach dem Likelihood-Kriterium hinzugefügt. Die Entwicklung der Erkennungsraten ist in den Abbildungen 13.11 und 13.12 zu sehen.

Sättigungseffekte sind in beiden Systemen deutlich zu erkennen. Während die Erkennungsrate auf den Trainingsdaten im untersuchten Bereich kontinuierlich wächst, erreicht die Test-Erkennungsrate in beiden Fällen ein Plateau, auch wenn im Ziffern-System (Abbildung 13.12) die Rate weiterhin leicht steigt. Es ist allerdings schwierig, eine Kante des Plateaus zu bestimmen; gerade bei den Ziffern übersteigt die Standardabweichung der Erkennungsraten die Differenz zwischen den Iterationen. Die theoretischen Überlegungen zur Generalisierung der Modelle lassen sich mit diesen Experimenten qualitativ bestätigen.

[3] Die Erkennungsrate auf den Testdaten ist schließlich sogar *besser* als auf den Trainingsdaten. Die ist jedoch auf die Verwendung eines kleineren Wörterbuch beim Test zurückzuführen.

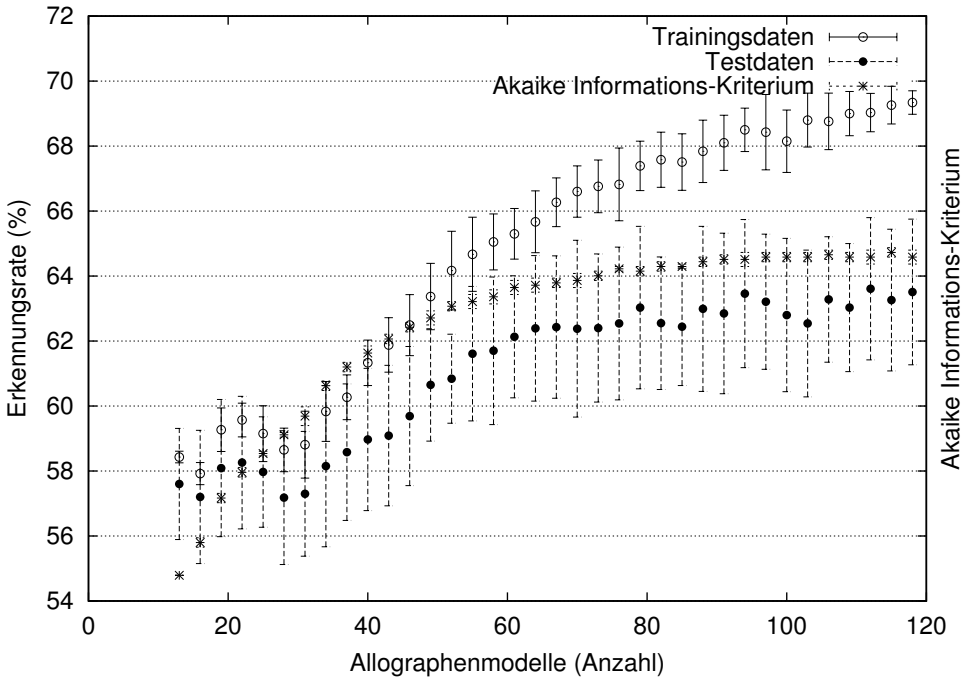

Abbildung 13.12: Sättigung und Modellwahlkriterien für US-ZIP-Lesesystem (CEDAR), Allographen werden nach dem Likelihood-Kriterium hinzugefügt.

13.3.3 Korrelation mit Modellwahlkriterien

In Kapitel 9 wurde mit dem Bayes'schen Formalismus ein theoretisches Maß für die optimale Wahl eines Modells vorgestellt. In Abschnitt 12.3 wurde es für HMM-Strukturen konkretisiert und die Berechnung des Akaike Informations-Kriteriums für das Schriftmodell definiert. Im folgenden wird untersucht, wie gut sich mit dem Kriterium die Fähigkeit des Systems zur Generalisierung beschreiben lässt.

Das Akaike Informations-Kriterium ist in den Abbildungen 13.11 und 13.12 neben den Erkennungsraten aufgetragen. In beiden Fällen bildet es den Verlauf der Test-Erkennungsrate qualitativ nach. Die Relevanz der Überlegungen zur Modellwahl wird dadurch bestätigt, und es zeigt sich, dass trotz der groben Abschätzung der Parameter für die Herleitung von Gleichung (12.14) plausible Ergebnisse erzeugt werden.

Man sieht bei einer genaueren Analyse der Ergebnisse aber auch, dass die Berechnung der Modellwahlkriterien nicht in eindeutiger Weise eine „beste“ Modellierung definiert. Modelliert man die Unsicherheit bei der Schätzung der Anzahl der freien Parameter in Abschnitt 12.3 mit einem variablen Faktor, dann hat die Wahl dieses Faktors eine entscheidende Bedeutung für die Position des Maximums. In der Praxis entsteht durch die Beachtung der Modellwahlkriterien also kein Vorteil. Die sicherste Methode, die Güte der Modellierung zu bestimmen, bleibt die direkte Bewertung des Modells mit einer Evaluierungsstichprobe. Meist wird jedoch ohnehin das größte Modell gewählt werden, dessen Berechnung für die konkrete Anwendung noch praktikabel ist.

Projekt	Startpunkt	Auswahl nach ... Likelihood	Erkennungsrate	bestes Ergebnis
Arab Emirate (Emirate)	85,3 %	53,0 %	85,3 %	85,3 %
Kanada (Adressen)	95,0 %	94,4 %	95,2 %	95,7 %
Deutschland (Adressen)	94,0 %	94,4 %	94,1 %	94,4 %
Deutschland (PLZ)	75,0 %	73,8 %	77,4 %	77,4 %
USA (Adressen)	84,1 %	84,7 %	84,9 %	85,7 %
USA (ZIP)	67,9 %	68,1 %	69,8 %	70,0 %
USA-CEDAR (Städte)	85,7 %	85,7 %	85,7 %	85,7 %
USA-CEDAR (ZIP)	65,5 %	66,0 %	68,3 %	68,3 %
Mittlere Erkennungsrate	81,6 %	77,5 %	82,6 %	82,8 %

Tabelle 13.3: Ergebnis der Allographadaption mit fest vorgegebener Allographenzahl. Im Durchschnitt werden sechs Varianten pro Graphem modelliert.

13.4 Konstante Modellkomplexität

Die bisher vorgestellten Methoden zur Allographbestimmung führen zu sehr großen Modellen, deren Berechnung viel Rechenzeit beansprucht. In konkreten Anwendungen ist Rechenzeit aber ein Faktor, der für die Einsatzfähigkeit des Systems entscheidend ist. Es wird deshalb ein Adaptionsverfahren vorgestellt, das die Modellkomplexität konstant hält, und das bei einer konstanten Gesamtanzahl von Allographen die Modellierung der einzelnen Grapheme optimal aufteilt.

Das allgemeine Vorgehen entspricht den bisherigen Verfahren, bei denen Allographen entweder nur hinzugefügt oder nur entfernt wurden, und das in Abbildung 13.1 skizziert ist. Jetzt werden aber in jeder Iteration Modelle sowohl hinzugefügt als auch entfernt; die Gesamtzahl der Modelle bleibt dabei gleich. In jedem Schritt werden 30 Prozent aller Grapheme geändert. Die Auswahl der Grapheme zur Änderung erfolgt nach einer Kombination der in Abschnitt 13.1.1 vorgestellten Strategien. Das Kriterium für das Hinzufügen eines Allographmodells ist eine niedrige Likelihood des Graphems, für die Entfernung eines Modells ein geringer Abstand der Zustandspfade.

Um zu verhindern, dass nach diesen Kriterien dasselbe Graphem erweitert und gleichzeitig wieder reduziert wird, werden die beiden Kriterien kombiniert. Dies erfolgt durch eine Bewertung der zwei Ranglisten, die für die Modelle nach den beiden Kriterien erstellt werden. Jedes Modell wird nach seiner Position in den Ranglisten bewertet; die Modelle mit der höchsten Bewertung werden anschließend erweitert, die mit der niedrigsten reduziert. Experimente haben ergeben, dass sich die Ergebnisse noch verbessern lassen, wenn auch die Rangliste der Allographhäufigkeiten in die Bewertung mit eingeht.

Für das Experiment wird eine feste Rate von sechs Allographen pro Zeichen vorgegeben. Startpunkt der Iterationen ist die gleichförmige Verteilung der Varianten. Ergebnisse der Erkennungsraten finden sich in Tabelle 13.3.

