On-line Erkennung kursiver Handschrift bei großen Vokabularen

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik

der Universität Karlsruhe (Technische Hochschule)

genehmigte

Dissertation

von

Stefan Manke

aus Freiburg im Breisgau

Tag der mündlichen Prüfung:	13. Februar 1998
Erster Gutachter:	Prof. Dr. Alexander Waibel
Zweiter Gutachter:	Prof. Dr. Walter F. Tichy

Berichte aus der Informatik

Stefan Manke

On-line Erkennung kursiver Handschrift bei großen Vokabularen

Shaker Verlag
Aachen 1998

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Manke, Stefan:
On-line Erkennung kursiver Handschrift bei grossen Vokabularen /
Stefan Manke. - Als Ms. gedr. -
Aachen : Shaker, 1998
(Berichte aus der Informatik)
Zugl.: Karlsruhe, Univ., Diss., 1998
ISBN 3-8265-3622-3

Als Manuskript gedruckt. Printed in Germany.

ISBN 3-8265-3622-3
ISSN 0945-0807

Shaker Verlag GmbH • Postfach 1290 • 52013 Aachen
Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9
Internet: www.shaker.de • eMail: info@shaker.de

Kurzfassung

Mit der vorliegenden Arbeit wird ein schreiberunabhängiges System zur On-line Erkennung von Einzelzeichen, Wörtern und Sätzen vorgestellt, das kontinuierliche Handschrift unabhängig vom Schreibstil erkennen kann. Die Erkennung erfolgt dabei selbst bei großen Vokabularen mit mehr als 20 000 Wörtern in Echtzeit. Die Erkennungsleistung dieses Handschrifterkenners wird auf unterschiedlichen Datenbasen mit Handschriftproben von mehr als 500 Schreibern evaluiert. Sowohl bei der Erkennung von Einzelzeichen als auch insbesondere von Wörtern und Sätzen werden hervorragende Ergebnisse erzielt. Mit 93.4% Erkennungsrate für Einzelwörter mit einem 20 000 Wörter umfassenden Vokabular wird der beste konkurrierende Erkenner um mehr als 4% übertroffen. Bei der Satzerkennung liegt dieses System mit 86.6% für dasselbe Vokabular mehr als 20% über dem bisher besten vergleichbaren Ansatz.

Erreicht wird diese Erkennungsleistung durch neue oder erweiterte Methoden der Vorverarbeitung und eine erstmals für die Handschrifterkennung adaptierte neuronale Architektur, das Multi-State Time Delay Neural Network, das eine Modellierung einzelner Zeichen durch eine Folge von Zuständen erlaubt. Syntaktisches Wissen über die zugrundeliegende Sprache in Form von Vokabularen und Sprachmodellen wird direkt in diese Architektur integriert. Hierzu wird eine Repräsentation des Vokabulars als Baumstruktur gewählt, die in der Handschrifterkennung bislang nur für rein auf Hidden Markov Modellen basierende Ansätze eingesetzt wurde. Diese Baumstruktur ermöglicht den Einsatz von effektiven Pruning-Methoden, um den Suchraum bei großen Vokabularen zu beschränken. Neben dieser Baumsuche werden in dieser Arbeit weitere Verfahren der Integration von Vokabularen in den Erkennungsprozeß vorgestellt und evaluiert.

Die Vorverarbeitung der handschriftlichen Eingabe berücksichtigt alle wesentlichen im Eingabesignal auftretenden Effekte, die die Erkennung erschweren. Unterschiedliche Normalisierungsmethoden werden jeweils anhand der Datenbasen evaluiert. Die hier für die Erkennung gewählte Repräsentation der Eingabe in Form einer zeitlich geordneten Sequenz von Merkmalsvektoren bewahrt den dynamischen Charakter des ursprünglichen Signals. Über die in anderen Systemen übliche Extraktion zeitlich lokaler Merkmale hinausgehend, werden hier zusätzlich globale, von der Zeit unabhängige Merkmale bei der Erkennung berücksichtigt.

Danksagung

Diese Arbeit ist in mancherlei Hinsicht nicht das Ergebnis einer einzelnen Person, sondern es haben eine Vielzahl von Freunden und Kollegen direkt oder indirekt dazu beigetragen. Diesen möchte ich hiermit danken.

Mein erster Dank gilt Prof. Alex Waibel, der diese Arbeit während der gesamten Dauer betreut und unterstützt hat. Die technische Ausstattung, die mir an seinem Lehrstuhl an der Universität Karlsruhe zur Verfügung stand, hätte nicht besser sein können. Er ließ es mir in dieser Hinsicht an nichts fehlen. Durch ihn habe ich gelernt, daß man auch bei der Forschung die Praxisrelevanz der Arbeit nicht aus den Augen verlieren sollte. Bei den unzähligen Demonstrationen und Präsentationen war ich immer wieder gezwungen, meine Ideen und Ergebnisse in eine vorzeigbare und auch für Laien verständliche Form zu bringen. Dankbar bin ich Prof. Waibel auch für seine Unterstützung, wenn es darum ging, die Ergebnisse auf internationalen Konferenzen einem größeren Publikum zu präsentieren.

Ich bedanke mich bei Prof. Walter F. Tichy für die Übernahme des Korreferats und insbesondere auch dafür, daß er sich so kurzfristig die Zeit für die Begutachtung genommen hat und ich so meinen engen Zeitplan einhalten konnte. Die bereits während des Studiums durch Prof. Tichy vermittelten Konzepte der Softwaretechnik haben mir auch bei der Durchführung dieser Arbeit geholfen, den Überblick über ungezählte Zeilen Code und ebenso viele Programmversionen zu behalten, und mir die Arbeit durch den Einsatz mancher nützlicher Werkzeuge erleichtert.

Ein besonderer Dank geht an Michael Finke, der durch viele Gespräche (begleitet durch zunehmend sicherer werdende Würfe mit dem Basketball) zu einigen Ideen dieser Arbeit beigetragen hat. Die Zeit, in der wir uns das Büro teilten, gehörte mit zu der produktivsten Phase während der Entstehung dieser Arbeit. Danken möchte ich ihm auch für das Korrekturlesen dieser Arbeit. Diesbezüglich geht auch herzlicher Dank an Hermann Hild, der sich ebenfalls kurzfristig die Zeit für wertvolle Kommentare genommen hat. Ganz wesentlich zu dieser Arbeit hat Michael Philippsen quasi in letzter Minute durch seine zahlreichen Anmerkungen beigetragen.

Meinen Eltern Heide und Klaus Manke danke ich dafür, daß sie das Vertrauen in mich gesetzt haben, daß ich das Informatik-Studium erfolgreich beenden würde, und mir ein sorgenfreies Studium ermöglicht haben. Mit der größte Dank geht an Tatjana Rollnik, die die meisten Höhen und Tiefen des Studiums und der Promotion mit mir geteilt hat und viele meiner daraus resultierenden Launen durchgestanden hat. Aus den meisten Tiefen hat sie mich wieder herausgeholt und mich vor manch unbedachter Entscheidung bewahrt. Meine Eltern und Tatjana ist diese Arbeit gewidmet.

Daß das Studium und die anschließenden Jahre dieser Arbeit zu einer sehr lustigen und geselligen Zeit wurden, verdanke ich den gemeinsamen Gamble-Abenden und Ausflügen mit Michael (der buchstäblich seit meiner ersten Stunde in Karlsruhe mein Wegbegleiter war), Bärbel, Annika, Ludwig, Sylvia, Burkhard, Ruth, Stephan, Ingrid, Markus, Paul, Karen, Uli, Susi und Silke. Bestimmt habe ich jemanden in dieser Aufzählung vergessen, jedoch sei auch dieser hier bedankt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Im Vergleich zur sog. OCR (**O**ptical **C**haracter **R**ecognition), d.h. der optischen Zeichenerkennung, die eines der ältesten und mit am intensivsten studierten Teilgebiete der Mustererkennung darstellt, wurde der maschinellen On-line Erkennung kursiver Handschrift bis vor wenigen Jahren wenig Aufmerksamkeit gewidmet. Erste Experimente auf diesem Gebiet wurden zwar bereits in den sechziger Jahren durchgeführt [Ear62], jedoch mangels adäquater technischer Voraussetzungen bald wieder aufgegeben. Seit Anfang der neunziger Jahre ist jedoch sowohl seitens der Forschung als auch Industrie ein wieder stark wachsendes Interesse an der On-line Handschrifterkennung zu beobachten. Angetrieben wird dieses zunehmende Interesse zum einen durch die mittlerweile verbesserten technischen Möglichkeiten, z.B. in Form von leistungsfähigen, portablen Computern (Laptops, Notepads und Palmtops) und hochauflösenden Digitalisiertabletts oder drucksensitiven Bildschirmen, auf denen sich mit schnurlosen Stiften ein natürlicheres Schreibgefühl erreichen läßt, als dies früher der Fall war [TSW90]. Zum anderen wurde auf der Suche nach ergonomischeren Mensch-Maschine-Schnittstellen der Stift als Eingabemedium wiederentdeckt.

Stiftbasierte Benutzerschnittstellen und Computer haben in einer Reihe von Situationen Vorteile gegenüber konventionellen tastatur- und mausbasierten Systemen. Da stiftbasierte Schnittstellen eine effiziente einhändige Bedienung des Computers ermöglichen, sind sie insbesondere für den mobilen Einsatz und für Bereiche, in denen der Computer stehend bedient werden muß oder nur wenig Platz zur Verfügung steht, von Vorteil. Außerdem kann eine solche Schnittstelle denjenigen, die bislang noch keinerlei Erfahrung mit Computern haben, die Scheu vor diesen Geräten nehmen, da sie ein wesentlich natürlicheres Aussehen und Bediengefühl als tastaturbasierte Systeme bieten.

Einen wichtigen Beitrag leistet der Stift auch als Eingabemedium in sog. *multimodalen* Benutzerschnittstellen [WVDM96, WSVY97, FM97]. Multimodale Systeme erlauben dem Benutzer nicht nur die bisher übliche Steuerung und Eingabe von Daten über Tastatur und Maus, sondern können darüber hinaus z.B. auch sprachliche

und handschriftliche Eingaben erkennen und verarbeiten. Für jede durchzuführende Aufgabe kann der Benutzer jeweils die ihm für diese Aufgabe am effizientesten erscheinende Eingabemodalität wählen. Für die Eingabe großer Textmengen bietet sich die sprachliche Eingabe an, während die Überarbeitung bestehender Texte (z.B. Korrektur, Löschen oder Einfügen einzelner Wörter) oder Eingabe kurzer Textfragmente (z.B. Notizen im Kalender, Einträge in Tabellenkalkulationen) effizienter handschriftlich erfolgen. Für die Erstellung von Programmen und andere Aufgaben stehen auch weiterhin Tastatur und Maus zur Verfügung.

Die Qualität stiftbasierter Systeme wird im wesentlichen durch folgende Faktoren bestimmt: die Systemergonomie, die Gestaltung der Benutzeroberfläche und die verwendete Technik zur Handschrifterkennung [Sch94]. Mittlerweile werden unterschiedliche stiftbasierte Systeme, z.B. unter der Bezeichnung „**P**ersönlicher **D**igitaler **A**ssistent" (PDA), auf dem Markt angeboten [Mey94]. Die Benutzer dieser Systeme sind jedoch oftmals über die vorhandenen Einschränkungen der Handschrifterkennung enttäuscht und sehen sie in den wenigsten Fällen wirklich als nützliche Instrumente an. Entweder sind die Erkennungsraten zu gering, oder aber es müssen erhebliche Einschränkungen, z.B. zu kleine Zeichensätze und Vokabulare, in Kauf genommen werden, die ein natürliches Schreiben verhindern. Eine zuverlässige und robuste On-line Handschrifterkennung mit hohen Erkennungsraten auch bei großen Zeichensätzen und Vokabularen ist also Voraussetzung für den Erfolg und die Akzeptanz stiftbasierter Benutzerschnittstellen.

1.2 Handschrifterkennung

Die On-line Handschrifterkennung ist lediglich eines der vielfältigen Teilgebiete der maschinellen Schrifterkennung. Die Unterteilung der Schrifterkennung in die einzelnen Forschungsbereiche wird in der entsprechenden Literatur nicht eindeutig gehandhabt, erfolgt aber häufig nach der Art der zu verarbeitenden Eingabe, d.h. des zu erkennenden Textes. Abbildung 1.1 zeigt eine mögliche Einteilung, wie sie insbesondere zur Einordnung des Themas dieser Arbeit sinnvoll erscheint.[1]

Hier werden zunächst die beiden Teilbereiche *Off-line* und *On-line Erkennung* unterschieden. Die Off-line Erkennung verarbeitet typischerweise bereits in Papierform existierenden Text, der mittels optischer Abtastung als Grauwert- oder Schwarzweißbild bestimmter Auflösung in den Rechner übertragen wird, um anschließend verarbeitet und erkannt zu werden. Das ursprüngliche Schreiben des Textes liegt dabei oftmals bereits längere Zeit vor dem Erkennungsprozeß (Abbildung 1.2b). Im Gegensatz dazu findet die On-line Erkennung während des Schreibvorgangs oder unmittelbar danach statt (Abbildung 1.2a). Voraussetzung hierfür ist das Schreiben auf einem speziellen Digitalisiertablett oder drucksensitiven Bildschirm, auf dem der Schriftzug während des Schreibens in Form einer zeitlichen Folge von Koordinaten aufgenommen

[1] Andere Einteilungen fassen z.B. den gesamten Bereich der Off-line Erkennung unter dem Begriff OCR zusammen.

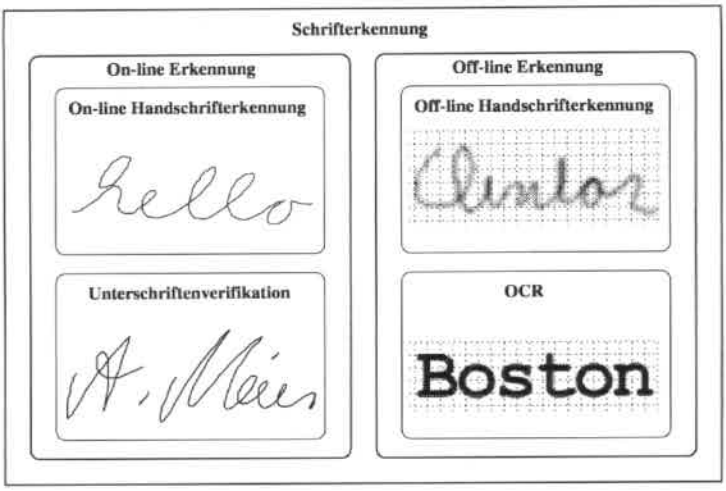


Abbildung 1.1: Teilgebiete der Schrifterkennung

werden kann. Die zur Erkennung notwendigen Daten befinden sich damit unmittelbar im Rechner und können sofort weiterverarbeitet werden.

Innerhalb der Bereiche Off-line und On-line Erkennung können wiederum einzelne Teilbereiche unterschieden werden, wobei Abbildung 1.1 jeweils nur die beiden wichtigsten darstellt. Ältester Teilbereich der Off-line Erkennung ist die *Optical Character Recognition*, welche sich mit der Erkennung maschinell produzierten, d.h. gedruckten Textes beschäftigt. Der jüngere, aber mittlerweile nicht weniger beachtete Teilbereich befaßt sich hingegen mit der *Erkennung handschriftlichen Textes*. Aufgrund der weitaus größeren Variabilität der Eingabe durch die unterschiedlichen Schreibstile der Schreiber ergeben sich hier noch deutlich schlechtere Ergebnisse, als dies bei der OCR der Fall ist, wo meist nur eine geringe Anzahl unterschiedlicher Zeichensätze berücksichtigt werden muß.

Im Fall der On-line Erkennung macht eine solche Unterteilung in die Erkennung maschinell produzierten und handschriftlichen Textes keinen Sinn, da lediglich eine handschriftliche Eingabe überhaupt einer Erkennung bedarf. Es ergibt sich daher eine Unterteilung nach der Art der Information, die aus der handschriftlichen Eingabe gewonnen werden soll. Mit der Frage, was geschrieben wurde, beschäftigt sich die *On-line Handschrifterkennung*, die Thema dieser Arbeit ist. D.h. die Trajektorie der handschriftlichen Eingabe, gegeben als eine zeitliche Sequenz von Datenpunkten, soll in eine entsprechende für den Computer verständliche textuelle Form umgewandelt werden. Im Mittelpunkt der *Unterschriftenverifikation* steht die Beantwortung der Frage, ob eine vorliegende Eingabe, in diesem Fall eine Unterschrift, tatsächlich von einer bestimmten Person stammt [PL89].

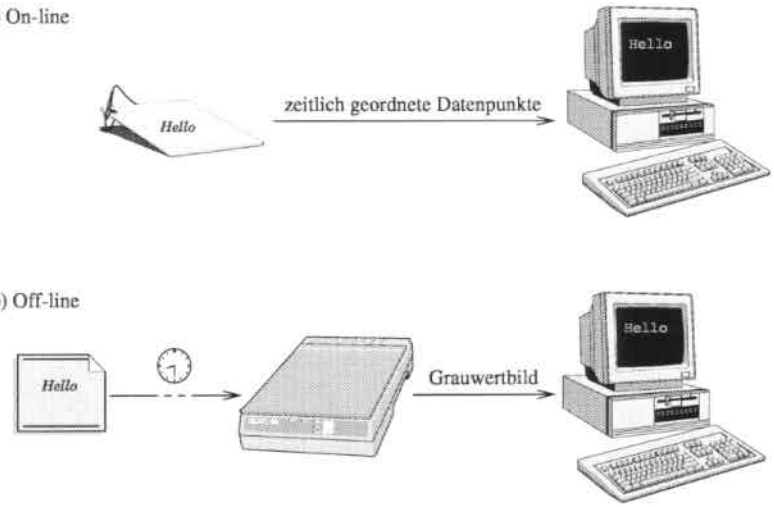


Abbildung 1.2: Zeitlicher Verlauf der On-line (a) und Off-line Erkennung (b)

1.3 On-line versus Off-line Handschrifterkennung

Die On-line und Off-line Handschrifterkennung verfolgen dasselbe Ziel: die maschinelle Umsetzung einer handschriftlichen Eingabe in eine für den Computer verständliche und damit maschinell weiterverarbeitbare Repräsentation. Dennoch unterscheiden sich einzelne Schritte im Erkennungsprozeß für beide Problembereiche erheblich, insbesondere weite Teile der Vorverarbeitung. Begründet liegen diese voneinander abweichenden Verarbeitungsschritte in dem unterschiedlichen Informationsgehalt der Eingabe.