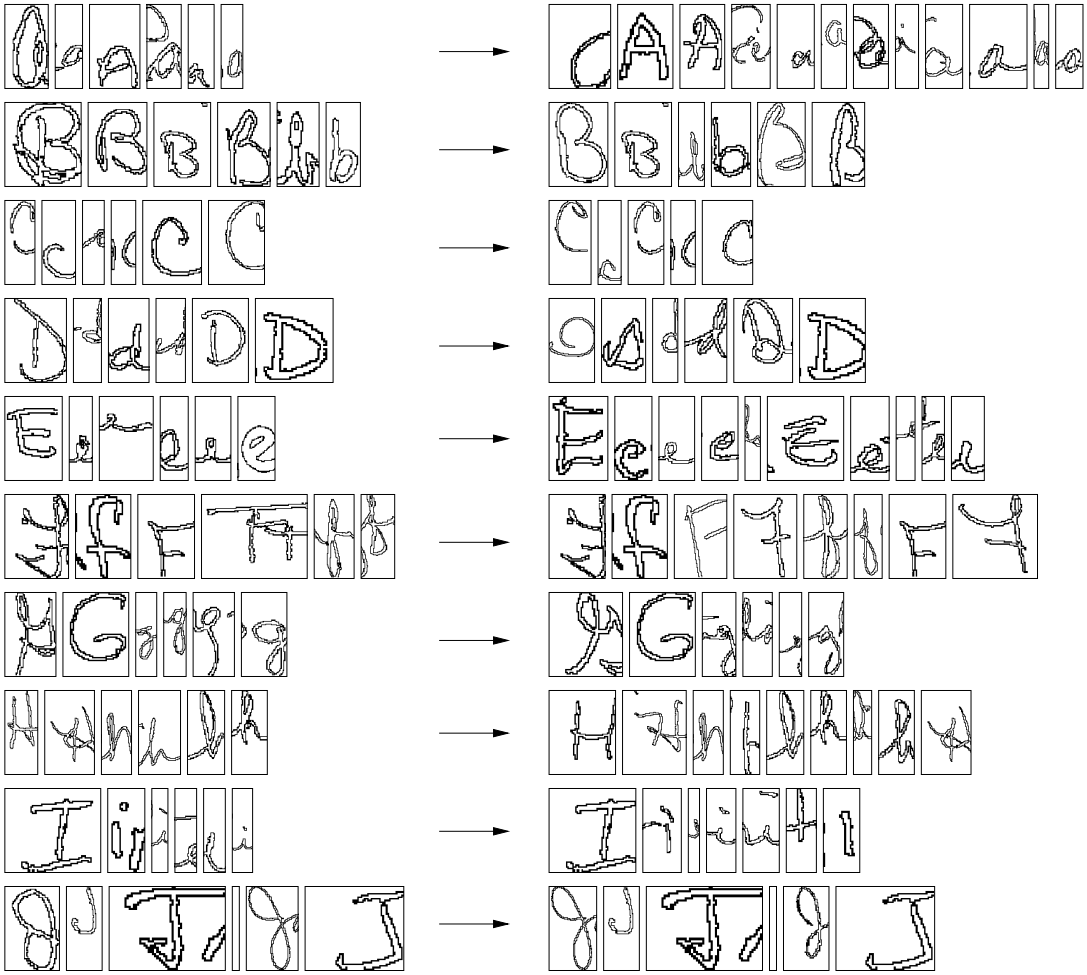

Abbildung 13.13: Hinzufügen und Entfernen von Allographen für US-Adressen. Links ist das Ausgangssystem mit genau sechs Varianten pro Graphem, rechts das Ergebnis nach den Iterationen.

Bei gleicher Gesamtanzahl der Allographen lässt sich die durchschnittliche Erkennungsrate von 81,6 Prozent – bei homogener Modellierung – auf 82,6 Prozent steigern. Damit wird ein Ergebnis erreicht, das fast so gut ist, wie das bisher beste Ergebnis aus Tabelle 13.2 auf Seite 146, bei dem ausgehend von jeweils acht Allographen nach Abstand reduziert wurde. Allerdings werden deutlich weniger Allographen modelliert, was sich direkt in der Rechenzeit niederschlägt. Statt 200 Allographen umfassen die mit der hier vorgestellten Methode erzeugten Schriftmodelle durchschnittlich nur 157 Allographen.

Eine Visualisierung der Ergebnisse für US-Adressdaten befindet sich Abbildung 13.13. Man sieht, welche Modelle sich geändert haben. Die Komplexität von Buchstaben wie „a“ und „e“ hat sich plausibler Weise, wie bei der Analyse der Einzelverfahren, stark erhöht. Seltene und einfache Buchstaben wie das „c“ und die in der Abbildung nicht gezeigten Buchstaben „q“, „x“ und „y“ werden auf drei bis fünf, Ziffern und Sonderzeichen sogar auf bis zu zwei Varianten reduziert.

Wie aufgrund der Analyse der Einzelkriterien zu erwarten war, ist die Modellierung plau-

Projekt	Startpunkt	Auswahl nach ... Likelihood	Erkennungsrate	bestes Ergebnis
Arab Emirate (Emirate)	86.4 %	71.4 %	86.4 %	86.4 %
Kanada (Adressen)	95.4 %	94.3 %	95.5 %	95.6 %
Deutschland (Adressen)	94.5 %	94.4 %	94.7 %	94.7 %
Deutschland (PLZ)	78.3 %	75.5 %	76.0 %	78.3 %
USA (Adressen)	84.6 %	84.1 %	84.8 %	85.1 %
USA (ZIP)	70.7 %	71.4 %	71.4 %	71.9 %
USA-CEDAR (Städte)	85.1 %	86.2 %	86.3 %	86.7 %
USA-CEDAR (ZIP)	68.3 %	71.7 %	71.7 %	71.7 %
Mittlere Erkennungsrate	82,9 %	81,1 %	83,4 %	83,8 %

Tabelle 13.4: Ergebnis der Längennachadaption des besten Systems (Entfernen nach Abstand, Start mit acht Varianten). Es kann eine weitere Verbesserung um 0,5 % erreicht werden.

sibel. Zusammen mit den Verbesserungen der Erkennungsleistung bei gleichbleibendem Rechenaufwand erweist sich das Verfahren damit ohne Einschränkungen als praxistauglich.

13.5 Längennachtraining

Als Abschluss der Experimente wird untersucht, wie sich die beiden in dieser Arbeit vorgestellten Methoden zur Topologiebestimmung zueinander verhalten. Erst durch die Kombination der Allographenclusterung mit der Bestimmung der Modelllänge (Kapitel 11) wird die Topologie des Schriftmodells vollständig automatisch bestimmt.

Es wird getestet, wie sich das System verhält, wenn in Anschluss an die Clusterung der Allographen die Länge der neu generierten Allographmodelle *einmal* neu adaptiert wird. Es wäre auch möglich, Allograph- und Längenadaption mehrfach alternierend durchzuführen. Wegen der sehr hohen Rechenzeiten und dem immer kleiner werdenden Optimierungspotential wird dieses Experiment jedoch nicht durchgeführt. Es wird sozusagen nur die erste Iteration eines alternierenden Prozesses ausgeführt. Die prinzipielle Möglichkeit der Kombination der beiden Verfahren lässt sich so zeigen. Es wird untersucht, wie sich ihr Potential zur Verbesserung des Schriftmodells und damit der Erkennungsleistung des Systems in Kombination summiert.

Für das Experiment wird versucht, die beste der bisher generierten Konfigurationen durch Längenadaption weiter zu optimieren. Startpunkt sind die Schriftmodelle, die durch iteratives Entfernen nach Abstand entstanden sind, ausgehend von einem Modell mit jeweils acht Allographen (vgl. Tabelle 13.2). Die Längenadaption erfolgt wie in Kapitel 11 beschrieben. Die Expansion der Zustandspfade erfolgt durch „Mischen“, die endgültigen Modelle werden aufgrund der Erkennungsrate auf den Trainingsdaten bestimmt.