Im Fall der Off-line Erkennung, bei der die Erkennung von dem eigentlichen Entstehungsprozeß der Handschrift zeitlich entkoppelt ist, stehen keine Informationen darüber zur Verfügung, in welcher zeitlichen Abfolge die Trajektorie der vorliegenden Eingabe entstanden ist. Die Erkennung basiert hier also auf einem statischen Schwarzweiß- oder Grauwertbild der Handschrift. Bei der On-line Erkennung hingegen liegen die Eingabedaten als ein Signal vor, das die Punkte der Trajektorie als Funktion über die Zeit beinhaltet. Die zeitliche Information über den Entstehungsprozeß der Schrift kann demnach für die Erkennung genutzt werden. Da der Informationsgehalt des On-line Signals höher als der des Grauwertbildes der Off-line Erkennung ist, sind bei vergleichbarer Eingabe bessere Erkennungsraten des On-line Erkenners zu erwarten, sofern die Systemarchitektur die zusätzlichen Informationen auch tatsächlich gewinnbringend nutzt (siehe Abschnitt 1.4) [Bei96, SSE96]. Mindestens sollte ein On-line Erkenner die Leistung eines Off-line Erkenners erreichen können, da jederzeit das On-line Signal durch Entfernung der zeitlichen Information

in ein Off-line Signal transformiert werden kann.

Die bei der On-line Handschrifterkennung vorhandenen dynamischen Schreibinformationen vereinfachen den Erkennungsprozeß gegenüber Off-line Systemen in mehrerlei Hinsicht. So stellen Überlappungen von Zeichen ein wesentlich geringeres Problem dar, da die Zeichen, wenn nicht durch die räumliche Distanz, dann doch durch die zeitliche Differenz zwischen der Entstehung der überlappenden Teile separierbar sind. Darüber hinaus können ggf. zusätzliche Informationen wie Schreibdruck, Geschwindigkeit und Beschleunigung zur Erkennung hinzugezogen werden. Der Schreibdruck schlägt sich in Off-line Systemen meist in unterschiedlichen Strichdicken nieder, die durch Konturfindung und Skelettierung aufwendig beseitigt werden müssen, da sie die Erkennung erschweren. Bei On-line Systemen wird der Schreibdruck zwar unter Umständen durch unterschiedliche Strichstärken visualisiert, beeinflußt jedoch nicht das eigentliche Eingabesignal.

Auf der anderen Seite wird die Erkennung durch die dynamischen Schreibinformationen auch erschwert. Denn während die Off-line Erkennung invariant gegenüber der zeitlichen Entstehung eines Zeichens oder Wortes ist, ist dies bei der On-line Erkennung nicht der Fall. Viele Zeichen bestehen aus mehreren Komponenten, d.h. einzelnen Strichen, die durch Auf- und Absetzen des Stiftes begrenzt sind, und können daher durch unterschiedliche Reihenfolgen der Komponenten erzeugt werden. Während dabei immer das gleiche statische Bild entsteht, erhält man jeweils verschiedene zeitliche Folgen von Koordinaten. Zusätzlich gibt es für jeden Strich noch zwei entgegengesetzte Schreibrichtungen. Abbildung 1.3 zeigt als Beispiel einige mögliche und auch tatsächlich verwendete Schreibreihenfolgen des Großbuchstabens „E".

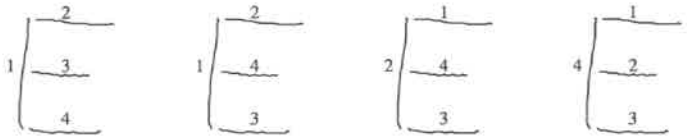


Abbildung 1.3: Unterschiedliche Schreibreihenfolgen des Buchstabens „E"

Nicht nur innerhalb eines einzelnen Zeichens stellt dies ein Problem dar, sondern auch über mehrere Zeichen hinweg. Während beim Schreiben von Wörtern und Sätzen zwar in der Regel die Schreibrichtung von links nach rechts eingehalten wird, gibt es mit den sog. verzögerten Strichen eine häufige Ausnahme. Hierzu gehören u.a. „i"-Punkte und „t"-Striche, die je nach Gewohnheit eines Schreibers entweder direkt nach dem zugehörigen Zeichen oder aber erst später nach dem nächsten Absetzen des Stiftes, z.B. am Ende des Wortes, hinzugefügt werden und damit zeitlich weit entfernt von ihren zugehörigen Zeichen sind. Beispiele solcher verzögerter „i"-Punkte und „t"-Striche zeigt Abbildung 1.4.

Bedingt durch die unterschiedlichen Voraussetzungen des Eingabesignals eröffnen sich für die On-line und Off-line Erkennung verschiedene Anwendungsbereiche. Off-line Systeme finden ihre Anwendungsbereiche überall dort, wo große Mengen bereits

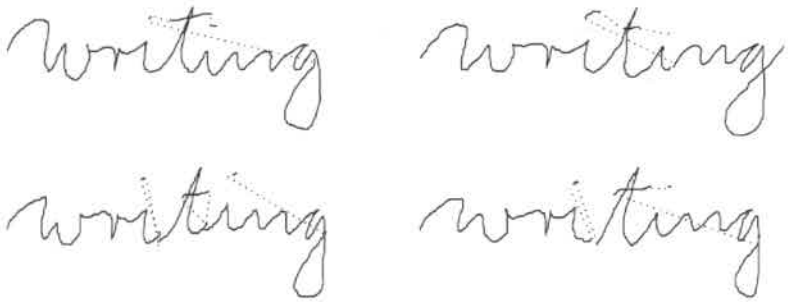


Abbildung 1.4: Beispiele verzögerter „i"-Punkte und „t"-Striche

existierenden Textes maschinell erkannt werden sollen. Dies ist z.B. bei der Sortierung von Briefen in Postverteilungszentren oder der Verarbeitung von Schecks bzw. Überweisungsformularen im Giroverkehr der Fall. Aufgrund der fehlenden dynamischen Schreibinformationen sind hier On-line Systeme nicht einsetzbar. Diese kommen in erster Linie bei der direkten Mensch-Maschine-Kommunikation zum Einsatz.

1.4 Der Wert zeitlicher Informationen

Es ist durch Experimente bereits seit Jahren bekannt, daß das Vorhandensein dynamischer Schreibinformationen zu verbesserten Erkennungsraten gegenüber traditionellen Off-line Systemen führt. In diesen Experimenten wurden On-line Daten durch Entfernung der zeitlichen Information in äquivalente Off-line Daten transformiert und bei einer ähnlichen Erkennerarchitektur eine deutliche Verschlechterung der Erkennungsrate beobachtet [SSE96]. Dies führt zu der Einsicht, daß bei der On-line Erkennung soviel Nutzen aus den dynamischen Schreibinformationen gezogen werden sollte wie möglich [Bei96]. Eine einfache Umwandlung der On-line Daten in Off-line Daten und die Anwendung der entsprechend aus der Off-line Erkennung bekannten Algorithmen führen offensichtlich nicht zum optimalen Ergebnis.

Mittlerweile ist auch in der Forschung auf dem Gebiet der Off-line Erkennung zu beobachten, daß versucht wird, aus den statischen Daten dynamische Informationen zu gewinnen, d.h. die ursprüngliche, zeitliche Entstehung der Trajektorie zu rekonstruieren, um die Erkennungsraten zu verbessern [BCCM93, GWS92, DR95, AA93].

1.5 Problematik der Handschrifterkennung

Die Leistung eines On-line Handschrifterkenners, d.h. seine Erkennungsrate, wird maßgeblich durch die Zahl und Art der Einschränkungen bezüglich der zugelassenen Eingabe bestimmt (Abbildung 1.5). Je größer die Zahl der Einschränkungen ist, desto

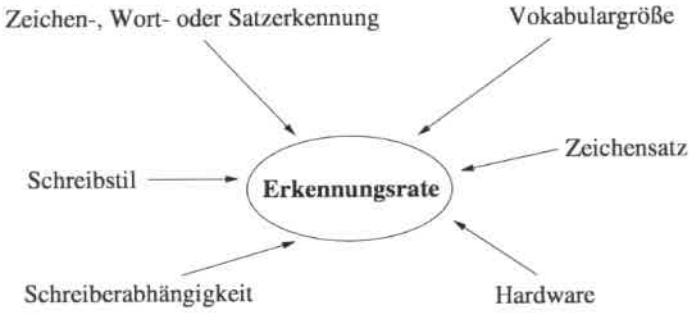


Abbildung 1.5: Einflußfaktoren auf die Erkennungsleistung

besser wird die Leistung des Erkennungssystems. Andersherum bedeutet dies, je mehr Freiheitsgrade bei der Eingabe bestehen, desto schlechter sind die zu erwartenden Erkennungsraten. Daher sind in der entsprechenden Literatur veröffentlichte Ergebnisse nur dann aussagekräftig, wenn alle Einschränkungen für das jeweilige System bekannt sind. Und selbst dann ist ein Vergleich unterschiedlicher Systeme äußerst schwierig, da jedes System andere Einschränkungen verlangt und nicht bekannt ist, wie sich das System ggf. ohne diese Einschränkung verhalten würde. Das Idealsystem ohne jegliche Einschränkungen, d.h. ein System, bei dem beliebiger Text, bestehend aus beliebigen Zeichen von einem beliebigen Schreiber in einem beliebigen Schreibstil, erkannt werden kann, hat die Forschung erst ansatzweise erreicht.

Für jedes Erkennungssystem sind also eine Reihe von Fragen zu beantworten, um die Leistung beurteilen zu können:

- Handelt es sich um einen Einzelzeichen-, Wort- oder Satzerkenner? Wie groß ist im Falle eines Wort- bzw. Satzerkenners der Wortschatz des Systems, oder kann es beliebige Wörter erkennen?

- Welche Schreibstile kann das System verarbeiten? Ist ein bestimmter Schreibstil, z.B. Druckschrift, Voraussetzung für eine gute Erkennungsleistung?

- Ist es ein schreiberabhängiges System, das nur für einen speziellen Benutzer funktioniert, oder ist das System schreiberunabhängig und damit ohne Anpassung von beliebigen Schreibern nutzbar?

- Wurde das System für eine spezielle Hardware entwickelt, oder kann es unterschiedliche technische Eigenschaften der Geräte zur Aufnahme der Handschrift durch entsprechende Vorverarbeitungstechniken kompensieren?

Für ein besseres Verständnis der jeweils auftretenden Problematik werden diese Fragestellungen in den folgenden Abschnitten ausführlicher beleuchtet.

1.5.1 Einzelzeichen-, Wort- und Satzerkennung

Nicht alle handschriftlichen Eingaben im Rahmen der Mensch-Maschine-Kommunikation erfordern universelle Handschrifterkenner, die alle Zeichen einer Sprache und beliebige Zeichenfolgen, d.h. uneingeschränkten Text, erkennen können. Häufig kann die Erkennung auf bestimmte Zeichenklassen (nur Ziffern, nur Klein- oder Großbuchstaben) und Textarten (Einzelzeichen oder isolierte Wörter) eingeschränkt werden. Diese Einschränkungen hängen jeweils von der aktuellen Anwendung ab. So gibt es in vielen Formularen festgelegte Eingabefelder für Datum, Namen, Zahlen, Geldbeträge usw., die ausgefüllt werden müssen. Innerhalb dieser Eingabefelder wird nur ein bestimmter Eingabetyp erwartet, z.B. ein Städtename, für dessen Erkennung ein Einzelworterkenner ausreicht. Das Augenmerk sollte also nicht nur auf universelle Texterkenner, die jeglichen Eingabetyp verarbeiten können, gerichtet werden, sondern auch auf weniger universelle Einzelzeichen- und Worterkenner, da sie in ihren jeweiligen Spezialbereichen in der Regel schneller und mit höheren Erkennungsraten arbeiten können.

Ein Großteil des Aufwandes der Forschung auf dem Gebiet der Handschrifterkennung wurde bislang für die Erkennung von Einzelzeichen verwendet. Diese ist insbesondere für Sprachen mit einer großen Anzahl unterschiedlicher Zeichen, wie z.B. Chinesisch mit mehreren Tausend Zeichen, von Bedeutung. Bei der *Einzelzeichenerkennung* wird von einem einzelnen, isolierten Zeichen ausgegangen, das es zu erkennen gilt. Eine Überlappung, Verbindung oder anderweitige Beziehung mit anderen Zeichen tritt in diesem Fall nicht auf. Die Schwierigkeit der Erkennungsaufgabe wird hier durch die Anzahl und Art der möglichen Zeichenklassen bestimmt. Je größer die Anzahl der möglichen Alternativen ist und je verwechselbarer die einzelnen Zeichen sind, desto schwieriger gestaltet sich die Erkennungsaufgabe und desto geringer ist die Erkennungsrate. So sind bei der Ziffernerkennung, bei der es lediglich die 10 Zeichen „0"...„9" zu unterscheiden gilt, die höchsten Erkennungsraten zu erwarten, zumal sich die Ziffern signifikant unterscheiden (außer bei manchen Schreibern die Ziffern „1" und „7") und nur wenige Schreibvarianten für die einzelnen Zeichen existieren. Bereits deutlich schlechtere Erkennungsraten erhält man bei der Klassifikation der Großbuchstaben „A"...„Z", was zum einen auf die größere Anzahl der Zeichenklassen und zum anderen auf die weitaus größere Anzahl der Schreibvarianten für jeden der Buchstaben zurückzuführen ist. Bei gleicher Anzahl von Zeichenklassen wie bei der Großbuchstabenerkennung gestaltet sich die Erkennung der Kleinbuchstaben "a"..."z" noch schwieriger. Der Grund hierfür liegt in der höheren Verwechselbarkeit der Kleinbuchstaben. So sind z.B. die Buchstaben „e" und „l" bei manchen Schreibweisen auch von Menschen ohne zusätzliche Kontextinformationen, wie benachbarte Buchstaben zum Größenvergleich, nicht voneinander zu unterscheiden. Gleiches gilt auch, wenn Ziffern, Klein- und Großbuchstaben (und ggf. noch Sonderzeichen) vermischt werden. In diesem Fall sind z.B. die Ziffer „0", der Kleinbuchstabe „o" und der Großbuchstabe „O" ohne Kontextinformationen kaum zu unterscheiden.

Bei der *Worterkennung* besteht die Eingabe nicht mehr nur aus einem einzigen Zei-

chen, sondern einer Folge von Zeichen, die zusammen ein Wort bilden. In Abhängigkeit des jeweiligen Schreibstils sind die einzelnen Zeichen entweder durch ein Abheben und Wiederaufsetzen des Stiftes getrennt (Druckschrift) oder direkt miteinander verbunden (kursive Handschrift) (siehe Abschnitt 1.5.2). Insbesondere kursive Handschrift macht eine Segmentierung des Wortes in seine Einzelzeichen schwierig, vor allem dann, wenn einzelne Zeichen des Wortes unlesbar sind, wie es häufig in natürlicher Handschrift vorkommt. In diesem Fall helfen ein Vokabular oder Sprachmodell bei der Erkennung der Zeichenfolge (siehe Abschnitt 1.5.4). Anhand des Wissens, welche Zeichenfolgen ein legales Wort der Sprache bilden, kann dann ggf. trotz einzelner nicht erkennbarer Wortteile auf das richtige Wort geschlossen werden, wenn die erkennbaren Zeichen zu diesem Wort passen. Der Nachteil dieses Vokabularansatzes, der in den meisten heutigen Erkennern zum Einsatz kommt, ist, daß das Erkennungssystem tatsächlich nur die Wörter aus dem Vokabular erkennen kann. Wird ein Wort geschrieben, das nicht im Vokabular aufgeführt ist, führt dies zu einer Fehlerkennung, da auf jeden Fall das zu dieser Eingabe am besten passende Wort aus dem Vokabular ausgegeben wird. Je nach Anwendung sollte daher das Vokabular möglichst groß gewählt werden, sofern die Erkennungsrate dies zuläßt, oder aber flexibel gestaltet werden, so daß es leicht erweitert oder angepaßt werden kann.

Zusätzlich zum Problem der Zerlegung von Wörtern in ihre Einzelzeichen, d.h. des Auffindens der Buchstabengrenzen, kommt bei der *Satzerkennung* das Auffinden der Wortgrenzen hinzu. Im Gegensatz zur Worterkennung ist diese Zerlegung bei kursiver Handschrift einfacher, da der Stift in den meisten Fällen nur zum Beginn eines neuen Wortes abgehoben und wieder aufgesetzt wird, was die Anzahl der möglichen Wortanfänge stark einschränkt. Bei Druckschrift fehlt dieser deutliche Hinweis auf einen Wortanfang jedoch, da auch jedes neue Zeichen durch ein Abheben und Wiederaufsetzen des Stiftes eingeleitet wird. Es wird lediglich zwischen den Wörtern etwas mehr Freiraum gelassen als zwischen den einzelnen Zeichen. Ebenso wie bei der Worterkennung, wo die Erkennung durch die Einführung von Vokabularen vereinfacht wird, können zur Erleichterung des Segmentierungsproblems auf Wortebene und zur Kompensierung unlesbarer Wörter Kontextinformationen in Form sog. Sprachmodelle genutzt werden. Diese Sprachmodelle werden aus großen Textmengen berechnet und geben die Wahrscheinlichkeit für bestimmte Wortkombinationen an, d.h. wie wahrscheinlich es ist, daß ein bestimmtes Wort auf ein anderes gegebenes Wort des Vokabulars folgt.

1.5.2 Schreibstile

Der Schreibstil beeinflußt in hohem Maße den Schwierigkeitsgrad der Handschrifterkennung. Je natürlicher der als Eingabe zugelassene Schreibstil ist, d.h. je weniger Einschränkungen es gibt, desto schwieriger ist das Erkennungsproblem zu lösen. Das einfachste Problem ist die Erkennung von sog. Blockschrift, welche dem Schreiber die wenigste Freiheit im Schreibstil läßt. Bei der Blockschrift sind die einzelnen Zeichen eines Wortes oder Satzes deutlich dadurch getrennt, daß sie entweder in vorgegebene

Kästchen oder mit deutlichem Abstand zueinander geschrieben werden müssen (Abbildung 1.6a). Überlappungen treten in diesem Fall nicht auf. Die Erkennung kann aufgrund des minimalen Segmentierungsproblems mit Mitteln der Einzelzeichenerkennung gelöst werden. Diese Art der Eingabe wird häufig auf Überweisungsformularen und ähnlichen Formularen verlangt, wo eine höchstmögliche Erkennungsrate gefordert ist.

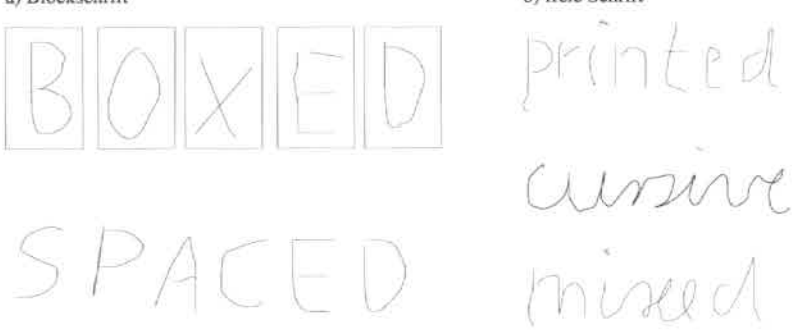


Abbildung 1.6: Beispiele unterschiedlicher Schreibstile

Weitaus schwieriger gestaltet sich die Erkennung kontinuierlicher Handschrift, die das Thema dieser Arbeit ist. Dies ist die natürlichste Form der Eingabe für einen Schreiber, denn es werden ihm keinerlei Beschränkungen bezüglich des Schreibstils auferlegt. Er kann den bereits vom Schreiben auf Papier gewohnten Schreibstil verwenden. Bei kontinuierlicher Handschrift können wiederum drei Schreibstile unterschieden werden: Druckschrift, Kursivschrift und eine Mischform aus Druck- und Kursivschrift (Abbildung 1.6b). Die ersten beiden Schreibstile sind dabei die extremsten Schreibstile in Bezug auf die Häufigkeit des Absetzens des Stiftes zwischen den Buchstaben. D.h. bei der reinen Druckschrift wird nach jedem Buchstaben der Stift abgehoben und an anderer Stelle, am Anfang des nächsten Buchstabens, wieder aufgesetzt. Dabei kann es im Gegensatz zur Blockschrift auch zu Überlappungen der einzelnen Buchstaben kommen. Bei der reinen Kursivschrift wird der Stift während des Schreibens lediglich abgesetzt, um „i"-Punkte, „t"-Striche u.ä. zu machen. Alle Buchstaben sind also durch Übergänge miteinander verbunden. Bei der Mischform aus Druck- und Kursivschrift wird der Stift jeweils nach ein paar Buchstaben abgehoben. Dieser Schreibstil ist bei Schreibern am häufigsten zu beobachten.

Eine wichtige Beobachtung bei der Unterscheidung in Druck- und Kursivschrift ist, daß sich für die beiden Schreibstile häufig sehr unterschiedliche Zeichenformen ergeben. So verwenden Schreiber, deren natürlicher Schreibstil die Druckschrift ist, meist Druckbuchstaben, während Kursivschreiber jeweils die kursive Variante der Buchstaben wählen, wie auch in Abbildung 1.6b zu sehen ist. Zusätzlich ergeben sich sowohl für die Druckbuchstaben als auch die meisten kursiven Buchstaben jeweils

mehrere Schreibvarianten.

Systeme zur On-line Handschrifterkennung warten in der Regel für Druckschrift mit deutlich höheren Erkennungsraten auf, als dies für Kursivschrift der Fall ist. Die Ursache hierfür liegt zum einen darin, daß in den meisten Fällen die einzelnen Zeichen eines Druckschriftschreibers deutlicher geschrieben sind und damit auch leichter maschinell zu erkennen sind. Zum anderen wird bei Druckschrift die Segmentierung dadurch erleichtert, daß jedem neu angefangenen Buchstaben ein Abheben und Wiederaufsetzen des Stiftes vorausgeht. Zu beachten ist jedoch, daß das Abheben und Wiederaufsetzen des Stiftes nicht zwangsläufig einen Buchstabenanfang markiert, sondern auch innerhalb der einzelnen Zeichen vorkommt, insbesondere bei Großbuchstaben, die aus mehreren Strichen zusammengesetzt werden.

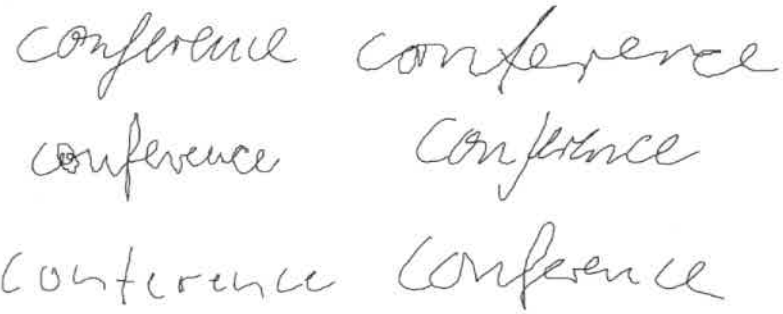


Abbildung 1.7: Beispiele aus den hier verwendeten Datenbasen mit Schreibvarianten unterschiedlicher Schreiber

1.5.3 Schreiberabhängige, schreiberunabhängige Erkennung, Adaption

Ein *schreiberabhängiger* Handschrifterkenner, der speziell für die Handschrift eines bestimmten Schreibers entwickelt wurde, wird für diesen Schreiber deutlich besser funktionieren als ein schreiberunabhängiges System, da die Anzahl der möglichen Schreibvarianten für die einzelnen Zeichen auf ein Minimum reduziert wird. Der Nachteil eines schreiberabhängigen Systems ist, daß es für jeden individuellen Schreiber extra angepaßt werden muß. Dafür bedarf es im einfachsten Falle eines „Trainings" des Systems mit einer größeren Menge von Handschriftproben des entsprechenden Schreibers.

In einem *schreiberunabhängigen* System wird versucht, von vornherein die unterschiedlichen Schreibvarianten (siehe z.B. Abbildung 1.7) zu berücksichtigen, so daß die Anpassung auf einen speziellen Schreiber nicht mehr erforderlich ist. Die zeitaufwendige Lernphase des Systems beim Endbenutzer entfällt. Das System kann also gleichzeitig von beliebig vielen Benutzern genutzt werden.

Ein Weg, die Vorteile eines schreiberabhängigen mit denen eines schreiberunabhängigen Systems zu verbinden, ist die *Adaption* auf den jeweiligen Benutzer, während er das System benutzt [GHA+92, SN97]. Dabei wird von einem schreiberunabhängigen Erkenner ausgegangen, der bereits gute Erkennungsleistungen auf beliebigen Handschriften aufweisen sollte. Dieser Erkenner kann dann aus den Fehlern, die er während der Benutzung macht und die manuell vom Benutzer korrigiert werden, lernen, diese Fehler in Zukunft zu vermeiden. Er stellt sich im Laufe der Zeit auf die Handschrift des Benutzers ein, und die Erkennungsleistung steigt für diesen einen Benutzer an. Beobachtungen bei solchen adaptiven Systemen haben gezeigt, daß in der Tat am Anfang ein Anstieg der Erkennungsleistung zu bemerken ist, die jedoch im weiteren Verlauf der Benutzung wieder etwas abfällt. Denn der Benutzer, der am Anfang noch verhältnismäßig sauber geschrieben hat, um die Grenzen des Systems auszuloten, wird durch die zunehmende Erkennungsleistung des Systems ermutigt und fängt an, unsauberer zu schreiben. Dadurch „verlernt" das System die anfänglich sauberere Schrift wieder. Ein solches System macht auch in Fällen, bei denen es von mehreren Benutzern im ständigen Wechsel verwendet wird, keinen Sinn, da es sich jedesmal neu auf den jeweiligen Benutzer einstellen muß und die Handschrift des Vorbenutzers wieder vergißt.

1.5.4 Vokabulare und Sprachmodelle

In nahezu allen heutigen Handschrifterkennern mit akzeptablen Erkennungsraten werden *Vokabulare* eingesetzt, welche die mögliche Eingabe auf eine vorgegebene Liste erlaubter Wörter beschränken. Wörter, die nicht im Vokabular aufgeführt sind, können auch nicht vom Erkenner erkannt werden und führen zu Fehlerkennungen. Die Größe des Vokabulars, d.h. die Anzahl der Wörter, die erkannt werden können, ist ein wichtiger Faktor bezüglich der Akzeptanz des Systems bei seinen Benutzern. Der Umstand, daß mit steigender Vokabulargröße aufgrund höherer Verwechselbarkeit die Erkennungsrate des Systems sinkt, führt dabei zu einem Dilemma. Denn zum einen sollte das Vokabular so groß wie möglich sein, um die Wahrscheinlichkeit zu erhöhen, daß das geschriebene Wort tatsächlich im Vokabular ist. Zum anderen möchte der Benutzer eine möglichst hohe Erkennungsrate, welche u.a. durch Beschränkung des Vokabulars erreicht werden kann. Die Wahl der Vokabulargröße muß also in Abhängigkeit der Erkennungsrate, die mit dieser Größe erreicht werden kann, erfolgen. Vokabulargrößen ab 20 000 Wörtern sollten dabei mindestens erreicht werden, um zu häufige Fehlerkennungen durch nicht vorhandene Wörter im Vokabular zu vermeiden. Nicht vergessen werden sollte auch der Umstand, daß die Vokabulargröße einen großen Einfluß auf die Geschwindigkeit und den Speicherbedarf des Erkenners hat, d.h. je größer das Vokabular, desto höher auch die Erkennungszeiten. Üblicherweise macht in Erkennungssystemen die Suche im Vokabular einen großen, wenn nicht gar den größten Teil der Erkennungszeit aus.

Neben Vokabularen sind *Sprachmodelle* eine weitere Möglichkeit, syntaktisches Wissen über die zugrundeliegende Sprache in die Erkennung einzubeziehen. Sprach-

modelle stellen bei der Erkennung von Einzelworten oder Sätzen die a priori Wahrscheinlichkeiten für bestimmte Zeichen- bzw. Wortfolgen zur Verfügung. Mit dem Einsatz von Sprachmodellen kann in der Regel die Fehlerrate deutlich reduziert werden, da durch sie häufige Zeichen- bzw. Wortfolgen der Sprache bei der Erkennung gegenüber seltenen Folgen favorisiert werden. Allerdings wird die Erkennung hierdurch auf eine bestimmte Anwendungsdomäne beschränkt, da die Sprachmodelle auf einem gegebenen Textkorpus dieser Domäne berechnet werden und somit auch nur die a priori Wahrscheinlichkeiten der Wortfolgen dieser Domäne widerspiegeln.

1.5.5 Länderspezifische Besonderheiten

Zu dem Problem, daß in einem schreiberunabhängigen System mehrere Schreibvarianten für die einzelnen Zeichen auftreten und dadurch die Erkennung erschwert wird, kommen in einigen Fällen noch länderspezifische Besonderheiten in der Schreibweise hinzu. Durch diese Besonderheiten wird ggf. nicht nur die Anzahl der Schreibvarianten für ein Zeichen erhöht, sondern auch die Verwechselbarkeit der Zeichen untereinander. D.h. die Erkennung von bestimmten Zeichen, die sich innerhalb einer Nationalität deutlich unterscheiden, wird dadurch erschwert, daß auch die Schreibweisen anderer Nationen berücksichtigt werden.

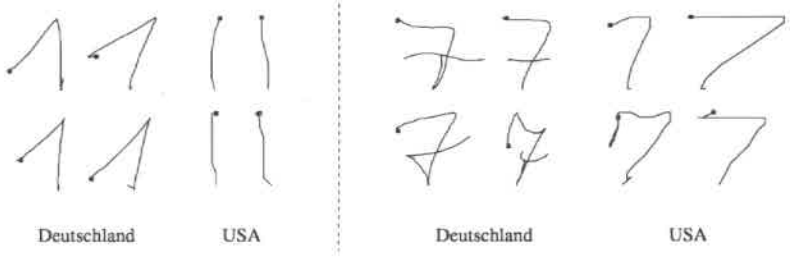


Abbildung 1.8: Nationale Varianten des gleichen Zeichens

Ein Beispiel hierfür sind die Unterschiede in der Schreibweise einzelner Zeichen zwischen deutschen und amerikanischen Schreibern. Abbildung 1.8 zeigt Beispiele aus den hier verwendeten Datenbasen für solche Schreibvarianten, die die Erkennung erschweren. Während im deutschen Raum die Zeichen „1“ und „7“ bei den meisten Schreibern deutlich zu unterscheiden sind, ist die Unterscheidung zwischen einer deutschen „1“ und einer amerikanischen „7“ schon schwieriger. Ebenso ähnelt ein amerikanisches „A“ vielfach eher einem deutschen „a“ als „A“.

1.5.6 Erkennungsprobleme durch Hardware

Mittlerweile sind eine Reihe unterschiedlicher Geräte auf dem Markt erhältlich, um Handschrift on-line aufzuzeichnen. Je nach verwendetem Gerät kann die aufgezeich-

nete Handschrift deutlich von der natürlichen Handschrift eines Schreibers auf Papier abweichen. Am häufigsten sind z.B. Verzerrungen der Schrift in vertikaler oder horizontaler Richtung aufgrund zu glatter Oberflächen oder eckige Trajektorien aufgrund zu geringer Abtastraten oder -auflösungen zu beobachten. Wird ein Handschrifterkenner lediglich für eine bestimmte Hardware entwickelt, so können die speziellen technischen Gegebenheiten des Gerätes ggf. berücksichtigt werden. Im Falle eines Erkenners, der auf unterschiedlichen Plattformen lauffähig sein soll, muß die durch die Geräte verursachte Variabilität der Eingabe soweit möglich durch den Erkenner kompensiert werden.

Abbildung 1.9: Unterschiedliche Hardware

Die Palette der erhältlichen Eingabegeräte reicht von einfachen Grafiktabletts, die ursprünglich für grafische Anwendungen entwickelt wurden, über drucksensitive Bildschirme bis zu integrierten LCD-Tabletts (siehe Abbildung 1.9). Grafiktabletts sind in der Regel über eine serielle Schnittstelle an den jeweiligen Rechner, auf dem der Handschrifterkenner läuft, angeschlossen. Die Aufnahme der Handschrift ist dabei vollständig von der Visualisierung der Eingabe auf dem separaten Bildschirm entkoppelt, d.h. die Schrift ist nicht auf dem Tablett, sondern lediglich auf dem Bildschirm sichtbar. Diese Art des Schreibens auf solchen Tabletts ist für die Benutzer sehr gewöhnungsbedürftig, da entweder ohne sichtbare Rückkopplung geschrieben oder während des Schreibens auf den Bildschirm geschaut werden muß. Eine Ausnahme bilden Grafiktabletts, bei welchen ein Blatt Papier auf die elektrostatische Oberfläche aufgelegt und mit einem speziellen Stift, der tatsächlich die Schrift auf dem Papier sichtbar macht, geschrieben werden kann. Diese Tabletts vermitteln im Vergleich zu den meisten anderen Geräten das natürlichste Schreibgefühl. Einen Mittelweg zwischen Grafiktabletts und LCD-Tabletts bilden drucksensitive Bildschirme, bei denen auf die Scheibe eines normalen Bildschirms eine drucksensitive Folie aufgebracht ist. Diese Folie übermittelt die aufgezeichneten Signale ebenfalls über eine serielle Schnittstelle an den Rechner. Der Vorteil dieser Bildschirme ist, daß die Visualisierung der elektronischen Schrift direkt an der Stelle des Schreibens stattfindet. Der Nachteil ist jedoch, daß der Schreiber in der Regel auf einer senkrechten Fläche schreiben muß, sofern der Bildschirm nicht waagerecht in die Tischplatte eingelassen wurde. Außerdem haben drucksensitive Bildschirme die Eigenschaft, daß die Hand beim Schreiben

Abbildung 1.10: Unterschiedliche Abtastraten

nicht aufgelegt werden darf, da ansonsten diese Auflagepunkte ebenfalls als Datenpunkte an den Rechner übermittelt würden. Ein weitaus angenehmeres Schreibgefühl vermitteln hier integrierte LCD-Tabletts. Diese Geräte verbinden die Technik der Grafiktabletts mit dünnen LCD-Bildschirmen, wie sie heute in den meisten Laptops verwendet werden. Der Vorteil gegenüber drucksensitiven Bildschirmen ist, daß sich die Schreibfläche in der Waagerechten befindet und der Handballen beim Schreiben aufgelegt werden kann. Außerdem ist bei LCD-Tabletts durch die dünnere Oberfläche das Problem der Paralaxe, d.h. die sich durch den nicht senkrechten Blickwinkel des Schreibers ergebene Differenz zwischen der Stiftspitze auf der Bildschirmoberfläche und dem zugehörigen dargestellten Pixel unter der Bildschirmoberfläche, deutlich geringer.