Die Ergebnisse sind in Tabelle 13.4 dargestellt und sind vielversprechend. Die mittle-

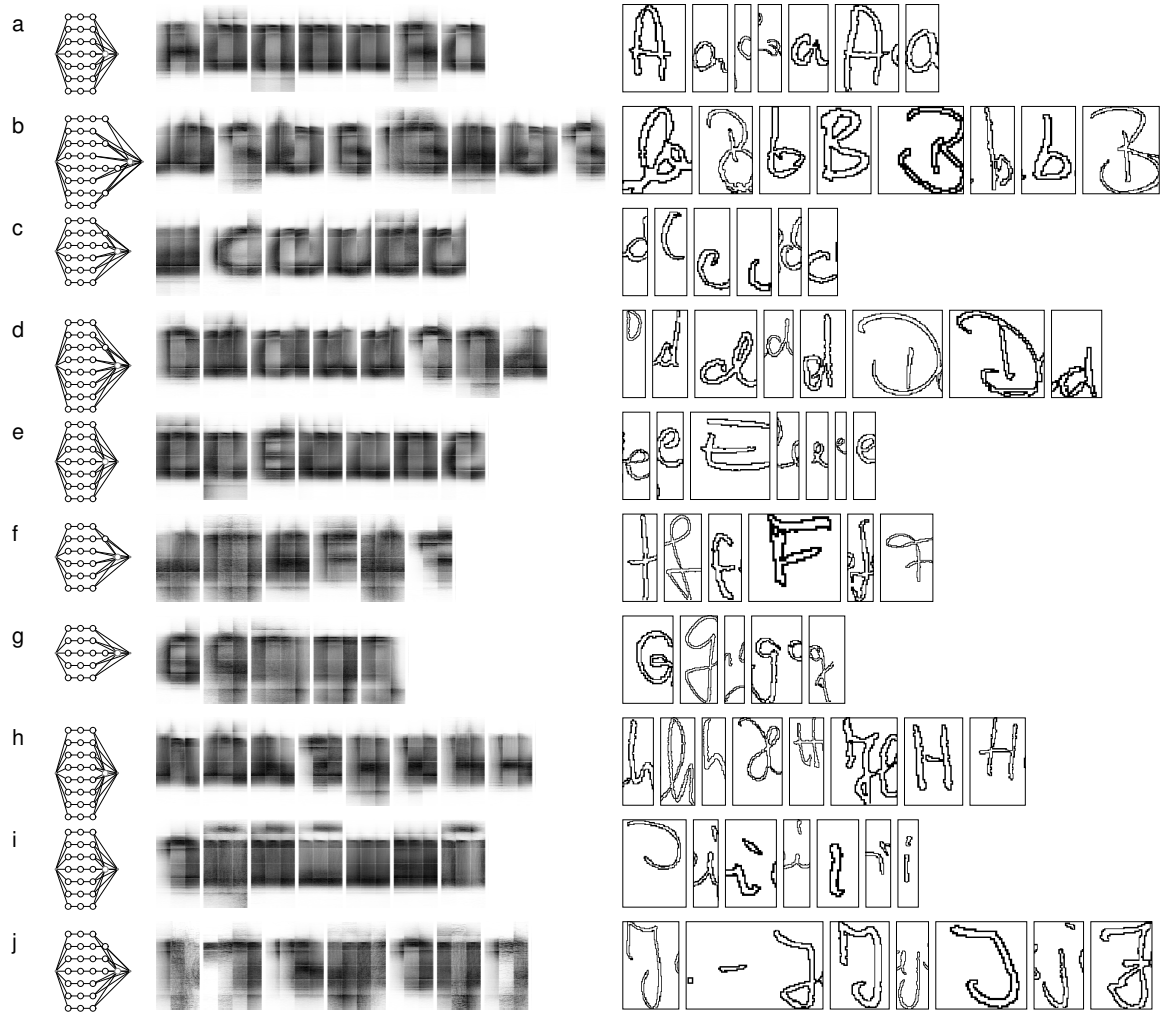

Abbildung 13.14: Ergebnis der Längennachadaption für deutsche Adressen.

re Erkennungsrate, die durch die Allographclusterung ohnehin schon um 7,52 Prozent auf 82,9 Prozent erhöht werden konnte, verbessert sich noch einmal auf 83,4 Prozent, was einer Gesamtsteigerung um 8,12 Prozent entspricht. Wie erwartet leisten die beiden Methoden voneinander unabhängig ihren Beitrag zur Verbesserung von Modellierung und Erkennungsleistung. Es muss aber auch darauf hingewiesen werden, dass sich in einem Fall, beim System deutsche Postleitzahlen, die Erkennungsleistung bei Auswahl nach Erkennungsrate auch verschlechtert. Das zeigt die Grenzen dieses Auswahlkriteriums und der Fähigkeit des Systems zur Generalisierung.

Visualisierungen der endgültigen Schriftmodelle finden sich in den Abbildungen 13.14, 13.15 und 13.16. Die Pfade der gezeigten Modelle werden meist – aber nicht ausschließlich, wie das arabische Beispiel zeigt – verlängert, was die Anpassungsfähigkeit der Modelle erhöht. Für die Interpretation der Resultate gilt das in den jeweiligen Kapiteln Gesagte. Es lässt sich zusammenfassend feststellen, dass die generierten Schriftmodelle plausibel sind.

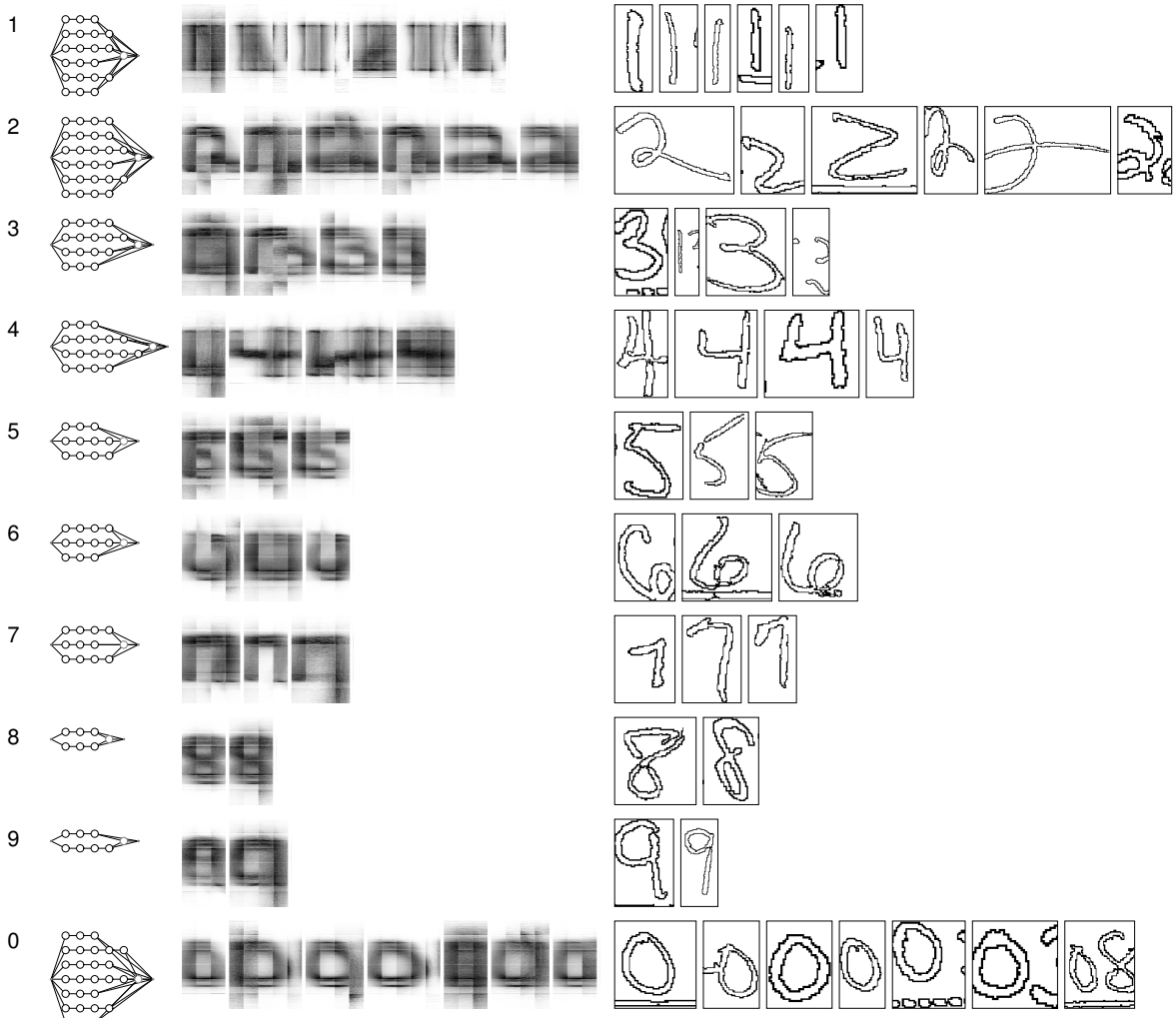

Abbildung 13.15: Ergebnis der Längennachadaption für US ZIP-Codes.

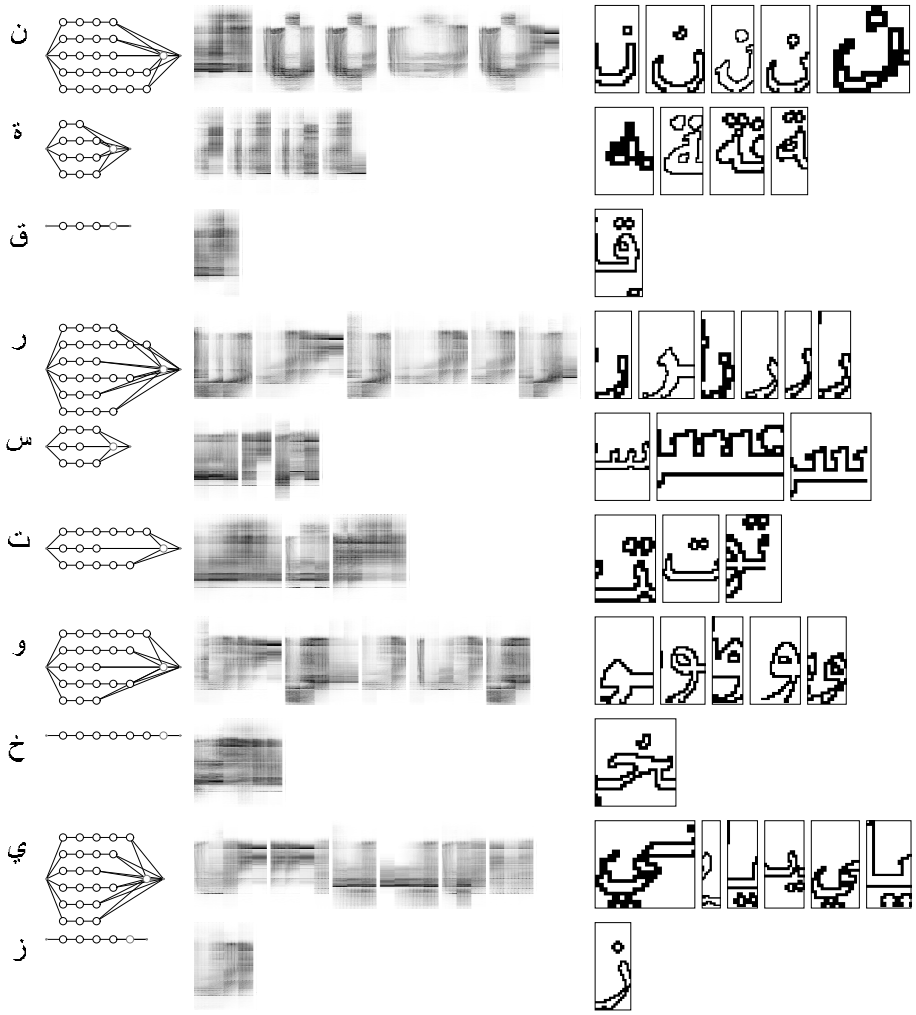

Abbildung 13.16: Ergebnis der Längennachadaption für arabische Emiratsnamen.

Kapitel 14

Zusammenfassung und Ergebnisse

Diese Arbeit behandelt die „Automatische Modellierung gebundener Handschrift in einem HMM-basierten Erkennungssystem“. Die Schriftmodellierung besteht darin, dass in einem existierenden System zur Handschrifterkennung die Struktur (oder Topologie) der Hidden-Markov-Modelle automatisch optimiert wird. Diese Struktur bildet die Grundlage für das Modell der Handschrift, das das untersuchte System zur Erkennung verwendet. Mit ihrer automatischen Bestimmung werden folgende Ziele verfolgt:

- Die Entwicklung eines *realistischen* Schriftmodells: Durch die automatische Analyse von Handschrift-Trainingsdaten wird ein Modell gebildet, das Einblick in die verschiedenen Schreibweisen bietet. Es wird Wissen über die untersuchte Schrift gewonnen. Gerade bei „unbekannten“ Schriften – zum Beispiel Arabisch oder Kyrillisch – ist das in der Praxis sehr wertvoll.

- Die Senkung des Entwicklungsaufwands: Das Wissen über die Struktur der Schrift wird bei der Entwicklung von neuen Erkennungssystemen benötigt. Wenn es automatisch generiert und angewendet wird, können die Kosten für die Erstellung eines neuen Systems beträchtlich reduziert werden.

- Die Erhöhung der Erkennungsleistung: Das Schriftmodell ist Grundlage des Erkennungssystem. Eine gute Modellierung resultiert in einer Erhöhung der Erkennungsleistung des Systems.

14.1 Ausgangspunkt: *Powerscript*

Am Ausgangspunkt der Arbeiten stand das Erkennungssystem *Powerscript*. Es basiert auf Standardtechniken der Erkennung mit Hidden-Markov-Modellen. Die einzelnen Schritte für Training und Erkennung werden im ersten Teil der Arbeit, in den Kapiteln 2 bis 6, detailliert beschrieben. Ausführlich werden Vorverarbeitung und Normierung des Eingabe-Wortbildes, die Berechnung einer Merkmalsfolge, das Baum-Welsh-Training der Modellparameter und

die Erkennung mit dem Viterbi-Algorithmus dargestellt. Das System ist so leistungsfähig, dass es in vielen Anlagen der Postautomatisierung erfolgreich eingesetzt wird.

Die Topologie der Buchstabenmodelle war aber fest vorgegeben und starr. Im Rahmen dieser Arbeit wurden deshalb zunächst die Datenstrukturen und Methoden entwickelt und implementiert, mit denen flexible Modellstrukturen generiert und gehandhabt werden können. Um die neu zu entwickelnden Optimierungsalgorithmen in das Gesamtsystem integrieren zu können, wurde das Trainingssystem vollständig neu implementiert. Zusätzlich wurden zahlreiche Tools für die Visualisierung der internen Modellrepräsentation und die Auswertung der Erkennungsleistung entwickelt.

Auch die neue *variable* Topologie der Buchstaben-HMMs bleibt auf eine bestimmte Grundstruktur eingeschränkt, die an die Aufgabe der Schrifterkennung angepasst ist. Die verschiedenen Schreibvarianten der einzelnen Zeichen werden durch parallele, lineare Pfade aus Modellzuständen repräsentiert. Variabel sind die *Anzahl* und die individuelle *Länge* dieser Zustandspfade.

14.2 Optimierung der Modellstruktur

Der zweite Teil der Arbeit widmet sich der automatischen Bestimmung der optimalen Modellstruktur. Dazu werden in Kapitel 9 zunächst allgemeine Kriterien zur Modellwahl entwickelt. Eine „richtige“ Modellierung zeichnet sich durch einen geeigneten Kompromiss zwischen zwischen notwendiger Komplexität und möglichst großer Einfachheit aus. Das Prinzip ist als *Occam's Razor* bekannt. Es wird ein Bayes'scher Ansatz verwendet, der aber zumeist nicht exakt gelöst werden kann. Mit Näherungen werden vier Modellwahlkriterien hergeleitet: das Akaike Informations-Kriterium, das Bayes'sche Informations-Kriterium, die exakte und die Maximumlösung der Cheeseman-Stutz-Näherung. Für die Anwendung auf die Struktur der konkreten Buchstaben-HMMs wird in Kapitel 12 das Akaike Informations-Kriterium ausgewählt und angepasst. Daneben werden mit einer Abstandsdefinition und der Entropie der Emissionsgewichte weitere Maße für die lokale Bewertung und den Vergleich von Buchstaben-HMMs entwickelt.

In Kapitel 10 werden bekannte Verfahren zur HMM-Topologiebestimmung vorgestellt. Im konkreten Fall liegen durch die Verteilung des Schriftmodells auf einzelne Buchstaben-HMMs und deren vorgegebene Grundstruktur aber besondere Bedingungen vor, so dass spezielle Verfahren zur Strukturoptimierung entwickelt wurden. Zwei verschiedene Algorithmen wurden spezifiziert und implementiert, die voneinander unabhängig die jeweilige Länge (Kapitel 11) oder die Anzahl (Kapitel 13) der Zustandspfade optimieren. Diese Trennung vereinfacht und beschleunigt die Verfahren und erlaubt eine separate Analyse des Optimierungspotentials. Sie ist aber auch durch unterschiedlichen Optimierungskriterien bedingt, da die Bayes'schen Modellwahlkriterien nur auf die Wahl der Allographenanzahl angewendet werden.

Beide Verfahren arbeiten iterativ: Ausgehend von einem vorgegebenen Modell werden mehrfach Buchstaben-HMMs modifiziert, die Parameter neu trainiert und die endgültigen Änderungen ausgewählt. Durch sequentielle Ausführung lassen sich beide Verfahren einfach kombinieren. Es wurde ein vollständiges Adaptions- und Trainingssystem implementiert, das nur durch die Vorgabe von Alphabet und Trainingsdaten automatisch eine funktionsfähige Konfiguration des Erkennungssystems generiert. Umfangreiche Analyseinstrumente wurden entwickelt, um die Ergebnisse der Modellbestimmungen statistisch auszuwerten und zu visualisieren.