Unabhängig davon, ob es sich um ein Grafiktablett, einen drucksensitiven Bildschirm oder ein LCD-Tablett handelt, unterscheiden sich alle Geräte in weiteren technischen Eigenschaften, die die zu erkennende Eingabe beeinflussen. Die Schreiboberfläche des Geräts und der verwendete Stift verursachen unter Umständen Verzerrungen der Schrift und häufige „Ausrutscher", wenn die Oberfläche zu glatt ist oder kein kabelloser Stift zur Verfügung steht. Die Abtastauflösung und Abtastrate des Geräts bestimmen, wie genau die Handschrift aufgezeichnet werden kann. Ist die Auflösung zu gering und die Handschrift zusätzlich noch sehr klein, stehen nur wenige und ungenaue Daten für die Erkennung zur Verfügung. Gleiches gilt bei zu geringer Abtastrate des Gerätes, wenn gleichzeitig sehr schnell geschrieben wird. In diesem Fall kann das Gerät nicht mit der Handschrift Schritt halten, und es werden zu wenige Datenpunkte aufgenommen. Abbildung 1.10 zeigt Beispiele mit unterschiedlicher Abtastrate aufgenommener Daten.

Neuere Geräte können bei Bedarf über die reine Trajektorie hinaus noch zusätzliche Informationen während des Schreibens erfassen, die bei der Erkennung genutzt werden können. Beispiele hierfür sind der Schreibdruck oder der Winkel zwischen dem Stift und der Oberfläche. Da aber nicht alle Geräte diese Informationen zur Verfügung stellen, bleiben diese Informationsquellen in den meisten existierenden Erkennern noch unberücksichtigt.

1.6 Beiträge dieser Arbeit

Diese Arbeit befaßt sich mit dem schwierigsten Problem der On-line Handschrifterkennung. Es wird ein schreiberunabhängiges System zur Erkennung von Einzel-

zeichen, Einzelwörtern und Sätzen vorgestellt, das kontinuierliche Handschrift unabhängig vom Schreibstil erkennen kann. Dabei erfolgt die Verarbeitung der handschriftlichen Eingabe selbst bei großen Vokabularen mit über 20 000 Wörtern in Echtzeit. Das Vokabular ist ohne Anpassung des Erkenners austauschbar und kann noch zur Laufzeit dynamisch erweitert werden. Unterschiedliche technische Gegebenheiten der Aufnahme-Hardware werden durch das System kompensiert. Über Einzelzeichen und -wörter hinausgehend, ist dieser Handschrifterkenner einer der ersten, welcher auch Wortfolgen verarbeiten kann.

Evaluiert auf drei verschiedenen Datenbasen mit handschriftlichen Daten von mehr als 500 Schreibern aus Deutschland und den USA, erzielt dieses System für die Einzelwort- und Satzerkennung die bislang besten publizierten Erkennungsraten (siehe Tabelle 1.1). Mit 93.4% Erkennungsrate auf Einzelworten für ein 20 000 Wörter umfassendes Vokabular wird der beste vergleichbare, auf Hidden Markov Modellen (HMM) basierende Erkenner, der auf einer unterschiedlichen Datenbasis getestet wurde, um mehr als 4% übertroffen. Das beste auf neuronalen Netzen basierende System erreicht im Vergleich nur 80% Erkennungsrate. Selbst mit einem 50 000 Wörter umfassenden Vokabular liegt die Erkennungsrate mit 91.2% noch über den Ergebnissen der anderen Ansätze mit 20 000 Wörtern. Bei der Erkennung von Wortfolgen wird das einzige vergleichbare System, für das bisher schreiberunabhängige Ergebnisse von 63% (ebenfalls auf unterschiedlichen Daten) veröffentlicht wurden, mit 86.6% Erkennungsrate für ein Vokabular mit 20 000 Wörtern um mehr als 23% übertroffen.

	Ansatz	Vokabulargröße	Erkennungsrate
Wörter			
Seni [Sen95]	TDNN	21 000	62.4%
Schenkel et al. [SGH94, Sch95]	TDNN	25 000	80.0%
Dolfing et al. [DHU97]	HMM	20 000	88.8%
Manke	MS-TDNN	20 000	93.4%
Sätze			
Ratzlaff et al. [RNM96]	HMM	20 000	63.0%
Manke	MS-TDNN	20 000	86.6%

Tabelle 1.1: Ergebnisse im Vergleich mit anderen Systemen (jeweils auf unterschiedlichen Datenmengen getestet)

Erreicht wird diese hervorragende Erkennungsleistung durch neue oder erweiterte Methoden der Vorverarbeitung und einen erstmals für die Handschrifterkennung eingesetzten hybriden Erkennungsansatz aus neuronalem Netzwerk und Hidden Markov Modellen zur Modellierung syntaktischen Wissens über die zugrundeliegende Sprache. Die wichtigsten Beiträge dieser Arbeit sind im folgenden aufgelistet:

- **Vorverarbeitung.** Die Vorverarbeitung der handschriftlichen Eingabe berücksichtigt im Vergleich mit anderen, insbesondere rein auf Hidden Markov Modellen basierenden Erkennern alle wesentlichen Effekte, die in dem Eingabesignal

auftreten können und die Erkennung erschweren. Hier wird erstmals eine nichtlineare Interpolation mittels Bézier-Kurven für die Rekonstruktion der Trajektorie bei Abtastlücken eingesetzt. Die Auswirkungen einzelner Normalisierungsschritte auf die Erkennungsrate werden jeweils anhand von unterschiedlichen Datenbasen evaluiert. Über die in anderen Systemen übliche Extraktion zeitlich lokaler Merkmale hinausgehend, werden hier zusätzlich globale, von der Zeit unabhängige Merkmale bei der Erkennung berücksichtigt [MFW94]. Ergebnis der Vorverarbeitung ist eine Repräsentation, die den zeitlichen Charakter des ursprünglichen Eingabesignals bewahrt.

- **Neuronaler Erkenner.** In diesem System wird eine Modellierung einzelner Zeichen durch eine Folge von drei Zuständen in Verbindung mit einer aus der Spracherkennung stammenden neuronalen Netzarchitektur, dem Multi-State Time Delay Neural Network (MS-TDNN), eingeführt [BMW93, MFW95b, MFW95a]. Diese Modellierung, die in ähnlicher Form sonst nur in rein HMM-basierten Ansätzen für die Handschrifterkennung verwendet wird, reduziert die Fehlerrate bei der Worterkennung relativ um über 50% gegenüber der in anderen neuronalen Erkennern üblichen Betrachtung der Zeichen als Ganzes. Das MS-TDNN ermöglicht eine gleichzeitige Erkennung und Segmentierung der Eingabe. Fehlerkennungen durch frühe Segmentierungsentscheidungen während der Vorverarbeitung werden so vermieden.

- **Worterkennung mit Vokabularen.** In dieser Arbeit werden unterschiedliche Methoden der Integration von Vokabularen in den Erkennungsprozeß entwickelt und auf unterschiedlichen Datenbasen evaluiert. Erstmals werden mit dieser Arbeit in der On-line Handschrifterkennung die in einer Baumstruktur angeordneten Wortmodelle des Vokabulars direkt in die Suche der Ausgabehypothese des Erkenners eingebunden. Dies führt gegenüber anderen Ansätzen mit einem der Suche nachgeschalteten Vokabular zu einer deutlichen Fehlerreduktion. Durch Pruning-Methoden kann darüber hinaus der Suchraum stark eingeschränkt werden, so daß eine Erkennung der Eingabe in Echtzeit ermöglicht wird. Durch die hierarchische Modellierung der Wörter als Folge einzelner Zeichen und Klassifikation des MS-TDNN auf Buchstaben- bzw. Zustandsebene kann das Vokabular ohne Änderung weiterer Parameter des Erkenners modifiziert werden [MFW96].

- **Satzerkennung.** Durch Modifikation der Baumrepräsentation des Vokabulars zur zusätzlichen Modellierung von Leerzeichen zwischen Wörtern und Erweiterung des Suchalgorithmus zur Berücksichtigung von Wortübergängen wird mit dieser Arbeit erstmals ein neuronaler Erkenner für Sätze realisiert. Auch hier wird das Segmentierungsproblem auf Wortebene wie bei der Einzelworterkennung durch direkte Einbindung in den Erkennungsprozeß gelöst. Neben Vokabularen werden hier zusätzlich Sprachmodelle eingesetzt, die die Fehlerrate relativ um etwa 25% reduzieren.

- **Erkennung beliebiger Schreibstile.** Durch die Realisierung der Erkennungs-

komponente mittels einer lernenden, neuronalen Netzarchitektur können beliebige Schreibstile erkannt werden. Sowohl für Druckschrift als auch die schwieriger zu erkennende Kursivschrift werden mit demselben neuronalen Netz, das auf einer Vielzahl von Trainingsmustern unterschiedlicher Schreiber trainiert wird, gleichermaßen gute Erkennungsraten erzielt. Eine Einschränkung der Eingabe auf einen bestimmten Schreibstil ist daher nicht erforderlich.

- **Modularer Aufbau.** Der streng hierarchische Aufbau des Systems und der einzelnen Erkennungsschritte ermöglicht die Verarbeitung sowohl von Einzelzeichen als auch Wörtern und Sätzen. Vergleichbare Systeme der On-line Handschrifterkennung wurden bislang jeweils nur speziell für eine dieser Erkennungsaufgaben entwickelt. Insbesondere sind alle bisher für die Worterkennung vorgestellten Ansätze, die auf neuronalen Netzarchitekturen basieren, nur eingeschränkt auf die Satzerkennung erweiterbar. Der hier entwickelte Erkenner wird auf allen drei Erkennungsaufgaben evaluiert und erzielt dabei jeweils hervorragende Erkennungsleistungen.

1.7 Gliederung

Dieses Kapitel beleuchtete zunächst die einzelnen Teilgebiete der Handschrifterkennung und die Probleme, die es dabei zu lösen gilt. Das folgende **Kapitel 2** stellt prinzipielle Erkennungsansätze vor und führt in einige grundlegende Techniken der Handschrifterkennung ein. Einen Überblick über die aktuelle Forschungsliteratur der On-line Handschrifterkennung gibt **Kapitel 3**. **Kapitel 4** präsentiert das Erkennungssystem im Überblick, stellt die in dieser Arbeit für Experimente verwendeten Datenbasen, Vokabulare und Sprachmodelle vor und listet die besten mit diesem Ansatz erzielten Erkennungsraten auf unterschiedlichen Erkennungsaufgaben auf. Die einzelnen Komponenten des Systems werden in den daran anschließenden Kapiteln behandelt. **Kapitel 5** beschreibt die Vorverarbeitungsschritte, die von dem ursprünglichen Eingabesignal zu einer Repräsentation führen, auf der die eigentliche Erkennung basiert. In den **Kapiteln 6**, **7** und **8** folgt die schrittweise Entwicklung der Erkennerarchitektur von einem Einzelzeichenerkenner, basierend auf einem MS-TDNN, zu einem Worterkenner mit effizienter vokabulargesteuerter Suche und zu einem Satzerkenner mit Sprachmodellen. **Kapitel 9** faßt die bis dahin präsentierten Ergebnisse zusammen, gibt anhand weiterer Experimente Antwort auf verbliebene Fragen bzgl. des Verhaltens der Erkennungsrate unter bestimmten Bedingungen und analysiert die Fehlerkennungen des Systems. Eine Zusammenfassung und ein Ausblick befinden sich abschließend in **Kapitel 10**.

Kapitel 2

Grundlagen der Handschrifterkennung

2.1 Einleitung

Obwohl sich die Forschung seit nunmehr über 30 Jahren mit der maschinellen Erkennung kursiver Handschrift beschäftigt und die Erkennungsleistung stetig verbessert wurde, sind heute existierende Systeme noch immer nur unter zahlreichen Einschränkungen nutzbar. Seit den ersten Experimenten Anfang der sechziger Jahre wurden sowohl für die On-line als auch Off-line Erkennung zahlreiche Techniken entwickelt, wobei viele der Techniken auf beide Erkennungsgebiete anwendbar sind. Einige Techniken aus den ersten Jahren der Forschung finden auch heute noch ihre Anwendung in aktuellen Systemen, andere wurden nach einiger Zeit durch bessere Ansätze ersetzt.

Ein Blick auf die Forschungsliteratur der vergangenen drei Jahrzehnte läßt deutliche Entwicklungstendenzen erkennen, denen jeweils mit mehr oder weniger Abweichungen gefolgt wurde [LB93]. Dabei haben ein Großteil der Techniken, die für die Off-line Erkennung entwickelt wurden, auch ihren Einzug in der On-line Erkennung gehalten und teilweise umgekehrt. Nicht nur die Parallelität der Techniken von On-line und Off-line Erkennung ist deutlich zu erkennen, sondern auch die der Sprach- und Handschrifterkennung. Tatsächlich stammen eine Vielzahl der Techniken in heutigen Handschrifterkennern ursprünglich aus der Spracherkennung. Daher beschäftigen sich eine Reihe von Forschungsgruppen mit beiden Gebieten, um jeweils von den Erkenntnissen und Fortschritten des anderen Gebietes profitieren zu können. Viele der in diesem Kapitel dargestellten Ansätze zur On-line Handschrifterkennung gelten also für die Off-line und Spracherkennung gleichermaßen.

Inhalt dieses Kapitels ist die Beschreibung zum einen der wesentlichen Erkennungsansätze bzw. -architekturen, die in heutigen Systemen wiederzufinden sind, und zum anderen der konkreten grundlegenden Techniken, die allgemein in diesen Ansätzen zum Einsatz kommen. Dies gibt einen Einblick in den aktuellen Stand der Technik und bildet die Basis für das Verständnis der folgenden Kapitel.

2.2 Das Erkennungsproblem

Das grundlegende Handschrifterkennungsproblem ist die automatische Abbildung einer handschriftlichen Eingabe auf die zugehörige korrekte textuelle Darstellung. Die Erkennungsaufgabe kann analog der sog. „Fundamentalgleichung der Spracherkennung" wie folgt formalisiert werden:

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|X) = \underset{W}{\operatorname{argmax}}\, \frac{P(X|W) \cdot P(W)}{P(X)} = \underset{W}{\operatorname{argmax}}\, P(X|W) \cdot P(W)$$

Gesucht ist für eine handschriftliche Eingabe X die wahrscheinlichste Zeichen- oder Wortsequenz W^*, die X erzeugt. In dieser Gleichung können bereits die wesentlichen Fragestellungen der Handschrifterkennung identifiziert werden.

- **Vorverarbeitung:** Wie repräsentiert man das ursprüngliche Eingabesignal geeignet durch Merkmalsvektoren X, auf denen die Erkennung basiert?
- **Modellierung:** Wie sollen Zeichen, Wörter und Sätze modelliert werden, um $P(X|W)$ zu berechnen?
- **Sprachmodelle:** Wie werden die a priori Wahrscheinlichkeiten $P(W)$ berechnet?
- **Suche:** Wie findet man die Zeichen- oder Wortsequenz W^*, die $P(X|W) \cdot P(W)$ maximiert?

Bei einer geeigneten Repräsentation und Modellierung der Handschrift können die Probleme der Hypothesenberechnung, Berechnung von Sprachmodellen und der Suche in vielen Aspekten ähnlich den analogen Problemen der Spracherkennung gelöst werden, wie diese Arbeit zeigt.

2.3 Modellierung von Handschrift

Die On-line Handschrifterkennung ist ein hierarchisches Problem, bei dem es gilt, die ursprünglich punktbasierte Repräsentation der Eingabe schrittweise zu größeren Einheiten zusammenzufassen. Die kleinste Einheit der Schrift ist der *Punkt*, die größte Einheit hängt von der Erkennungsaufgabe ab. Bei der Einzelzeichenerkennung ist dies ein einzelnes *Zeichen* aus einem gegebenen Zeichensatz Z, bei der Worterkennung ein *Wort*, ggf. aus einem festen Vokabular W, und letztlich bei der Satzerkennung ein vollständiger *Satz*. Ein einzelnes Zeichen ist in der Regel die, aus Sicht des Menschen und der Sprache, kleinste bedeutungstragende Einheit der Handschrift. Durch Zusammenfassen mehrerer Zeichen entstehen Wörter, eine Folge von Wörtern bildet einen Satz:

$$\begin{aligned} S &= W_1 W_2 \ldots W_N \\ W_i &= c_{i,1} c_{i,2} \ldots c_{i,M_i} \qquad (c_{i,j} \in Z) \end{aligned}$$

Diese für den Menschen intuitiven Ebenen der Schrift müssen in der automatischen Handschrifterkennung nicht zwangsläufig eine Rolle spielen. So kann ein Satz auch direkt als Folge einzelner Zeichen definiert werden, ohne die Zwischenrepräsentation Wort zu verwenden:

$$S = c_1 c_2 \ldots c_N \qquad (c_i \in Z)$$

In diesem Fall würde eine Erkennung lediglich auf Zeichenebene stattfinden und beliebige Zeichenfolgen produzieren. Umgekehrt müssen nicht unbedingt einzelne Zeichen als Einheit verwendet werden. Hier wäre dann entweder das Wort die kleinste bedeutungstragende Einheit im Erkennungsprozeß.[1] Oder aber es werden andere Untereinheiten von Wörtern, z.B. aus mehreren Zeichen bestehende Silben, bei der Erkennung genutzt. In der Praxis erweist sich die intuitive Modellierung der Schrift durch Zeichen, Wörter und Sätze jedoch letztendlich als am zweckmäßigsten, weil nur so ein hierarchischer Ansatz für alle Erkennungsaufgaben realisierbar ist. Tatsächlich wird diese Modellierung in nahezu allen aktuellen Systemen zur Handschrifterkennung verwendet.