14.3 Ergebnisse

Mit den entwickelten und implementierten Verfahren konnten alle der eingangs genannten Ziele erreicht werden:

- Die Ergebnisse der Modellierung sind plausibel. Die Visualisierungen der generierten Schriftmodelle sind intuitiv erklärbar. Gängige Schreibweisen werden identifiziert. Allerdings zeigen sich nicht nur Merkmale der Schrift selber, sondern auch Eigenschaften der Vorverarbeitung der Wortbilder.

- Der Entwicklungsaufwand für neue Erkennungssysteme wurde gesenkt. Zum Beispiel konnte auch ohne Detailwissen über die Schreibweisen in Arabisch ein leistungsfähiges Erkennungssystem generiert werden.

- Die Erkennungsleistung wurde erhöht. Auch in bereits existierenden Projekten konnte das Optimierungsverfahren eingesetzt werden, um die Leistung des Erkennungssystems zu verbessern. Im Durchschnitt stieg die Erkennungsrate um 8,1 Prozent. Beide entwickelten Verfahren – zur Bestimmung der Schreibvarianten und zur Bestimmung der Modelllängen – leisten ihren eigenen Beitrag zur Erreichung dieses Ergebnisses. Auch bei konstanter Gesamtkomplexität des Schriftmodells, also bei gleichbleibendem Rechenaufwand, werden deutliche Verbesserungen der Erkennungsleistung erzielt.

Die theoretisch hergeleiteten Kriterien für die Wahl der richtigen Modellgröße lassen sich auf das Schriftmodell anwenden und korrelieren gut mit gemessenen Erkennungsraten auf unabhängigen Testdaten. Ein praktischer Einsatz für die Modellauswahl erweist sich jedoch als nicht sinnvoll: Die direkte Bewertung der Modelle aufgrund der Erkennungsleistung auf einer Evaluationsstichprobe ist genauso gut handhabbar, hat aber zudem den Vorteil der unmittelbaren Praxisrelevanz. Ohnehin ist aufgrund von Rechenzeitrestriktionen meistens eine maximale Modellkomplexität vorgegeben, die das endgültige Modell direkt bestimmt.

Die Arbeiten werden im praktischen Einsatz in der Postautomatisierung bei Siemens Dematic verwendet. Da die Optimierung der Parameter *offline* erfolgt, können allein durch den Austausch von Konfigurationen auch bestehende Systeme im Einsatz verbessert werden.

Durch die Erweiterungen des Systems sind auch neue Anwendungsmöglichkeiten entstanden. Die Neustrukturierung des Trainingssystem erlaubt eine Nachadaption von Struktur und Parametern im laufenden System. Dieses Vorgehen wird bei Siemens Dematic im Rahmen des öffentlich geförderten BMBF-Projekts „*Adaptive Read*“ verfolgt [Kol02].

Anhang A

Konvergenz des Baum-Welsh-Trainings

Es lässt sich zeigen, dass man mit dem Baum-Welsh-Training tatsächlich ein lokales Maximum im Parameterraum finden kann. Voraussetzung dafür ist die Tatsache, dass mit jeder Trainingsiteration die neuen Parameter $\overline{\lambda}$ die Trainingsdaten besser beschreiben als die des vorigen Schrittes λ, dass also

$$P(O \mid \overline{\lambda}) \geq P(O \mid \lambda). \tag{A.1}$$

Der Beweis dieser Aussage beruht auf zwei mathematischen Hilfssätzen. Der erste ist eine Anwendung der Jensen-Ungleichung für Erwartungswerte, $E(f(X)) \leq f(E(X))$, die für beliebige konkave Funktionen f und Zufallsverteilungen X gilt.

Lemma A.1 (Jensen-Ungleichung) *Gegeben seien die reellen Zahlen $u_q > 0$ und $v_q \geq 0$ mit der Einschränkung $\sum v_q > 0$. Aus der Konkavität der Logarithmus-Funktion folgt dann*

$$\log \frac{\sum_q v_q}{\sum_q u_q} \geq \frac{1}{\sum_q u_q} \cdot \sum_q (u_q \log v_q - u_q \log u_q). \tag{A.2}$$

Man formuliert die zu beweisende Ungleichung (A.1) durch Logarithmieren in eine äquivalente Aussage

$$\log \frac{P(O \mid \overline{\lambda})}{P(O \mid \lambda)} \geq 0 \tag{A.3}$$

um, und assoziiert $u_q = P(O,q \mid \lambda)$ und $v_q = P(O,q \mid \overline{\lambda})$. Durch Anwendung der Eigenschaften $\sum_q u_q = P(O \mid \lambda)$ und $\sum_q v_q = P(O \mid \overline{\lambda})$ und die Definition einer Hilfsfunktion

$$Q(\lambda, \overline{\lambda}) = \sum_q P(O,q \mid \lambda) \cdot \log P(O,q \mid \overline{\lambda}). \tag{A.4}$$

ergibt die Jensen-Ungleichung dann eine weitere Formulierung der zu beweisenden Ungleichung

$$\log \frac{P(O \mid \overline{\lambda})}{P(O \mid \lambda)} \geq \frac{1}{P(O \mid \lambda)} \cdot [Q(\lambda, \overline{\lambda}) - Q(\lambda, \lambda)]. \tag{A.5}$$

Es bleibt also zu zeigen, dass $Q(\lambda,\overline{\lambda}) - Q(\lambda,\lambda)$ positiv ist. Dies ist dann sicher erfüllt, wenn $Q(\lambda,\cdot)$ gerade für $\overline{\lambda}$ sein Maximum erreicht. An dieser Stelle kommt der zweite Hilfssatz ins Spiel.

Lemma A.2 (Maximum) *Wenn $u_i > 0$, für $i = 1,\ldots,N$, dann erreicht, unter der Einschränkung $\sum_i v_i = 1$, die Funktion*

$$F(v) = \sum_i u_i \log v_i \tag{A.6}$$

ihre einziges globales Maximum für

$$v_i = \frac{u_i}{\sum_i u_i}. \tag{A.7}$$

Nach Lemma A.2 ist $Q(\lambda,\overline{\lambda})$ also genau dann maximal, wenn

$$P(O,q \mid \overline{\lambda}) = \frac{P(O,q \mid \lambda)}{P(O \mid \lambda)}. \tag{A.8}$$

Indem man Q in unabhängige Terme umformuliert, die die Übergangs- und Emissionswahrscheinlichkeiten der Parameter $\lambda = (A,B,\pi)$ einzeln darstellen, erhält man für die Maximumbedingung genau die in den vorigen Abschnitten dargestellten Erwartungswerte als Schätzungen für die neuen Parameter. Die Lernregel des Baum-Welsh-Algorithmus ist also gerade so formuliert worden, dass sich die *Likelihood* der Trainingsdaten mit jeder Iteration verbessert und so ein lokales Maximum im Parameterraum erreicht werden kann. Damit stellt das Baum-Welsh-Training eine leistungsfähige Methode dar, das System an eine große Menge von Beispielen optimal anzupassen.

Anhang B

Dirichlet-Verteilung von Parametern

Die Parameter eines HMM mit diskreten Ausgaben lassen sich vollständig als Sammlung von Multinomialverteilungen beschreiben. Für jeden Zustand der einzelnen Buchstaben-HMMs stellt sowohl die Emission der Symbole als auch der Übergang zum nächsten Zustand eine Wahl aus einer gegebenen Menge von n Alternativen dar. Sie lassen sich jeweils durch die Wahrscheinlichkeits-Parameter $\theta = (\theta_1, \ldots, \theta_n)$ darstellen, von denen wegen der Bedingung $\sum_i \theta_i = 1$ nur $n-1$ Parameter frei wählbar sind. Die Parameter θ stehen dabei im folgenden in gleicher Weise für jeden Satz der voneinander unabhängigen Wahrscheinlichkeiten (a_i, b_i, π_i) der Zustände i eines Modells λ.