Bislang wurden lediglich die in Abbildung 2.1 dargestellten, mit Ausnahme des Punktes, bedeutungstragenden Ebenen der Schrift betrachtet. Die Erkennung kann

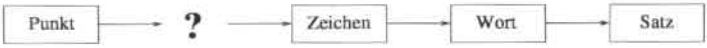


Abbildung 2.1: Modellierung der Handschrift

aber wesentlich vereinfacht werden, wenn auch Zeichen selbst wiederum durch – aus Sicht der Sprache – abstrakte Größen modelliert werden, die einzelne, zeitlich aufeinanderfolgende Punkte der Trajektorie zu einer größeren Einheit zusammenfassen. Eine Folge dieser Einheiten ergibt dann ein Zeichen. Gegebenenfalls kann auch mehr als eine Ebene zusätzlich zwischen Punkten und Zeichen eingefügt werden. Sind die Einheiten abstrakt aus Sicht der Sprache, da dort keine kleinere Unterteilung als einzelne Zeichen existiert, so können sie doch aus Sicht der Trajektorie der Schrift eine anschauliche Bedeutung haben. In der Spracherkennung gibt es die analoge Unterteilung von gesprochenen Buchstaben in die Phoneme. Es gibt eine Vielzahl von Unterteilungsmöglichkeiten, die in der Literatur nicht einheitlich benannt sind. In unterschiedlichen Systemen wird ein und derselbe Begriff für unterschiedliche Dinge verwendet. Eine häufig verwendeter Ansatz ist die Unterteilung der Trajektorie in sog. *Strokes*. Ein Stroke ist dabei z.B. definiert als ein Segment der Trajektorie zwischen zwei lokalen Extrema der Trajektorie oder anderweitig definierten Punkten wie Wendestellen und ähnlichem [KR89, ST92, Sch93, HBT96].

In dieser Arbeit wird eine auch aus Sicht der Trajektorie abstrakte Modellierung der Zeichen verwendet. Jedes Zeichen besteht hier aus einem Anfangs-, Mittel- und Endteil, wobei die Grenzen zwischen diesen Teilen automatisch bestimmt werden,

[1] Dieses Vorgehen entspricht dem weiter unten beschriebenen ganzheitlichen Erkennungsansatz und setzt die Verwendung eines Vokabulars voraus.

wie in Kapitel 6 beschrieben wird. Im Vorgriff auf nachfolgende Kapitel werden diese Zeichensegmente bereits hier als *Zustände* bezeichnet. Zu der bereits gezeigten Modellierungshierarchie wird also eine weitere Ebene mit den Zuständen s_i^j hinzugefügt:

$$\begin{aligned} S &= W_1 W_2 \ldots W_N \\ W_i &= c_{i,1} c_{i,2} \ldots c_{i,M_i} \qquad (c_{i,j} \in Z) \\ c_i &= s_i^1 s_i^2 s_i^3 \end{aligned}$$

Vollständig ergibt sich also für diese Arbeit die in Abbildung 2.2 dargestellte Hierarchie.

Die Modellierung von Zeichen durch eine Folge von Zuständen wurde bisher nur in rein auf Hidden Markov Modellen basierenden Handschrifterkennern verwendet. Mit dem in dieser Arbeit vorgestellten Ansatz wird diese Art der Modellierung erstmals auch in einer hybriden Architektur aus neuronalem Netzwerk und Hidden Markov Modellen eingesetzt.

Abbildung 2.2: In dieser Arbeit verwendete Modellierung der Handschrift

2.4 Erkennungsstrategien

Seit Beginn der Forschung auf dem Gebiet der Handschrifterkennung wurden zwei grundsätzliche Erkennungsprinzipen verfolgt: *ganzheitliche Ansätze* und *analytische Ansätze*. Im ersten Fall wird die Erkennung global auf der ganzen Repräsentation eines Wortes durchgeführt, d.h. kein Versuch unternommen, einzelne Zeichen oder andere Teile der Wörter individuell zu identifizieren. Im zweiten Fall werden Wörter nicht als Ganzes betrachtet, sondern als eine Sequenz kleinerer Einheiten, z.B. einzelner Zeichen. Die Erkennung wird nicht auf der Wortebene, sondern auf einer darunter liegenden Ebene auf Basis dieser kleineren Einheiten durchgeführt.

2.4.1 Ganzheitliche Ansätze

Ganzheitliche Ansätze folgen einem einfachen Schema: Im ersten Schritt erfolgt die Normalisierung der originalen Eingabe und Extraktion von Merkmalen, die für die Erkennung geeignet erscheinen. Der zweite Schritt beinhaltet die globale Erkennung durch Vergleich der erzeugten Repräsentation der unbekannten Eingabe mit der von gespeicherten Referenzen in einem Lexikon.

Diese Vorgehensweise führt zu zwei wichtigen praktischen Konsequenzen:

- Da eine Segmentierung der Eingabe in einzelne Zeichen vermieden und die Erkennung lediglich auf der globalen Ebene durchgeführt wird, sind diese Ansätze

sehr robust gegen die oftmals erheblichen Deformationen einzelner Zeichen, wie sie in freier, kursiver Handschrift zu beobachten sind.

- Die Erkennung ist zwangsläufig auf ein fest vorgegebenes Vokabular beschränkt, d.h. es können lediglich Wörter aus dem jeweiligen Vokabular erkannt werden.

Der zweite Punkt fällt um so mehr ins Gewicht, wenn ein Training des Erkenners auf Wortebene erforderlich ist. In diesem Fall kann das Vokabular lediglich durch einen Trainingsschritt modifiziert oder erweitert werden. Diese Eigenschaft beschränkt ganzheitliche Ansätze auf Anwendungen, in denen das Vokabular statisch definiert ist und nicht häufig geändert werden muß. Die Größe des Vokabulars wird dabei ggf. durch Speichergrenzen und Laufzeitanforderungen beschränkt. Je größer das Vokabular ist, desto mehr Referenzen müssen im Speicher gehalten und bei der Erkennung verglichen werden. Wegen der praktischen Einschränkungen ganzheitlicher Ansätze werden sie heute nur noch für einige wenige Spezialanwendungen in der Off-line Erkennung eingesetzt.

2.4.2 Analytische Ansätze

Bei analytischen Ansätzen erfolgt die Erkennung nicht auf der globalen Wortebene, sondern auf definierten Wortuntereinheiten. Diese Einheiten können im einfachsten Fall ganze Zeichen sein, jedoch auch beliebige andere Wortteile. In der Regel stehen die Untereinheiten in Relation zu ganzen Zeichen, um die Erkennung unabhängig von einem spezifischen Vokabular zu machen. D.h. jedes Wort ist als einfache Sequenz dieser Untereinheiten darstellbar. Das Vokabular kann in diesem Fall dynamisch definiert werden, und ein Training des Erkenners auf Wortebene ist nicht unbedingt erforderlich.

Ein wesentlicher Unterschied zwischen den einzelnen analytischen Ansätzen ist neben den verwendeten Erkennungseinheiten die Art, wie bei der Erkennung die Segmentierung der Eingabe in diese Einheiten erfolgt. Analytische Ansätze fallen dabei in zwei Kategorien [WSG+94]:

- analytische Ansätze mit expliziter Segmentierung (Eingabesegmentierung)
- analytische Ansätze mit implizierter Segmentierung (Ausgabesegmentierung)

Im ersten Fall erfolgt vor der Erkennung eine explizite Segmentierung der Eingabe in die Untereinheiten, die dann jeweils getrennt voneinander erkannt werden, wie in Abbildung 2.3 auf der Ebene von Zeichen dargestellt ist. Im zweiten Fall findet die Segmentierung und Erkennung zur selben Zeit statt.

2.4.3 Analytische Ansätze mit expliziter Segmentierung

Die Erkennung erfolgt bei diesen Ansätzen in drei aufeinanderfolgenden Schritten:

- **Segmentierung** der Eingabe in die jeweiligen Untereinheiten (z.B. einzelne Zeichen).

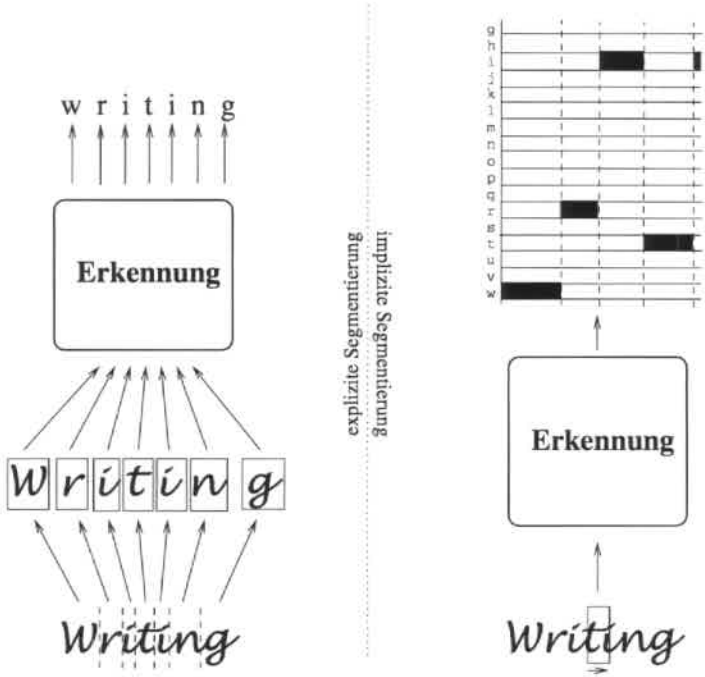


Abbildung 2.3: Explizite und implizite Segmentierung

- Unabhängige **Erkennung** der einzelnen Einheiten durch einen einfachen Einzelzeichenerkenner.
- **Kontextuelle Nachbearbeitung** unter Verwendung von lexikalischem, syntaktischem und ggf. semantischem Wissen.

Vorteil dieser Methodik ist, daß für die Erkennung lediglich ein Einzelzeichenerkenner benötigt wird, der unabhängig von allen anderen Komponenten entwickelt und trainiert werden kann. Dafür ist jedoch eine aufwendige Segmentierungskomponente erforderlich. Der größte Nachteil dieser Ansätze ist, daß aufgrund der frühen Segmentierungsentscheidung eine falsche oder ungenaue Segmentierung zu Fehlerkennungen führen kann.

Fehlerkennungen einzelner Wortteile können unter Umständen durch eine kontextuelle Nachbearbeitung korrigiert werden. Zur Verfügung stehen hierfür z.B. orthographische Korrekturtechniken, welche auf Statistiken über das Auftreten bestimmter Kombinationen aufeinanderfolgender Zeichen beruhen (z.B. N-Gramme, Hidden-Markov Modelle mit Viterbi-Algorithmus), oder direkte Vergleiche einer Erkennungshypothese mit allen Einträgen eines Vokabulars (z.B. Editierdistanzen, DP-

Matching) [WF74]. Im zweiten Fall ist die Erkennung wiederum auf ein vorgegebenes Vokabular eingeschränkt, während im ersten Fall beliebige Wörter der jeweiligen Sprache erkannt werden können. Die statistischen Verfahren sind zwar in der Regel bei großen Vokabularen erheblich schneller, produzieren aber unter Umständen nicht existierende Wörter.

Zur expliziten Segmentierung ist folgendes festzustellen: Um ein optimales Erkennungsergebnis zu erzielen, müßten die Zeichengrenzen exakt bestimmt werden. Die Zeichgrenzen ließen sich jedoch weitaus genauer bestimmen, wenn bekannt wäre, was geschrieben wurde, d.h. das Erkennungsergebnis bereits vorläge.

2.4.4 Analytische Ansätze mit impliziter Segmentierung

Analytische Ansätze mit impliziter Segmentierung verlagern das Problem der Segmentierung, sofern in manchen Fällen überhaupt von Segmentierung gesprochen werden kann, soweit wie möglich an das Ende des Erkennungsprozesses. Erkennung und Segmentierung finden gleichzeitig statt, die Segmentierung ist üblicherweise ein Nebenprodukt der Erkennung. Nur in bestimmten Fällen, z.B. wenn für ein späteres Training unbekannte Daten mit den Zeichengrenzen versehen werden sollen, sind die Segmentierungsinformationen überhaupt von Interesse, da lediglich das Erkennungsergebnis benötigt wird. Fehler durch zu frühe Segmentierungsentscheidungen werden hierdurch vermieden, darüber hinaus kann auf aufwendige Segmentierungsalgorithmen verzichtet werden. Syntaktisches Wissen über erlaubte Zeichenfolgen findet auch hier wieder in Form statistischer Methoden oder direkt als Vokabular seinen Einsatz, um Fehlerkennungen zu reduzieren [Sen95]. Diesem Ansatz entspricht auch die Architektur des in dieser Arbeit vorgestellten Erkennungssystems.

2.5 Neuronale Netze

Seit der Definition eines ersten neuronalen Modells von McCulloch und Pitts im Jahr 1943 [MP43] haben sich künstliche neuronale Netze als ein effektives Werkzeug für die Mustererkennung erwiesen. Neuronale Netze haben daher sowohl in der On-line als auch Off-line Schrifterkennung eine lang zurückreichende Tradition. Noch immer bilden sie in vielen heutigen Systemen die Basis für die Erkennung (siehe z.B. [KR91, GAL+91, MBLD92, MBPV93, Sch95, Sen95]). Hinter dem Einsatz neuronaler Netze steckt die Idee, die kognitiven Fähigkeiten des menschlichen Gehirns mit seinen schätzungsweise über 10^{11} Neuronen (Nervenzellen), welche bestimmte Informationen über eine vorgegebene Verbindungsstruktur an nachfolgende Neuronen weitergeben, zumindest ansatzweise nachzubilden. Mittlerweile hat sich eine Vielzahl neuronaler Netzarchitekturen entwickelt, die sich u.a. hinsichtlich der Verbindungsstruktur (vollständige Verbindung, modulare Verbindung, rekurrente Verbindung), der Transferfunktion (Sigmoid, Tangens Hyperbolicus, Radial-Basis Funktionen) oder des verwendeten Lernalgorithmus (überwachtes oder unüberwachtes Lernen) unterscheiden. In dieser Arbeit kommt ein mehrschichtiges Netz zum Einsatz, das mit

dem Backpropagation-Verfahren trainiert wird. Dieses Netz berechnet die a posteriori Wahrscheinlichkeit für einen zeitlich begrenzten Bereich der Eingabe, daß dieser Bereich ein Zeichen bzw. Teil eines Zeichens einer bestimmten Ausgabeklasse enthält. Dieser Abschnitt gibt daher einen kurzen Einblick in die generelle Funktionsweise und die Eigenschaften mehrschichtiger Netze.

2.5.1 Geschichtliche Entwicklung

Das 1943 von McCulloch und Pitts vorgeschlagene mathematische Modell eines Neurons ist ein *logisches Schwellwertelement* mit zwei möglichen Zuständen (siehe Abbildung 2.4). Dieses Schwellwertelement oder Neuron besitzt M Eingangsleitungen x_i und eine Ausgangsleitung y. Eine Eingangsleitung ist entweder aktiv (Eingabe „1“) oder ruhig (Eingabe „0“). Die Aktivitäten aller Eingangsleitungen kodieren die Eingangsinformation daher als binäres Muster aus M Bit. Der Zustand des Neurons ergibt daraus durch gewichtete Summation aller Eingabesignale x_i und Vergleich der Summe mit einem Schwellwert s. Überschreitet die Summe den Schwellwert, so ist das Neuron „erregt“, andernfalls befindet es sich im Ruhezustand. Der erregte Zustand und Ruhezustand sollen dem Feuern bzw. Nichtfeuern eines Aktionspotentials biologischer Neuronen entsprechen und werden im Modell durch die binären Werte Null und Eins für die Aktivität der Ausgabeleitung repräsentiert. Die Ausgabe y eines Neurons ist also definiert durch

$$y = f(\sum_i w_i x_i - s),$$

wobei für die Schwellwertfunktion f gilt $f(x) = 1$ für $x \geq 0$ und $f(x) = 0$ für $x < 0$. McCulloch und Pitts zeigten, daß jede beliebige logische Funktion durch geeignete Kombination derartiger Schwellwertelemente realisiert werden kann.

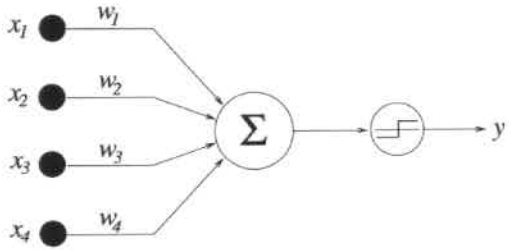


Abbildung 2.4: Das McCulloch-Pitts Neuron von 1943

Den nächsten Meilenstein in der Historie neuronaler Netze bildete 1962 das von Rosenblatt vorgeschlagene *einschichtige Perzeptron* [Ros58, Ros62]. Es besteht aus einer gegebenen Anzahl N von Neuronen, denen über M Eingänge Eingabemuster zugeführt werden (Abbildung 2.5). Jedes Eingabemuster wird durch einen M-komponentigen Merkmalsvektor $x = (x_1, x_2, \ldots, x_M)^T$ beschrieben und gehört zu

einer von N Musterklassen. Die Klassifizierung der Eingabemuster und die genaue Festlegung der Anzahl und Bedeutung der Komponenten x_i werden durch die Anwendungsaufgabe bestimmt. So kann es sich bei einem Off-line Ziffernerkenner beispielsweise um Grauwertpixel der Ziffern handeln. Das Perzeptron lernt die korrekte Klassifikation der Mustervektoren anhand bekannter Klassifikationsbeispiele im Laufe einer Trainingsphase. Zur Klassifikation eines Musters x berechnet jedes Neuron n einen binären Ausgabewert y_n gemäß

$$y_n = f(\sum_{i=1}^{M} w_{ni} x_i)$$

Die Koeffizienten w_{ni} bestimmen das Verhalten des Neurons n und werden im Laufe der Trainingsphase so angepaßt, daß das Neuron n nur auf die Eingabemuster seiner zugeordneten Klasse mit einem Ausgabewert $y_n = 1$ reagiert.

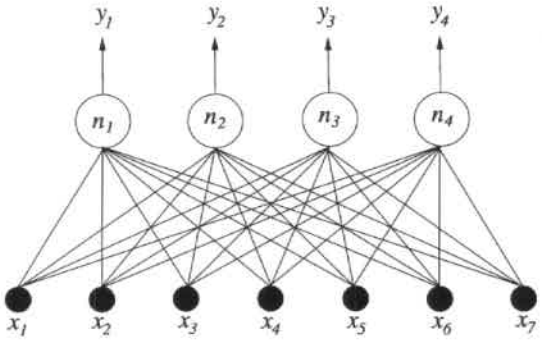


Abbildung 2.5: Das Perzeptron von Rosenblatt 1962

Einen Rückzug vieler Forschergruppen und insbesondere auch Förderer aus der Forschung über neuronale Netze lösten 1969 Minsky und Papert mit ihrer Feststellung aus, daß das von Rosenblatt vorgeschlagene Perzeptron viele wichtige Probleme, die nicht linear separierbar sind, gar nicht repräsentieren kann [MP69]. Dies zeigten sie anhand einiger sehr einfacher Probleme, wie z.B. des XOR- oder „Parity"-Problems, die das Perzeptron nicht zu lösen vermag.

Erst Anfang der achtziger Jahre, also nach fast 15 Jahren Stagnation, folgte ein neuerlicher Aufschwung neuronaler Netze. Dieses erneute Interesse ist hauptsächlich auf die Entwicklung und weite Publikation des Lernverfahrens *Backpropagation*, eines Gradientenabstiegsverfahrens für *mehrschichtige Perzeptrone* (multi-layer perceptron, MLP) im Jahr 1986 zurückzuführen [RHW86]. Mittlerweile hat sich das MLP als erfolgreichste Netzarchitektur durchgesetzt.

2.5.2 Mehrschichtige neuronale Netze

Ein MLP ergibt sich durch die Hintereinanderschaltung mehrerer Neuronenschichten (Abbildung 2.6), wodurch im Gegensatz zum einschichtigen Perzeptron auch die Modellierung von komplexen, nicht-linearen Entscheidungsebenen möglich wird. Jedes Neuron einer Schicht des MLP ist mit allen Neuronen der nachfolgenden Schicht verbunden. Die erste Schicht nimmt das Eingabemuster auf und wird daher als Eingabeschicht bezeichnet. Die letzte Schicht, deren Aktivitätsmuster die Ausgabe des MLP bildet, ist die Ausgabeschicht. Alle Schichten zwischen der Eingabe- und Ausgabeschicht sind sog. verborgene Schichten. Die Neuronen dieser verborgenen Schichten besitzen also keine direkten Eingabe- und Ausgabeleitungen. Ein Neuron der verbor-

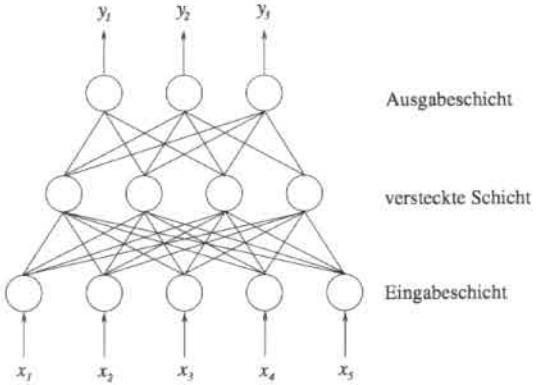


Abbildung 2.6: Aufbau eines mehrschichtigen neuronalen Netzes

genen Schichten hat als Aktivität

$$y_i = f(\sum_j w_{ij} y_j),$$

wobei y_j die Ausgabeaktivitäten der Neuronen der vorhergehenden Schicht sind. $f(x)$ ist eine typischerweise sigmoide Transferfunktion, d.h. $f(x)$ ist nicht-negativ, überall monoton steigend und strebt für $x \to \pm\infty$ gegen asymptotische Sättigungswerte. Häufig wird die Sigmoid-Funktion

$$f(x) = \frac{1}{1 + e^{-x}}$$

als Transferfunktion gewählt. Der Aktivitätszustand der verborgenen Neuronen ist nur mittelbar über die interne Verschaltung des Netzes zu beeinflussen, kann also nicht durch die „Außenwelt" festgelegt werden, da keine direkten Verbindungen nach außen existieren. Lange Zeit war nicht klar, wie die Parameter w_{ij} durch Lernen so eingestellt werden können, daß das Netz für eine bestimmte Eingabe die gewünschte Transformation in ein Ausgabesignal durchführt. Das Backpropagation-Verfahren bietet hierfür eine Lösung.

2.5.3 Lernen durch Gradientenabstieg

Die Lernaufgabe besteht darin, die Gewichtsmatrix W der Gewichte w_{ij} des MLP so einzustellen, daß die Muster der gegebenen Trainingsmenge möglichst optimal klassifiziert werden. Bei der allgemein üblichen 1-aus-N Kodierung gehört jedes Muster der Trainingsmenge zu einer eindeutigen Klasse. Jedes der N Neuronen der Ausgabeschicht ist fest einer dieser Klassen zugeordnet. Wird ein Muster einer bestimmten Klasse durch das Netz propagiert, sollte das zugehörige Ausgabeneuron eine möglichst hohe Aktivierung erreichen, die übrigen $N-1$ Neuronen entsprechend eine möglichst geringe. Während der Lernphase des MLP werden für jedes Trainingsmuster die aktuellen Aktivierungen y_i der Ausgabeneuronen dem zugehörigen Sollwert $d_i \in \{0,1\}$ gegenübergestellt und die Abweichung des Ist-Wertes vom Sollwert festgestellt. Der Sollwert d_i hat den Wert 1, sofern das angelegte Muster zur Klasse i gehört, ansonsten den Wert 0. Ein Maß dafür, wie gut das MLP die Aufgabe der Zuordnung der Ausgabemuster y zu den Eingabemustern x erfüllt, ist der Fehler

$$\sum_{n=1}^{N} \mathrm{E}(y_n - d_n)$$

über alle Musterpaare (y_n, d_n) mit der Fehlerfunktion E. Die bekannteste und am häufigsten verwendete Fehlerfunktion ist der **quadratische Fehler** (Mean Squared Error, MSE)

$$E_{\mathrm{MSE}} = \frac{1}{2}\sum_j (y_j - d_j)^2,$$

der hier der Einfachheit halber nur für ein Muster angegeben ist. Für einen Satz vorgegebener Musterpaare (y_p, d_p) ist E eine Funktion aller Gewichte W. Im Hinblick auf die zu lösende Aufgabe sind die Gewichte W also optimal gewählt, wenn der Fehler E minimal ist. Die Bestimmung geeigneter Gewichte ist dementsprechend äquivalent zur Minimierung der Fehlerfunktion E.

Die Suche nach einem Minimum der Fehlerfunktion E erfolgt mit dem Backpropagation-Algorithmus, einem Gradientenabstiegsverfahren, das auf dem Prinzip des steilsten Abstiegs beruht. Gradientenverfahren berechnen den Gradienten der Fehlerfunktion und steigen entgegengesetzt zum Gradienten nach unten, bis ein Minimum erreicht ist. Hierbei wird versucht, den Fehler dadurch zu minimieren, daß eine Änderung aller Gewichte um einen Bruchteil des negativen Gradienten der Fehlerfunktion vorgenommen wird:

$$\Delta W = -\eta \nabla E(W)$$

Die Änderung ΔW der Gewichtsmatrix ist proportional zum negativen Gradienten $-\nabla E(W)$ der Fehlerfunktion mit dem Lernfaktor η (auch Schrittweite genannt). Für ein einzelnes Gewicht gilt somit

$$\Delta w_{ij} = -\eta \frac{\partial}{\partial w_{ij}} E(W)$$

Um ein zu starkes Springen der Gewichtsänderungen in verschiedene Richtungen zu verringern, kann bei der Berechnung von ΔW zusätzlich ein sog. Momentum hinzugenommen werden, das einen Anteil des zuvor gemachten Schrittes berücksichtigt. Das Gradientenabstiegsverfahren findet nicht notwendigerweise das globale Minimum der Fehlerfunktion, sondern kann ggf. in einem lokalen Minimum stecken bleiben.

2.5.4 Alternative Fehlerfunktion

Wie bereits erwähnt, ist der quadratische Fehler das bekannteste und am häufigsten eingesetzte Fehlermaß. Vorteil dieses Fehlermaßes ist, daß große Abweichungen des Ist-Wertes y_j vom Sollwert d_j stärker berücksichtigt werden als kleine Abweichungen. Probleme mit dem MSE treten jedoch dann auf, wenn eine große Anzahl von Ausgabeneuronen und ein 1-aus-N Klassifikationsproblem vorliegen. In diesem Fall erzeugt der Wert 0 für alle Ausgabeneuronen bereits einen sehr kleinen Fehler, so daß der Lernimpuls zu gering ist für das Netz, um den Sollwert zu erlernen. Dieses Problem wird z.B. durch die aus der Informationstheorie stammenden **Cross Entropy**, die binäre Sollwerte voraussetzt, vermieden:

$$E_{\mathrm{CE}} = -\sum_j [d_j \log(y_j) + (1-d_j)\log(1-y_j)] = \sum_j \begin{cases} -\log(y_j) & \text{für } d_j = 1 \\ -\log(1-y_j) & \text{für } d_j = 0. \end{cases}$$

Hier werden Abweichungen der Ausgabe y_j vom Sollwert d_j überproportional bestraft, so daß auch bei einer großen Anzahl von Ausgabeneuronen ein ausreichend großer Lernimpuls erzeugt wird. Die Cross Entropy wurde in allen Experimenten dieser Arbeit als Fehlermaß verwendet.

2.5.5 Eigenschaften mehrschichtiger Netze

Zwei wesentliche Eigenschaften mehrschichtiger neuronaler Netze machen sie interessant für die Handschrifterkennung. Zum einen können bereits mit einem MLP mit nur einer verborgenen Schicht stetige Funktionen beliebig genau approximiert werden, sofern die Anzahl verborgener Neuronen nur groß genug gewählt wird [HSW89]. Zum anderen entsprechen die Ausgaben eines hinreichend komplexen MLPs bei einer 1-aus-N Repräsentation gerade den a posteriori Klassenwahrscheinlichkeiten der Trainingsmuster [RL91]. Dies gilt, wenn während des Trainings ein ausreichend kleiner Fehler auf den Trainingsdaten erzielt wird und ausreichend viele, entsprechend der a priori Klassenzugehörigkeit verteilte Trainingsmuster zur Verfügung stehen. Dabei können während des Trainings sowohl die MSE Fehlerfunktion als auch die Cross Entropy und andere Standard-Fehlerfunktionen verwendet werden [IP90].

2.6 Hidden Markov Modelle

Hidden Markov Modelle (HMM) haben sich seit den achtziger Jahren zu einem wichtigen und erfolgreichen Werkzeug in der Spracherkennung entwickelt [Rab89, WL90].

Gerade dieses Anwendungsgebiet hat die Entwicklung der Theorie der HMMs in den letzten Jahren stark vorangetrieben. Der Erfolg von HMMs begründet sich unter anderem auf ihre Fähigkeit, sequentielle Daten mit zeitlicher Variabilität zu verarbeiten. Mittlerweile kommen sie auch in der Handschrifterkennung entweder alleine oder in Verbindung mit neuronalen Netzen mehr und mehr zum Einsatz [NWF86, HOKK93, NBNB93, BL93, FNCB93, BBNN94, SMSC94, MSSC94, KAM+94, Sch95, DHU97]. In dieser Arbeit werden HMMs zur hierarchischen Modellierung von Einzelzeichen, Wörtern und Sätzen eingesetzt (siehe Abschnitt 2.6.5).

Jedes zu erkennende Wort wird im HMM durch ein abstraktes, statistisches Modell repräsentiert, dessen Gewichte automatisch anhand einer entsprechenden Datenbasis erlernt werden können. Gesucht ist das Wort w_i aus einem Vokabular $W = \{w_1, w_2, \ldots\}$, das für eine gegebene handschriftliche Eingabe die höchste a posteriori Wahrscheinlichkeit hat:

$$P(w_i|X) = \frac{P(X|w_i)P(w_i)}{P(X)}$$

Die a priori Wahrscheinlichkeit $P(w_i)$ wird durch das Sprachmodell bestimmt, P(X) leistet aufgrund seiner Unabhängigkeit von w_i keinen Beitrag zur Entscheidung. Aufgabe des HMM ist es, die Verteilungsdichte $P(X|w_i)$ für die gegebene handschriftliche Repräsentation X des Wortes w_i durch $P(X|\lambda_i)$ zu modellieren. $P(X|\lambda_i)$ wird in einem zweistufigen Zufallsprozeß berechnet: Zunächst wird in jedem Zeitschritt ein interner Zustand des HMMs bestimmt und anschließend davon abhängig die Wahrscheinlichkeit für einen Ausgabevektor.

2.6.1 Definition des HMM

Ein HMM λ ist vollständig definiert durch eine Menge von N Zuständen $S = \{S_1, S_2, \ldots, S_N\}$, ein Beobachtungsalphabet $V = \{v_1, v_2, \ldots, v_M\}$ mit M Symbolen, die $N \times N$ Matrix $A = (a_{ij})$ der *Übergangswahrscheinlichkeiten*, die Matrix $B = (b_j(k))$ der *Emissionswahrscheinlichkeiten* in einem Zustand j und einen *Startzustand* $\pi = \{\pi_i\}$. Üblicherweise wird die kompakte Notation

$$\lambda = (A, B, \pi)$$

zur Angabe der vollständigen Parameter eines HMMs verwendet. Die Zustände S_i werden in einem diskreten stochastischen Prozeß durchlaufen, so daß die Zustandsfolge $q = q_1 q_2 \ldots q_T$ mit $q_t \in S$ entsteht. Der initiale Zustand q_1 des Modells wird gemäß der Wahrscheinlichkeitsverteilung π bestimmt:

$$\pi_i = P(q_1 = S_i) \qquad \text{für } i = 1 \ldots N$$

Die Matrix der Übergangswahrscheinlichkeiten A legt den Wechsel von einem Zustand q_t in den nächsten Zustand q_{t+1} fest:

$$a_{ij} = P(q_t = S_j|q_{t-1} = S_i) \qquad \text{für } i, j = 1 \ldots N$$

Für den Fall, daß jeder Zustand innerhalb eines Schrittes von jedem anderen Zustand erreicht werden kann, wie an einem einfachen Beispiel in Abbildung 2.7a dargestellt, ist $a_{ij} > 0$ für alle i, j. Für einfache Links-Rechts Modelle, wie sie in der Sprach- und

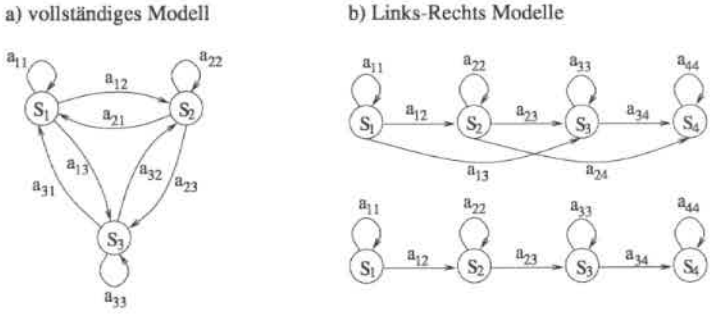


Abbildung 2.7: Ein vollständig verbundenes HMM (a) und einfache Links-Rechts Modelle (b)

Handschrifterkennung üblicherweise verwendet werden, gilt

$$\begin{aligned} a_{ij} &= 0 \quad \text{für } j < i \text{ und} \\ a_{ij} &= 0 \quad \text{für } j > i + \Delta \end{aligned}$$

D.h. es kann von einem Zustand nicht in einen bereits früher erreichten Zustand zurückgekehrt werden. Der Parameter Δ spezifiziert in diesem Fall die maximale Sprungweite eines Übergangs. In dem oberen Links-Rechts Modell von Abbildung 2.7b wurde $\Delta = 2$ gewählt, in dem unteren Modell $\Delta = 1$. Diese Modelle sind insbesondere dazu geeignet, Signale zu repräsentieren, deren Eigenschaft sich über die Zeit hinweg ändert.

Der durch A und π charakterisierte Prozeß wird *Markovkette* genannt. Sie bildet die erste Stufe des zweistufigen Zufallsprozesses. In der zweiten Stufe wird in Abhängigkeit des aktuellen Zustands q_t die Ausgabe $o_t = v_k$ generiert, wobei v_k ein Symbol des Beobachtungsalphabets V ist. Die Matrix der Emissionswahrscheinlichkeiten $B = (b_j(k))$ mit

$$b_j(k) = P(o_t = v_k | q_t = S_j) \qquad \text{für } j = 1 \ldots N, k = 1 \ldots M$$

spezifiziert die Wahrscheinlichkeit, in der Folge $O = o_1 o_2 \ldots o_T$ die Ausgabe $o_t = v_k$ zu beobachten.