Wenn nun eine Beobachtung X durch die Anzahl der Ereignisse $c_1, \ldots, c_n$ repräsentiert ist, so erfolgt die Berechnung ihrer Likelihood durch

$$P(X|\theta, M_s) = \prod_{i=1}^{n} \theta_i^{c_i}. \tag{B.1}$$

Die Konjugierte zur Multinomialverteilung ist gerade die Dirichlet-Verteilung

$$P(\theta|M_s) = P_{\text{dirichlet}}(\theta) = \frac{1}{B(\alpha_1, \ldots, \alpha_n)} \prod_{i=1}^{n} \theta_i^{(\alpha_i - 1)} \tag{B.2}$$

mit den Hyperparametern $\alpha_1, \ldots, \alpha_n$. Die Normierungskonstante $B(\alpha_1, \ldots, \alpha_n)$ ist dabei die n-dimensionale Beta-Funktion

$$B(\alpha_1, \ldots, \alpha_n) = \frac{\Gamma(\alpha_1) \cdot \ldots \cdot \Gamma(\alpha_n)}{\Gamma(\alpha_1 + \ldots + \alpha_n)}. \tag{B.3}$$

Die Wahl der Konjugierten hat zwei Vorteile. Der erste besteht in den angenehmen mathematischen Eigenschaften der konjugierten Verteilung: Ist die a-priori-Verteilung die Konjugierte der Likelihood, so ist die a-posteriori-Verteilung ein Mitglied derselben Familie von Verteilungen [DeG02]. Damit ist also auch die a-posteriori-Wahrscheinlichkeit Dirichlet-verteilt

$$P(\theta|X, M_s) = \frac{1}{B(c_1 + \alpha_1, \ldots, c_n + \alpha_n)} \prod_{i=1}^{n} \theta_i^{(c_i + \alpha_i - 1)}. \tag{B.4}$$

Darüber hinaus ist das Integral über das Produkt aus Likelihood und a-priori-Wahrscheinlichkeit geschlossen lösbar [Sto94]:

$$\begin{aligned}\int_{\theta} P(X|\theta,M_s)P(\theta,M_s)d\theta &= \frac{1}{B(\alpha_1,\ldots,\alpha_n)}\int_{\theta}\prod_{i=1}^{n}\theta_i^{(c_i+\alpha_i-1)}d\theta \\ &= \frac{B(c_1+\alpha_1,\ldots,c_n+\alpha_n)}{B(\alpha_1,\ldots,\alpha_n)}.\end{aligned} \tag{B.5}$$

Diese wichtige Eigenschaft erlaubt die Berechnung der Evidenz in Gleichung (9.5).

Der zweite Vorteil besteht in der Interpretierbarkeit der Hyperparameter α als zusätzliche hypothetische Ereignisse. Anhand von Gleichung (B.4) sieht man, dass im EM-Training nach dem MAP-Kriterium die neuen Parameter $\overline{\theta}$ jeder Iteration durch

$$\overline{\theta}_i = \frac{c_i+\alpha_i-1}{\sum_j c_j+\alpha_j-1} \tag{B.6}$$

geschätzt werden können. Für $\alpha_i > 1$ entspricht dies dem Hinzufügen von $\alpha_i - 1$ zusätzlichen Ereignissen zu den Trainingsdaten. Damit lässt sich das MAP-Training durch minimale Anpassungen in das Baum-Welsh-Training integrieren. Bei $\alpha_i = 1 \forall i$ ist die Verteilung der Parameter gleichförmig und die MAP- ist identisch zur ML-Schätzung.

Die α_i entsprechen Konfidenzwerten für die einzelnen Parameter θ_i. Für ein Nachtraining eines Modells mit den schon bekannten Basisparametern $\tilde{\theta}$ bedeutet dies, dass sich durch das Setzen der Hyperparameter nach

$$\frac{\alpha_i}{\sum_j \alpha_j} = \tilde{\theta}_i \tag{B.7}$$

die Information des Basissystems in den Hyperparametern α_i wiederfindet. Durch die Wahl von $\sum_j \alpha_j$ kann die Gewichtung des Basissystems gesteuert werden.

Anhang C

Eigenschaften der geschlossenen CS-Lösung

Um die interessanten Eigenschaften der geschlossenen Lösung der Cheeseman-Stutz-Näherung (9.25) als Modellwahlkriterium zu verdeutlichen, wird für eine kompakte Notation die Hilfsfunktion

$$K(c_1,\ldots,c_n;\alpha) \equiv \frac{B(c_1+\alpha,\ldots,c_n+\alpha)}{B(\alpha,\ldots,\alpha)} \tag{C.1}$$

eingeführt, die die Bewertung eines Modells darstellt. Verschiedene Modellierungen können unterschiedliche viele Variablen n besitzen, die Summe der Beobachtungen $\sum_i c_i$ ist jedoch nur von der Menge der Daten abhängig und somit konstant. Unter diesen Bedingungen können verschiedene Modellkonstellationen anhand von Beispielen verglichen werden.

Die erste Eigenschaft betrifft die Bewertung der Modellkomplexität. Sie ist zunächst über die Anzahl der Variablen n gegeben: Mit zunehmendem n nimmt die Bewertung des Modells ab. Kleinere Modelle werden also bevorzugt. Es lässt sich nachrechnen, dass

$$K(6,6;1) > K(4,4,4;1). \tag{C.2}$$

Die Entropie der Auftretenswahrscheinlichkeiten ist ein Maß für den Informationsgehalt des Modells. Modelle mit niedriger Entropie werden höher bewertet:

$$K(10,2,0;1) > K(4,4,4;1). \tag{C.3}$$

Die beiden Effekte können sich kompensieren, so dass ein komplexes Modell mit niedriger Entropie besser bewertet werden kann als ein einfacheres mit höherer Entropie:

$$K(10,2,0;1) > K(6,6;1). \tag{C.4}$$

Konfigurationen mit einer höheren Anzahl von Parametern werden also dann als geeigneter eingestuft, wenn die Unbestimmtheit der Parameter beim größeren Modell abnimmt, das Modell die Daten also besser beschreiben kann.

Dies gilt auch für den Fall mehrerer Sätze voneinander unabhängiger Parameter, zum Beispiel die Parameter verschiedener Buchstabenmodelle. Für deren Bewertung wird das Produkt der einzelnen Bewertungen herangezogen

$$K(1,1;1)\cdot K(1,1;1) < K(2,2;1) < K(2,0;1)\cdot K(0,2;1). \tag{C.5}$$

Diese Eigenschaft des betrachten Kriteriums lässt es sinnvoll erscheinen, seine Anwendung im Fall von HMMs auf die beobachteten Emissionen zu beschränken, da hier eine scharfe Verteilung der Wahrscheinlichkeiten auf eine geeignete Modellierung schließen lässt. Auch wächst die Anzahl der Parameter mit der Anzahl der Zustände und damit der Komplexität des Modells.

Die zweite interessante Eigenschaft des Kriteriums betrifft den Einfluss der Datenmenge. Liegen viele Trainingsdaten vor, so sinkt der Schätzfehler und eine größere Spezialisierung der Modelle wird möglich. Diese gewünschte Eigenschaft wird mit dem Kriterium erreicht. Es ist

$$\begin{aligned} & K(8,8;1) > K(12,4,0;1), \\ \text{aber} \quad & K(16,16;1) < K(24,8,0;1). \end{aligned} \tag{C.6}$$

Unabhängig von der relativen Aufteilung der Beobachtungen wird eine Erhöhung der Parameteranzahl also bei einer größeren Menge an Daten favorisiert. Häufige Buchstaben werden auf diese Weise genauer modelliert als seltene, und auch die Eingangswahrscheinlichkeiten erhalten bei der Modellauswahl eine Bedeutung, indem sie die Häufigkeit steuern, mit der die verschiedenen Modellpfade gewählt werden.

Einen ähnlichen Einfluss hat der bisher nicht beachtete Parameter α der Dirichlet-Verteilung. Die Vergrößerung von α entspricht in ihrer Wirkung einer Reduktion der Datenmenge. Zum Beispiel gilt

$$\begin{aligned} & K(10,10;1) < K(15,5,0;1), \\ \text{aber} \quad & K(10,10;2) > K(15,5,0;2). \end{aligned} \tag{C.7}$$

Mit α steht also ein Parameter zur Verfügung, mit dem es möglich ist, den Grad der Generalisierung zu steuern.

Literaturverzeichnis

[Aki72] H. AKIMA. Interpolation and smooth curve fitting based on local procedures. *Comm. ACM*, 15(10):914, 1972. 30

[Auf97] C. AUFMUTH. Revealing the hidden Markov recognizer. In *Proc. of the 4th Int. Conf. on Document Analysis and Recognition*, Seiten 460–463, Ulm, Germany, August 1997. IEEE Computer Society Press. 66

[Bah01] C. BAHLMANN und H. BURKHARDT. Measuring HMM similarity with the Bayes probability of error. In *Proc. of the 6th Int. Conf. on Document Analysis and Recognition*, Seiten 406–411, Seattle, WA, September 2001. IEEE Computer Society Press. 125, 126, 127

[Ban00] M. BANGE. Parallele Simulation neuronaler Netze auf Multimedia-Erweiterungen moderner Mikroprozessoren. Diplomarbeit, Universität Ulm, 2000. 76

[Bau67] L. E. BAUM und J. A. EAGON. An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology. *Bulletin of American Mathematical Society*, 73:360–363, 1967. 17

[Bau70] L. E. BAUM, T. PETRIE, G. SOULES und N. WEISS. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *Annals of Mathematical Statistics*, 41:164–171, 1970. 21

[Bis95] C. M. BISHOP. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995. 46, 50

[Bou96] H. BOURLARD und N. MORGAN. Hybrid connectionist models for continuous speech recognition. In *Automatic Speech and Speaker Recognition*, Seiten 259–283. Kluwer Academic Publishers, 1996. 43

[Boz87] H. BOZDOGAN. Model selection and Akaikes's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, September 1987. 92

[Bra96] T. BRANTS. Estimating Markov model structures. In *Proc. of the 4th Int. Conf. on Spoken Language Processing*, Volume 2, Seiten 893–896, Philadelphia, PA, 1996. 105

[Bra98] T. BRANTS. Estimating Hidden Markov Model Topologies. In J. GINZBURG, Z. KHASIDASHVILI, C. VOGEL, J.-J. LÉVY und E. VALLDUVÍ, editors, *The Tbilisi Symposium on Logic, Language and Computation: Selected Papers*, Seiten 163–176. CSLI Publications, Stanford, California, 1998. 104

[Bra01] A. BRAKENSIEK, A. KOSMALA und G. RIGOLL. Comparing adaptation techniques for on-line handwriting recognition. In *Proc. of the 6th Int. Conf. on Document Analysis and Recognition*, Seiten 486–490, Seattle, WA, September 2001. 88

[Bro87] I. N. BRONSTEIN und K. A. SEMENDJAJEW. *Taschenbuch der Mathematik*. Harri Deutsch, 23th edition, 1987. 59

[Cad00] I. V. CADEZ, S. GAFFNEY und P. SMYTH. A general probabilistic framework for clustering individuals and objects. In *Proc. of the ACM 7th International Conference on Knowledge Discovery and Data Mining*, Seiten 140–149, New York, NY, 2000. ACM Press. 99

[Cae93] T. CAESAR, J. GLOGER und E. MANDLER. Preprocessing and feature extraction for a handwriting recognition system. In *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition*, Seiten 408–411, Tsukuba Science City, Japan, Oktober 1993. IEEE Computer Society Press. 25

[Che88] P. CHEESEMAN, J. KELLY, M. SELF, J. STUTZ, W. TAYLOR und D. FREEMAN. AutoClass: A Bayesian classification system. In *Proc. of the Fifth Int. Conference on Machine Learning*, Seiten 54–64, Ann Arbor, MI, 1988. Morgan Kaufmann Publishers. 89

[Che96] P. CHEESEMAN und J. STUTZ. Bayesian classification (AutoClass): Theory and results. In U. M. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH und R. UTHURUSAMY, editors, *Advances in Knowledge Discovery and Data Mining*, Seiten 153–180. MIT Press, 1996. 92

[Chi97] D. M. CHICKERING und D. HECKERMAN. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997. 92

[Cor90] T. H. CORMEN, C. E. LEISERSON und R. L. RIVEST. *Introduction to Algorithms*. MIT Press, Cambridge, Mass., London, England, 1990. 17

[DeG02] M. H. DEGROOT und M. J. SCHERVISH. *Probabiltiy and Statistics*. Addison-Wesley, 3rd edition, 2002. 165

[Dem77] A. P. DEMPSTER, N. M. LAIRD und D. B. RUBIN. Maximum Likelihood from Incomplete Data via the EM Algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39:1–38, 1977. 21

[Dol98] J. G. A. DOLFING. *Handwriting Recognition and Verfication: A Hidden Markov Approach*. Dissertation, Eindhoven University of Technology, 1998. 7

[Fal95] M. FALKHAUSEN, H. REININGER und D. WOLF. Calculation of distance measures between hidden Markov models. In *Proceedings of Eurospeech*, Seiten 1487–1490, Madrid, Spain, 1995. 123

[For00] M. R. FORSTER. Key concepts in model selection: Performance and generalizability. *Journal of Mathematical Psychology*, 44(1):205–231, 2000. 86, 87, 88, 92

[Fuk90] K. FUKUNAGA. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1990. 59

[Gau94] J. GAUVIN und C.-H. LEE. Maximum a posteriori estimation for multivariate gaussian mixture observations of Markov chains. *IEEE Trans. on Speech and Audio Processing*, 2(2):291–298, April 1994. 88

[Gha01] Z. GHAHRAMANI. An introduction to hidden Markov models and Bayesian networks. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42, 2001. 13, 88, 89, 90

[Gib97] P. GIBBS und H. SUGIHARA. What is Occams's Razor? http://www.weburbia.com/physics/occam.html, 1997. 86

[Glo97] J. M. GLOGER, A. KALTENMEIER, E. MANDLER und L. ANDREWS. Reject management in a handwriting recognition system. In *Proc. of the 4th Int. Conf. on Document Analysis and Recognition*, Seiten 556–559, Ulm, Germany, August 1997. 68

[Gui98] D. GUILLEVIC und C. Y. SUEN. Recognition of legal amounts on bank cheques. *Pattern Analysis and Applications*, 1(1):28–41, 1998. 9

[Hae92] R. HAEB-UMBACH und H. NEY. Linear discriminant analysis for improved large vocabulary speech recognition. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 2, Seiten 13–16, San Francisco, 1992. 58

[Ham98] J. HAMAKER, A. GANAPATHIRAJU und J. PICONE. Information theoretic approaches to model selection. In *Proc. of the 5th Int. Conf. on Spoken Language Processing*, Sydney, Australia, 1998. 105

[Ham99] J. HAMAKER und J. ZHAO. Bayesian information criterion for automatic model selection. final report, ISIP, Mississippi State University, 1999. 105

[Hec95] D. HECKERMAN. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, März 1995. 91

[Hon01] A. HONKELA. Nonlinear switching state-space models. Master's thesis, Helsinki University of Technology, 2001. 90

[Hua89] X. HUANG und M. JACK. Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language*, 3(3):239–251, 1989. 42

[Hua01] X. HUANG, A. ACERO und H.-W. HON. *Spoken Language Processing*. Prentice-Hall, 2001. 124

[Hul94] J. J. HULL. A database for handwriting recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, Mai 1994. 70

[Joh92] R. A. JOHNSON und D. W. WICHERN. *Applied Multivariate Statistical Analysis*. Prentice-Hall International, 1992. 46

[Jua85] B. JUANG und L. RABINER. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, 64(2):391– 408, Februar 1985. 123

[Jua90] B. JUANG und L. RABINER. The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(9S):1639–1641, 1990. 98

[Kal91] A. KALTENMEIER. *Modellbasierte Worterkennung in Spracherkennungssystemen für großen Wortschatz*. Dissertation, Universität Ulm, 1991. 11

[Kal93a] A. KALTENMEIER. Hidden Markov Modelle - Ein einheitlicher Ansatz für die Erkennung geschriebener und gebundener Sprache. Technical Report F3-93-025, Daimler Benz, November 1993. 40

[Kal93b] A. KALTENMEIER, T. CAESAR, J. GLOGER und E. MANDLER. Sophisticated topology of hidden Markov models for cursive script recognition. In *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition*, Seiten 139–142, Tsukuba Science City, Japan, Oktober 1993. 40

[Kal93c] A. KALTENMEIER, F. CLASS, P. REGEL-BRIETZMANN, T. CAESAR, J. GLOGER und E. MANDLER. Hidden Markov models — a unified approach to recognition of spoken and written language. In *Mustererkennung 93, 15. DAGM-Symposium Lübeck*, Berlin, 1993. Springer-Verlag. 11

[Knu98] D. E. KNUTH. *The Art of Computer Programming, Volume 3, Sorting and Searching*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1998. 62

[Kol02] P. KOLLER. Adaptive Read: Dokumentenerkennung lernt das Lernen. *Computer Zeitung*, 9. September 2002. 162

[Kwo88] P. C. K. KWOK. A thinning algorithm by contour generation. *Communications of the ACM*, 31(11):1311–1324, 1988. 31

[Kwo98] S. KWONG, Q. H. HE, K. F. MAN und K. S. TANG. A maximum model distance approach for HMM-based speech recognition. *Pattern Recognition*, 31(3):219–229, 1998. 121

[Lee90] K.-F. LEE, H.-W. HON und R. REDDY. An overview of the sphinx speech recognition system. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(1):35–45, Januar 1990. 43

[Lee00] J. J. LEE, J. KIM und J. H. KIM. Data driven design of HMM topology for online handwriting recognition. In Schomaker and Vuurpijl [Sch00a], Seiten 239–248. 99