In der Praxis wird häufig vereinfachend angenommen, daß die nächste Entscheidung nur auf dem unmittelbar vorhergehenden Zustand getroffen wird:

$$\begin{aligned} P(q_t | q_1 \ldots q_{t-1}, o_1 \ldots o_{t-1}) &= P(q_t | q_{t-1}) \\ P(o_t | q_1 \ldots q_t, o_1 \ldots o_{t-1}) &= P(q_t | q_t) \end{aligned}$$

In diesem Fall wird von einem Hidden Markov Modell erster Ordnung gesprochen. Darüber hinaus ist der beschriebene Zufallsprozeß stationär, d.h. der absolute Zeitpunkt t spielt für die jeweilige Entscheidung keine Rolle.

Sind nun die Zustände S, das Beobachtungsalphabet V und $\lambda = (A, B, \pi)$ gegeben, kann das HMM mit folgendem Algorithmus eine Beobachtungssequenz $O = o_1 o_2 \dots o_T$ generieren:

1. **Schritt 1:** Wähle gemäß der Wahrscheinlichkeitsverteilung π einen initialen Startzustand $q_1 = S_i$.
2. **Schritt 2:** Setze $t = 1$.
3. **Schritt 3:** Wähle ein Symbol $o_t = v_k$ gemäß den Emissionswahrscheinlichkeiten $b_i(k)$ des aktuellen Zustands q_t.
4. **Schritt 4:** Gehe gemäß den in A gegebenen Übergangswahrscheinlichkeiten $a_{q_t, q_{t+1}}$ vom aktuellen Zustand q_t in einen neuen Zustand q_{t+1} über.
5. **Schritt 5:** Setze $t = t + 1$; wenn $t < T$ ist, gehe zurück zu Schritt 3.

Dieser einfache Algorithmus kann zum einen als Generator von Beobachtungen und zum anderen als Modell dafür eingesetzt werden, wie eine gegebene Beobachtungssequenz durch ein entsprechendes HMM erzeugt wurde.

Drei grundlegende Probleme müssen gelöst werden, um Hidden Markov Modelle der obigen Form für den praktischen Einsatz nutzbar zu machen:

1. Wie kann die Wahrscheinlichkeit $P(O|\lambda)$ einer bestimmten Beobachtungssequenz $O = o_1 o_2 \dots o_T$ bei gegebenem Modell $\lambda = (A, B, \pi)$ effizient berechnet werden?
2. Wie kann bei gegebenem Modell λ die Zustandsfolge $Q = q_1 q_2 \dots q_T$ bestimmt werden, die die Beobachtungsfolge $O = o_1 o_2 \dots o_T$ am wahrscheinlichsten verursacht hat?
3. Wie können die Parameter des Modells $\lambda = (A, B, \pi)$ so eingestellt werden, daß sie $P(O|\lambda)$ maximieren?

Lösungen für diese Probleme stellen die folgenden Abschnitte in Form der Vorwärts-, Viterbi- und Baum-Welch-Algorithmen vor.

2.6.2 Der „Vorwärts"-Algorithmus

Es soll nun für die Beobachtungssequenz $O = o_1 o_2 \dots o_T$ die Wahrscheinlichkeit $P(O|\lambda)$ bestimmt werden, daß O durch das Modell λ erzeugt wurde. Dabei kann O durch mehrere unterschiedliche Zustandsfolgen $q \in S^T$ erzeugt worden sein. Setzt man eine konkrete Zustandsfolge $Q = q_1 q_2 \dots q_T$ voraus, kann $P(O|\lambda, Q)$ durch Aufmultiplizieren der Übergangs- und Emissionswahrscheinlichkeiten entlang von Q berechnet

werden:

$$P(O|Q,\lambda) = \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1},q_t} b_{q_t}(o_t)$$

Die Wahrscheinlichkeit $P(O|\lambda)$ ist damit die Summe der Wahrscheinlichkeiten aller möglichen Pfade $q \in S^T$:

$$P(O|\lambda) = \sum_{q \in S^T} P(O|q,\lambda) = \sum_{q \in S^T} \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1},q_t} b_{q_t}(o_t)$$

Da die Anzahl möglicher Zustandsfolgen und damit der Berechnungsaufwand $O(TN^T)$ für obige Formel exponentiell mit der Länge T der Beobachtungssequenz wachsen, ist diese Art der Berechnung für den praktischen Einsatz nicht nutzbar. Mit dem sog. *Vorwärtsalgorithmus*, der auf dem Prinzip der dynamischen Programmierung beruht, steht jedoch ein effizienterer Algorithmus zur Verfügung. Hierfür werden die Vorwärtsvariablen $\alpha_t(i)$ definiert als

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = S_i|\lambda) \text{ für } t = 1 \ldots T, i = 1 \ldots N,$$

die wie folgt rekursiv mit Aufwand $O(T)$ berechnet werden können:

$$\alpha_t(j) = \begin{cases} \pi_j b_j(o_1) & \text{für } t = 1 \\ \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) & \text{für } t > 1 \end{cases}$$

Mit dieser Definition ergibt sich die gesuchte Wahrscheinlichkeit $P(O|\lambda)$ als

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Eine andere Sichtweise der Berechnung von $P(O|\lambda)$ ist, eine Bewertung dafür zu finden, wie gut ein gegebenes Modell auf die Beobachtungssequenz paßt. Diese Sichtweise ermöglicht es z.B., bei mehreren konkurrierenden Modellen dasjenige Modell auszuwählen, welches zu der Beobachtung am besten paßt.

2.6.3 Der Viterbi-Algorithmus

Das Problem, bei gegebenem Modell λ die „optimale" Zustandsfolge $Q = q_1 q_2 \ldots q_T$ zu bestimmen, die am wahrscheinlichsten die Beobachtungssequenz $O = o_1 o_2 \ldots o_T$ verursacht, ist nicht eindeutig lösbar. Die Lösung hängt jeweils von der Definition des Optimalitätskriteriums ab. Das in der Sprach- und Handschrifterkennung am häufigsten verwendete Optimalitätskriterium ist, diejenige Zustandsfolge Q (auch Pfad genannt) zu bestimmen, die $P(Q|O,\lambda)$ maximiert:

$$Q = \operatorname*{argmax}_{q \in S^T} P(q|O,\lambda) = \operatorname*{argmax}_{q \in S^T} P(O,q|\lambda)$$

Q kann mit dem *Viterbi-Algorithmus*, einer wiederum auf dynamischer Programmierung basierenden Variation des Vorwärtsalgorithmus, berechnet werden. Anstelle der

partiellen Gesamtwahrscheinlichkeit $\alpha_j(t)$ wird hier die beste Bewertung entlang eines Pfades zum Zeitpunkt t berechnet

$$\delta_t(i) = \max_{q \in S^{T-1}} P(o_1 o_2 \dots o_t, q_1 q_2 \dots q_{t-1}, q_t = S_j | \lambda)$$

und in der Rekursion der Definition des Vorwärtsalgorithmus die Summe durch eine Maximumbildung ersetzt, so daß:

$$\delta_t(j) = \begin{cases} \pi_j b_j(o_1) & \text{für } t = 1 \\ \max_{i=1}^{N} \delta_{t-1}(i) a_{ij} b_j(o_t) & \text{für } t > 1 \end{cases}$$

Mit dieser Definition erhält man mit

$$P(O, q|\lambda) := P^*(O|\lambda) = \max_{j=1}^{N} \delta_T(j)$$

eine meist ausreichende Näherung für $P(O|\lambda)$.

2.6.4 Der Baum-Welch-Algorithmus

Trainingsmethoden für HMMs basieren üblicherweise auf dem Maximum-Likelihood Kriterium. Die Parameter des Modells werden, ausgehend von initialen Parametern, anhand einer Trainingsmenge iterativ so eingestellt, daß mit jeder Iteration die Wahrscheinlichkeit steigt, daß die Trainingsmenge durch dieses Modell erzeugt wurde. Ist $\lambda = (A, B, \pi)$ ein Modell für ein Wort w und O ein Trainingsmuster für w, so sind die Parameter λ^* gesucht, die die Produktionswahrscheinlichkeit von O maximieren:

$$\lambda^* = \operatorname*{argmax}_{\lambda} P(O|\lambda)$$

Eine analytische Lösung für die Berechnung der optimalen Parameter λ ist aufgrund des zweistufigen Zufallsprozesses nicht möglich. Tatsächlich gibt es kein optimales Verfahren, welches für eine endliche Beobachtungssequenz die Modellparameter bestimmt. Mit dem *Baum-Welch-* oder *forward-backward*-Algorithmus existiert jedoch ein Verfahren, das durch iterative Schätzung zumindest ein lokales Minimum berechnet. Ausgehend von den Parametern $\lambda = (A, B, \pi)$ schätzt dieses Verfahren aus den Trainingsdaten neue Parameter $\hat{\lambda} = (\hat{A}, \hat{B}, \hat{\pi})$, für die gilt, daß entweder $\lambda = \hat{\lambda}$ oder $P(O|\hat{\lambda}) > P(O|\lambda)$ ist. Um $\hat{\lambda}$ zu bestimmen, werden die im Training durchlaufenen Zustände und Zustandsübergänge gezählt und daraus mit den in [Rab89] vorgestellten Schätzverfahren neue Werte für $\hat{A}$, $\hat{B}$ und $\hat{\pi}$ berechnet.

2.6.5 Hidden Markov Modelle in der Handschrifterkennung

In Abschnitt 2.3 wurde dargelegt, wie Handschrift hierarchisch auf unterschiedlichen Ebenen modelliert werden kann. Diese Modellierung wird nun mit HMMs entsprechend ausgedrückt. Sie stellen die Grammatik dar, die angibt, wie Zeichen, Wörter und Sätze jeweils aus den nächstkleineren Einheiten aufgebaut werden. Die HMMs

in Abbildung 2.8a modellieren einzelne Zeichen durch eine Sequenz dreier abstrakter Zustände wie in dieser Arbeit verwendet. Bei der Erkennung eines dieser Zeichen müssen jeweils alle drei Zustände dieser Zeichen-HMMs durchlaufen werden. Wörter der Sprache werden entweder durch eine Konkatenation der HMMs der einzelnen Zeichen dieses Wortes gebildet (Abbildung 2.8b) oder durch einfache Rückkopplung der Zeichen-HMMs, wenn bei der Erkennung kein Vokabular verwendet wird (Abbildung 2.8c). Für die Satzerkennung ist schließlich nur noch eine zusätzliche Rückkopplung vom Ende der Wort-HMMs zurück zu den Wortanfängen nötig (Abbildung 2.8d). In dieser Arbeit werden die Emissionswahrscheinlichkeiten für die Zustände der verwendeten HMMs aus den a posteriori Ausgabewahrscheinlichkeiten eines neuronalen Netzes bestimmt.

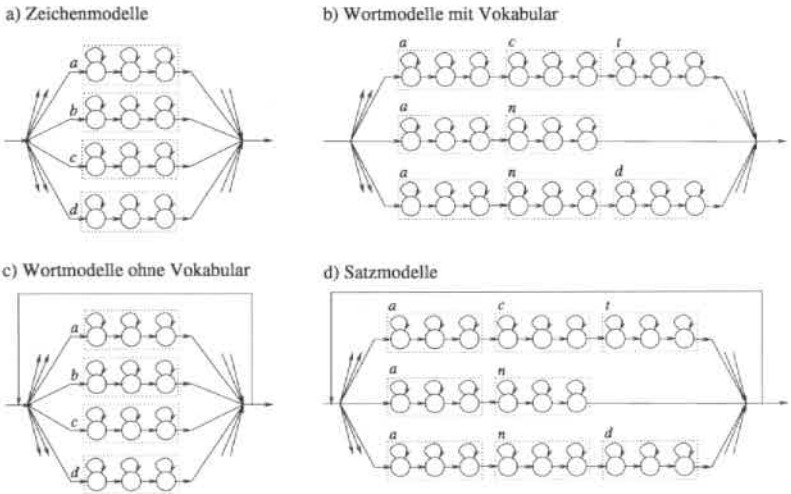


Abbildung 2.8: Modellierung der Schrift mit Hidden Markov Modellen

2.7 Sprachmodelle

Sprachmodelle stellen die a priori Wahrscheinlichkeiten $P(W)$ für Wortfolgen $W = w_1 w_2 \ldots w_n$ (bzw. $P(C)$ für Zeichenfolgen $C = c_1 c_2 \ldots c_n$ wenn die Einzelworterkennung ohne Vokabular erfolgt) bei der Maximierung des Ausdrucks $P(X|W) \cdot P(W)$ für eine gegebene Eingabe X zur Verfügung. $P(X|W) \cdot P(W)$ ist die gesuchte Ausgabe des Erkennungssystems. Die Verwendung statistischer Sprachmodelle ist die am weitesten verbreitete Technik der Sprachmodellierung, insbesondere auf dem Gebiet der kontinuierlichen Spracherkennung. Da in der Handschrifterkennung das gleiche grundlegende Erkennungsproblem zu lösen ist, kommen sie auch hier mittlerweile

häufig zum Einsatz [SNBK93]. Statistische Sprachmodelle bestimmen Wahrscheinlichkeiten für Wortübergänge aus möglichst großen Textkorpora, im Gegensatz zu linguistischen Ansätzen, die versuchen, syntaktisches und semantisches Wissen über die Sprache explizit durch Grammatiken zu modellieren.

Die a priori Wahrscheinlichkeit $P(W)$ ist unabhängig von der Eingabe X und kann daher isoliert betrachtet werden. Problematisch ist die Bestimmung der $P(W)$ nur dann, wenn sie für ganze Sätze geschätzt werden sollen. Im Falle eines Einzelworterkenners muß lediglich für jedes einzelne Wort w_i die a priori Wahrscheinlichkeit $P(w_i)$ ermittelt werden. Bei der Erkennung von Wortfolgen hingegen ergibt sich $P(W)$ für $W = w_1 w_2 \dots w_n$ aus

$$P(w_1 w_2 \dots w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1 w_2) \cdots P(w_n|w_1 w_2 \dots w_{n-1})$$

Die einzelnen Faktoren enthalten zu viele Variablen, um auf verfügbaren Textmengen geschätzt werden zu können. Daher beschränkt man sich in der Regel auf sog. N-Gramme. Diese beruhen auf der Annahme, daß in $P(w_i|w_1 w_2 \dots w_{i-1})$ mit einem zunehmenden Abstand eines Wortes w_j vom Wort w_i dessen Einfluß auf w_i sinkt. Umgekehrt haben die unmittelbar vorangehenden Wörter w_{i-1} und w_{i-2} einen stärkeren Einfluß auf w_i. N-Gramme tragen diesem Umstand dadurch Rechnung, daß anstelle des vollständigen Kontextes $w_1 w_2 \dots w_{i-1}$ nur der eingeschränkte Kontext $w_{i-N+1} \dots w_{i-1}$ betrachtet wird. $P(w_i|w_{i-N+1} \dots w_{i-1})$ gibt also die Wahrscheinlichkeit für ein Wort w_i an, unter der Voraussetzung, daß unmittelbar vorher die $N-1$ Wörter $w_{i-N+1} \dots w_{i-1}$ beobachtet worden sind. Für $P(w_1 w_2 \dots w_n)$ ergibt sich daher

$$P(w_1 w_2 \dots w_n) = P(w_1) \cdot P(w_2|w_1) \cdots P(w_{N-1}|w_1 \dots w_{N-2}) \prod_{i=N}^{n} P(w_i|w_{i-N+1} \dots w_{i-1})$$

Aufgrund der nur begrenzt zur Verfügung stehenden Textmengen, die nicht alle Wortfolgen abdecken können, und des hohen Berechnungsaufwandes bei großen Vokabularen werden in der Praxis meist nur 1-Gramme (Monogramme), 2-Gramme (Bigramme) und 3-Gramme (Trigramme) verwendet.

N-Gramme werden aus möglichst großen Textkorpora geschätzt, indem die Auftrittshäufigkeiten der entsprechenden Worttupel gezählt werden:

$$P(w_n|w_{i-N+1} \dots w_{i-1}) = \frac{\#w_{i-N+1} \dots w_{i-1} w_i}{\#w_{i-N+1} \dots w_{i-1}}$$

Aus einem endlichen Textkorpus können die echten Wahrscheinlichkeiten nur angenähert werden. Problematisch hierbei ist, daß selbst bei großen Textmengen bei der Berechnung von Trigrammen nur ein kleiner Teil aller möglichen Wortfolgen auftritt. Für den Großteil aller möglichen Trigramme kann also gar keine Wahrscheinlichkeit angenähert werden. Hier behilft man sich damit, daß in diesen Fällen auf Bigramme und Monogramme zurückgegriffen wird. Kann also ein Trigramm $P(w_3|w_1 w_2)$ nicht berechnet werden, wird auf das Bigramm $P(w_3|w_2)$ zurückgegriffen. Ist auch dieses nicht verfügbar, verwendet man das Monogramm $P(w_3)$.