[Li00a] C. LI. *A Bayesian Approach to Temporal Data Clustering using Hidden Markov Models*. Dissertation, Vanderbilt University, 2000. 85, 90, 93, 98, 104, 107, 109, 133, 138

[Li00b] C. LI und G. BISWAS. A Bayesian approach to temporal data clustering using the hidden Markov model methodology. In *Int. Conference on Machine Learning*, Seiten 543–550, Stanford, CA, Juni 2000. 107

[Lin80] Y. LINDE, A. BUZO und R. M. GRAY. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28(1):84–95, Januar 1980. 45, 98

[Lyn99] R. B. LYNGSØ, C. N. S. PEDERSEN und H. NIELSEN. Metrics and similarity measures for hidden Markov models. In *Proceedings of ISMB*, Seiten 178–186, 1999. 123

[Mac92] D. J. C. MACKAY. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992. 88, 90, 95

[Mac97] D. J. C. MACKAY. Ensemble learning for hidden markov models. Technical Report CB3 0HE-UK, Cavendish Laboratory, University of Cambridge, 1997. 90

[Mac98] D. J. C. MACKAY. Choice of basis for Laplace approximation. *Machine Learning*, 33(1), Oktober 1998. 90

[Mak96] B. MAK und E. BARNARD. Phone clustering using the bhattacharyya distance. In *Proc. of the 4th Int. Conf. on Spoken Language Processing*, Seiten 2005–2008, Philadelphia, 1996. 105, 126

[Man90] E. MANDLER und M. F. OBERLÄNDER. Ein single-pass Algorithmus für die schnelle Konturkodierung von Binärbildern. In *Proceedings of the 12th DAGM Symposium*. Springer Verlag, 1990. 27

[Neu98] C. NEUKIRCHEN, D. WILLETT und G. RIGOLL. Soft state-tying for HMM-based speech recognition. In *Proc. of the 5th Int. Conf. on Spoken Language Processing*, Sydney, Australia, 1998. 88

[Ost97] M. OSTENDORF und H. SINGER. HMM topology design using maximum likelihood successive state splitting. *Computer Speech and Language*, 11(1):17–41, 1997. 104

[Per00] M. P. PERRONE und S. D. CONNELL. K-means clustering for hidden Markov models. In Schomaker and Vuurpijl [Sch00a], Seiten 229–238. 98

[Pur94] H. PURNHAGEN. N-Best Search Methods Applied to Speech Recognition. Master's thesis, Department of Telecommunications, Norwegian Institut of Technology, 1994. 64

[Rab89a] L. R. RABINER, C. H. LEE, B. H. JUANG und J. G. WILPON. HMM clustering for connected word recognition. In *Proc. of the 14th Int. Conf. on Acoustics, Speech and Signal Processing*, Seiten 405–408, 1989. 80, 98

[Rab89b] L. R. RABINER. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, Februar 1989. 13, 15

[Ras01] C. E. RASMUSSEN und Z. GHAHRAMANI. Occams's razor. In T. LEEN, T. DIETTERICH und V. TRESP, editors, *Advances in Neural Information Systems*, Volume 13. MIT Press, 2001. 86

[Rob96] T. ROBINSON, M. HOCHBERG und S. RENALS. The use of recurrent neural networks in continuous speech recognition. In *Automatic Speech and Speaker Recognition*, Seiten 233–258. Kluwer Academic Publishers, 1996. 43

[Rot00] J. ROTTLAND und G. RIGOLL. Tied posteriors: An approach for effective introduction of context dependency in hybrid nn/hmm lvcsr. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Istanbul, 2000. 21, 43

[Sak78] H. SAKOE und S. CHIBA. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978. 13, 124

[Sch77] J. SCHÜRMANN. *Polynomklassifikatoren für die Zeichenerkennung*. Oldenbourg Verlag, München, Wien, 1977. 21

[Sch78] G. SCHWARZ. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. 91

[Sch96] J. SCHÜRMANN. *Pattern Classification: A Unified View of Statistical and Neural Approaches*. Wiley-Interscience, 1996. 43

[Sch00a] L. R. B. SCHOMAKER und L. G. VUURPIJL, editors. *7th Int. Workshop on Frontiers in Handwriting Recognition*, Nijmegen, The Netherlands, September 2000. 173, 174

[Sch00b] M. SCHÜSSLER. *Automatische Optimierung der Parameter eines Handschriftlesesystems*. Dissertation, Universität Erlangen–Nürnberg, 2000. 101, 110

[Sch03a] M.-P. SCHAMBACH. Determination of the number of writing variants with an HMM based cursive word recognition system. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition*, Edinburgh, Scotland, August 2003. 138

[Sch03b] M.-P. SCHAMBACH. Model length adaptation of an HMM based cursive word recognition system. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition*, Edinburgh, Scotland, August 2003. 114

[Sey99] K. SEYMORE, A. MCCALLUM und R. ROSENFELD. Learning hidden Markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999. 98

[Sin96] H. SINGER und M. OSTENDORF. Maximum likelihood successive state splitting. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 2, Seiten 601–604, Atlanta, GA, 1996. 104

[Smy97] P. SMYTH. Clustering sequences with hidden Markov models. In M. C. MOZER, M. I. JORDAN und T. PETSCHE, editors, *Advances in Neural Information Processing 9*, Seiten 648–654. MIT Press, 1997. 99

[Sri93] S. N. SRIHARI, V. GOVINDARAJU und A. SHEKHAWAT. Interpretation of handwritten addresses in US mailstream. In *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition*, Seiten 291–294, Tsukuba Science City, Japan, Oktober 1993. IEEE Computer Society Press. 10

[Sri97] S. N. SRIHARI und E. J. KUEBERT. Integration of hand-written address interpretation technology into the united states postal service remote computer reader system. In *Proc. of the 4th Int. Conf. on Document Analysis and Recognition*, Seiten 892–896, Ulm, Germany, August 1997. IEEE Computer Society Press. 10

[Ste99] T. STEINHERZ, E. RIVLIN und N. INTRATOR. Offline cursive script word recognition — a survey. *International Journal on Document Analysis and Recognition*, 2(2/3):90–110, 1999. 7, 80

[Sto93] A. STOLCKE und S. OMOHUNDRO. Hidden Markov model induction by Bayesian model merging. In S. J. HANSON, J. D. COWAN und C. L. GILES, editors, *Advances in Neural Information Processing Systems*, Volume 5, Seiten 11–18. Morgan Kaufmann, San Mateo, CA, 1993. 88

[Sto94] A. STOLCKE und S. M. OMOHUNDRO. Best-first model merging for hidden Markov model induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley, CA, 1994. 85, 87, 94, 104, 135, 138, 166

[Tak92] J. TAKAMI und S. SAGAYAMA. A successive state splitting algorithm for efficient allophone modelling. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seiten I.573–576, 1992. 103

[Tak97] T. TAKARA, K. HIGA und I. NAGAYAMA. Isolated word recongnition using the HMM structure selected by the genetic algorithm. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seiten 967–970, Munich, 1997. 101

[Tap90] C. C. TAPPERT, C. Y. SUEN und T. WAKAHARA. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, August 1990. 7

[Tho86] M. G. THOMASON und E. GRANUM. Dynamic programming inference of Markov networks from finite sets of sample strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):491–501, Juli 1986. 105

[Ueb97] W. UEBEL, L. ANDREWS, M. DEHESA, T. LANGE, D. NEUN und M.-P. SCHAMBACH. Dokumentation zu Powerscript (HMR-Verfahren). Interner Bericht, Siemens ElectroCom, Juli 1997. 25

[Vas96] R. C. VASKO, A. EL-JAROUDI und J. R. BOSTONG. An algorithm to determine hidden Markov model topology. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1996. 100, 101

[Vit67] A. J. VITERBI. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, 1967. 19

[Wal84] K. WALL und P.-E. E. DANIELSSON. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision, Graphics, and Image Processing*, 28(2):220–227, 1984. 27

[Wil97a] D. WILLETT, C. NEUKIRCHEN und J. ROTTLAND. Dictionary-based discriminative HMM parameter estimation for continuous speech recognition system. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seiten 1515–1518, Munich, 1997. 136

[Wil97b] D. WILLETT und G. RIGOLL. A New Approach to Generalized Mixture Tying for Continous HMM-Based Speech Recognition. In *5th European Conference on Speech Communication and Technology*, Seiten 1175–1178, Rhodes, Greece, 1997. 43

[Wol97] B. WOLTERMANN. Schreiblinienfindung auf Kursivschriftwörtern durch Bestimmung der Einhüllenden. Praktikumsbericht, DaimlerBenz Forschungszentrum, Ulm, März 1997. 30