Die Perplexität gibt den mittleren Verzweigungsgrad eines Sprachmodells an und kann als ein Maß für den Schwierigkeitsgrad des Sprachmodells betrachtet werden. Sie mißt den Informationsgehalt eines Textes und beruht auf dem Begriff der Entropie. Die Perplexität für Trigramme ist definiert als

$$PP = \frac{1}{e^{\frac{1}{n}\sum_{i=1}^{n} log P(w_i|w_{i-2}w_{i-1})}}$$

2.8 Handschrift- und Spracherkennung im Vergleich

Wie die Problemdefinition in Abschnitt 2.2 bereits zeigt, bestehen einige Gemeinsamkeiten zwischen der On-line Handschrift- und Spracherkennung. Analog dem Sprachsignal ist die handschriftliche Eingabe, repräsentiert als Folge von $x-$, y-Koordinaten, ein Signal über die Zeit. In beiden Fällen sind die zur erkennenden Einheiten der Sprache (Zeichen in der Schrifterkennung, Phoneme in der Spracherkennung) wohldefiniert und endlich in der Anzahl. Aus diesen werden jeweils Wörter gebildet, die ihrerseits Bestandteil von Sätzen sind. Ähnlich den Koartikulationseffekten der Sprache, hängt die Form eines Zeichens bei kursiver Schrift oftmals geringfügig von benachbarten Zeichen ab. Bei Druckschrift tritt der Effekt allerdings nur selten auf. In dieser Arbeit wird er nicht gesondert durch entsprechende Modellierung berücksichtigt, da im Vergleich zur Gesamtlänge des zeitlichen Signals eines Zeichens der dadurch betroffene Bereich nur sehr kurz ist. Die unterschiedlichen Schreibweisen eines Zeichens entsprechen unterschiedlichen Sprechweisen von Phonemen. Sowohl in der Handschrift- als auch Spracherkennung muß das Eingabesignal in Zeichen/Phoneme und Wörter zerteilt werden. Die Segmentierung erfolgt hierbei entweder explizit vor der Erkennung oder integriert in den Erkennungsprozeß. Unterschiedliche Charakteristiken der Aufnahme-Hardware, d.h. des Mikrophones bzw. Grafiktabletts, müssen sowohl in der Sprach- als auch Handschrifterkennung berücksichtigt und ggf. durch entsprechende Vorverarbeitung der Daten kompensiert werden.

Obwohl die Gemeinsamkeiten zwischen Handschrift- und Spracherkennung überwiegen, sind hier einige grundsätzliche Unterschiede zu nennen, die zum einen die Erkennung erschweren und zum anderen aber auch erleichtern. Während in der Spracherkennung Wortgrenzen bei kontinuierlicher Sprache ohne Pausen schwer zu detektieren sind, treten sie bei der Handschrifterkennung durch Abheben und Wiederaufsetzen des Stiftes zwischen aufeinanderfolgenden Wörtern, d.h. eine Unterbrechung des sonst kontinuierlichen Punkteflusses, deutlicher hervor. Daher sollten hier Wortfolgen leichter zu segmentieren sein. Allerdings ist zu berücksichtigen, daß nicht jedes Abheben des Stiftes einen Wortübergang bedeutet und umgekehrt in manchen Fällen ein Wortübergang auch ohne Unterbrechung der Trajektorie erfolgen kann.

Ein weiterer Unterschied besteht darin, daß im Sprachsignal die Sprachvektoren eines Phonems immer in ununterbrochener zeitlicher Reihenfolge auftreten, d.h. nicht durch Sprachvektoren anderer Phoneme unterbrochen werden. Ein Phonem befindet

sich also immer vollständig in einem zusammenhängenden Bereich des Sprachsignals. In der Handschrifterkennung hingegen ist dies häufig nicht der Fall, insbesondere bei Wörtern, die z.B. die Buchstaben „t", „i" und „j" enthalten. Diese Buchstaben werden oftmals erst nach Beendigung des Wortes durch die zugehörigen „i"-Punkte und „t"-Striche vervollständigt. Darüber hinaus kann ein bereits beendeter Bereich der Eingabe jederzeit nachträglich durch Einfügen, Löschen oder Überschreiben von Zeichen korrigiert werden, wodurch die Erkennung erschwert wird.

Wie bereits früher erläutert, wird zwischen On-line und Off-line Handschrifterkennung unterschieden. Eine solche Unterteilung existiert für die Spracherkennung nicht, obwohl auch dort Sprache off-line, d.h. nicht im interaktiven Dialog mit dem Erkenner, erkannt werden kann. In diesem Fall wird die Sprache auf Tonbändern aufgenommen und erst zu einem späteren Zeitpunkt digitalisiert und erkannt. Aber auch dann erhält man weiterhin ein dynamisches Sprachsignal über die Zeit. Dem Tonband zur Archivierung von Sprache entspricht in der Handschrifterkennung Papier, auf dem handschriftliche Texte festgehalten werden. Allerdings geht hier die zeitliche Information bei der nachträglichen Digitalisierung der Texte verloren, d.h. es entsteht ein statisches Grauwertbild, in dem die zeitliche Entstehung der Trajektorie nicht mehr kodiert ist. Handschrift hat also zwei grundsätzlich verschiedene Repräsentationen, auf denen ggf. jeweils mit unterschiedlichen Techniken die Erkennung basieren kann. Eine intuitive Darstellung der Sprache ohne den zeitlichen Ablauf existiert nicht.

Wie die vorgestellten Erkennungsansätze im folgenden Kapitel zeigen, haben sich aufgrund der vielen Gemeinsamkeiten der Problemstellung auch die entwickelten Techniken der Handschrift- und Spracherkennung mittlerweile stark angenähert. Viele Erkennungsmethoden wurden zunächst für die Spracherkennung entwickelt und anschließend für die Handschrifterkennung adaptiert.

2.9 Zusammenfassung

Mit Ausnahme einiger weniger Spezialanwendungen basieren heutige Erkennungssysteme für kontinuierliche Handschrift auf analytischen Ansätzen mit expliziter oder impliziter Segmentierung. Die explizite Segmentierung der Eingabe in einzelne Zeichen erfordert aufwendige Segmentierungsalgorithmen, die unabhängig von dem Erkennungsergebnis die Zeichengrenzen bestimmen, was insbesondere bei kursiver Handschrift wegen der frühen Segmentierungsentscheidung häufig zu Fehlerkennungen führt. Diese Ansätze werden daher überwiegend für die Off-line Erkennung maschinell gedruckten Textes eingesetzt, wo die Zeichengrenzen leichter zu finden sind. Analytische Ansätze mit impliziter Segmentierung verlagern die Segmentierungsentscheidung an das Ende des Erkennungsprozesses, so daß das Erkennungsergebnis mit in die Entscheidung einbezogen werden kann. Frühe Segmentierungsentscheidungen, die in den nachfolgenden Erkennungsschritten nicht mehr korrigiert werden können, werden so vermieden.

Neuronale Netze und Hidden Markov Modelle sind aufgrund ihrer Eigenschaften

die derzeit am häufigsten verwendeten Techniken zur Erkennung sowohl von Einzelzeichen als auch kontinuierlicher Handschrift. Der Erfolg neuronaler Netze begründet sich auf ihre Fähigkeit, zum einen stetige Funktionen beliebig genau approximieren und zum anderen die a posteriori Wahrscheinlichkeiten der gegebenen Trainingsmuster schätzen zu können. Hidden Markov Modelle verdanken ihren Erfolg insbesondere der Fähigkeit, sequentielle Daten mit zeitlicher Variabilität zu verarbeiten, wie sie bei der On-line Handschrifterkennung vorliegen. Sowohl neuronale Netze als auch Hidden Markov Modelle erlernen ihre Parameter automatisch anhand einer entsprechenden Datenbasis mit Trainingsmustern. In dieser Arbeit werden die diskriminativen Fähigkeiten neuronaler Netze mit Hidden Markov Modellen zur Modellierung der syntaktischen Informationen der Sprache in einer hybriden Architektur vereinigt.

Syntaktisches Wissen über die zugrundeliegende Sprache kann durch Vokabulare und Sprachmodelle in die Erkennungsentscheidung eingebracht werden. Vokabulare schränken die Erkennung auf eine Liste legaler Worte der Sprache ein. Worte des Vokabulars werden dabei durch Hintereinanderschaltung der Modelle einzelner Zeichen gebildet. Sprachmodelle stellen darüber hinaus die a priori Wahrscheinlichkeit für bestimmte Wortfolgen in der Sprache zur Verfügung. Diese Wahrscheinlichkeiten werden anhand großer Textmengen berechnet. In der Praxis werden üblicherweise nur Mono-, Bi- und Trigramme verwendet.

Kapitel 3

Verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über verwandte Arbeiten aus dem Forschungsgebiet der Handschrifterkennung. Das Hauptaugenmerk gilt dabei Systemen zur On-line Erkennung von Wörtern und Sätzen, die typische Erkenneransätze repräsentieren, d.h. jeweils für eine Klasse von Erkennern stehen. Berücksichtigt werden aber auch Ansätze der Off-line Erkennung, die für die On-line Erkennung adaptiert wurden. Beispiele für ganzheitliche und analytische Ansätze mit expliziter Segmentierung werden hier nur der Vollständigkeit halber vorgestellt, da sich Ansätze mit impliziter Segmentierung als deutlich überlegen erwiesen haben. Spezielle Ansätze der Einzelzeichenerkennung, die keinen Beitrag zur Wort- und Satzerkennung leisten, werden außer Acht gelassen, da die Einzelzeichenerkennung nicht primäres Thema dieser Arbeit ist. Soweit möglich werden für alle beschriebenen Arbeiten Erkennungsergebnisse und die Bedingungen, unter denen sie erzielt wurden, angegeben, um den derzeitigen Stand der Forschung in der Handschrifterkennung zu verdeutlichen. Die Gruppierung der Arbeiten in einzelne Abschnitte dieses Kapitels ist lediglich eine grobe Orientierung und nicht eindeutig, da manche Systeme mehreren Konzepten zuzuordnen sind. Einen ausführlicheren Überblick über Erkennungsansätze und konkrete Systeme der On-line Handschrifterkennung geben **Tappert et al.** [TSW90], **Nouboud et al.** [NP90] und **Lecolinet et al.** [LB93]. Einen Einblick in die Off-line Handschrifterkennung und deren geschichtliche Entwicklung vermitteln die Zusammenfassungen von **Govindan et al.** [GS90], **Suen et al.** [SBM80] und **Harmon** [Har72].

3.1 Ganzheitliche Ansätze

Eines der ersten Systeme zur On-line Erkennung kursiver Wörter wurde von **Earnest** [Ear62] bereits im Jahr 1962 entwickelt. Obwohl für die On-line Erkennung entworfen, folgt sein Ansatz eher Off-line Methoden, da kein Gebrauch von den zeitlichen Informationen gemacht wird. Das gesamte zu erkennende Wort wird durch einen 3-wertigen Merkmalsvektor repräsentiert, der die Anzahl der Oberlängen, Unterlängen und „t"-Striche spezifiziert. Dieser Merkmalsvektor wird mit den theoretischen Merkmalsvektoren von 10 000 Wörtern eines Vokabulars verglichen und daraus die Liste der Wörter mit derselben Kodierung ermittelt. Aus dieser Liste mit im Durchschnitt 8

Wörtern wird abschließend dasjenige Wort gewählt, das die x-Position der Ober-, Unterlängen und „t"-Striche am besten erklärt. Getestet mit 107 zufällig ausgewählten Wörtern von 5 Schreibern erzielte dieses System eine Erkennungsrate von 60%.

Einen ähnlichen Ansatz verfolgten **Frishkopf et al.** [FH61] etwa zur selben Zeit. Im Gegensatz zu der Arbeit von Earnest wurde hier die zeitliche Information mit in die Erkennung einbezogen. Wörter werden durch die zeitlich geordnete Sequenz der lokalen Extremalpunkte der Stifttrajektorie in horizontaler und vertikaler Richtung repräsentiert. Jeder Extremalpunkt wird durch einen Merkmalsvektor mit folgenden Informationen ergänzt: Art des Extremalpunktes (Minimum, Maximum), die Neigung der Trajektorie zwischen dem vorherigen und dem aktuellen Extremum (positiv, negativ), Krümmung der Trajektorie zwischen dem aktuellen und dem folgenden Extremum (konkav, konvex) und letztlich die vertikale Position des Extremums (Unter-, Oberlänge, Mittelbereich). Nach der Berechnung dieser Merkmale wird die Liste der Extrema mit denen von Wörtern aus einem Vokabular verglichen, die eine ähnliche Anzahl Extrema aufweisen. Das Wort bzw. die Wörter mit der besten Übereinstimmung bilden die Ausgabehypothese des Systems. Beim Test von 500 Wörtern von 5 Schreibern befand sich die gesuchte Antwort in 46% aller Fälle unter den besten zwei Hypothesen des Erkenners. In 85% aller Fälle war das gesuchte Wort unter den besten 20 Hypothesen zu finden.

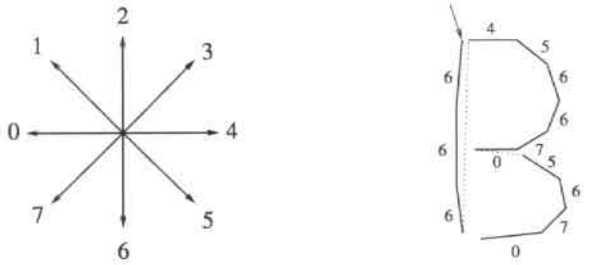


Abbildung 3.1: Kodierung einer Trajektorie durch eine Folge von Freeman Codes

Auf einer Repräsentation der Wörter durch eine Folge sog. Freeman Codes [Fre61] basiert das Erkennungssystem von **Farag** [Far79], das als eines der ersten Systeme Hidden Markov Modelle zur Handschrifterkennung einsetzte. Für jeweils zwei aufeinanderfolgende Punkte der Trajektorie wird der Verbindungsgeraden eine von 8 Richtungen zugewiesen, wie in Abbildung 3.1 dargestellt ist. Für das dort abgebildete Zeichen „B" ergibt sich daraus die Kodierung „6664566705670". Für eine Gewichtung der Übereinstimmung beim Vergleich eines Wortes mit den Prototypen eines kleinen, 10 Wörter umfassenden Vokabulars werden Hidden Markov Modelle eingesetzt. Die Zustände der Markov Ketten entsprechen dabei den 8 Richtungsvektoren der Freeman Kodierung. Getestet auf 200 Wörtern von 10 unterschiedlichen Schreibern, die gleichzeitig auch für das Training der Hidden Markov Modelle dienten, wurde hier eine Erkennungsrate von 98% erzielt.

Die in diesem Abschnitt vorgestellten Ansätze sind insofern interessant, als sie

bereits Methoden verwenden, die auch heute noch eingesetzt werden, wenn auch meist in weiterentwickelten Formen. So finden sich in heutigen On-line Systemen häufig noch immer Merkmale, wie Ober-, Unterlängen, Richtungen, Winkeländerungen usw. Auch die Beobachtung, daß lokale Minima und Maxima der Stifttrajektorie wichtige Informationsquellen bei der Erkennung sind, ist heute noch von Bedeutung.

3.2 Analytische Ansätze mit expliziter Segmentierung

Bereits sehr früh wurden neben ganzheitlichen Ansätzen auch analytische Ansätze mit expliziter Segmentierung entwickelt, bei denen Wörter durch Zerlegung in kleinere Einheiten und nicht mehr als Ganzes erkannt werden. Eines der ersten Systeme dieser Art wurde 1964 von **Mermelstein et al.** [ME64] entwickelt. Wörter werden dort anhand der lokalen y-Extrema (Abbildung 3.2a) in eine zeitlich geordnete Folge von aufwärts und abwärts gerichteten Liniensegmenten unterteilt. Jedes der Liniensegmente wird anschließend entsprechend seiner statistischen Wahrscheinlichkeit einer von 12 Klassen zugeordnet. Die sich daraus ergebende zeitliche Folge dieser Klassen wird analog den oben vorgestellten ganzheitlichen Ansätzen mit Prototypen der Wörter eines gegebenen Vokabulars verglichen und das am besten passende Wort ausgewählt. Experimente mit diesem Ansatz wurden anhand von 100 Wörtern (12 verschiedene Wörter) von 4 Schreibern durchgeführt. Die Erkennungsrate betrug 90%, wenn die Wahrscheinlichkeiten der Liniensegmente auf den Testdaten mitberechnet wurden, sonst 60%. Lediglich anhand der lokalen y-Minima segmentiert wird die Eingabe in dem System von **Ehrich et al.** [EK75], das ein Jahrzehnt später entwickelt wurde.

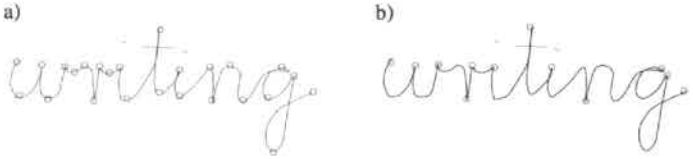


Abbildung 3.2: Segmentierung anhand der lokalen y-Extrema (a) und der minimalen Geschwindigkeit (b)

Alternative Segmentierungspunkte werden von **Morrasso et al.** verwendet [Mor89, MBPV93]. Anstatt durch lokale Extrema sind ihre Segmentierungspunkte durch Punkte der Trajektorie mit minimaler Schreibgeschwindigkeit gegeben (Abbildung 3.2b). Dies sind insbesondere Punkte, in denen abrupt die Schreibrichtung geändert wird. Die so gefundenen Liniensegmente werden anschließend durch einen 9-dimensionalen Merkmalsvektor kodiert und mit selbstorganisierenden Karten von Kohonen [Koh88] klassifiziert, so daß sich eine Folge von Buchstabenhypothesen ergibt, die abschließend mit einem 4000 Wörter umfassenden Vokabular verglichen

wird. Mit diesem Ansatz erreichten Morrasso et al. eine schreiberunabhängige Worterkennungsrate von knapp 70%. Dieselbe Segmentierungstechnik kam auch bei **Flann et al.** zum Einsatz [FS93]. Für die Klassifikation der Segmente werden hier jedoch mehrschichtige Perzeptrons verwendet. Bei einem Vokabular von 1 000 Wörtern ist die schreiberabhängige Erkennungsrate mit 90% angegeben.

Weitere Techniken der Segmentierung beschreiben **Teulings et al.** in [TS92]. Eine Zusammenfassung von Segmentierungstechniken der Off-line Handschrifterkennung geben **Dunn et al.** in [DW92].

Systeme, die auf einer Segmentierung der Eingabe anhand der lokalen Extrema der Trajektorie beruhen, zeigen eine schlechte Generalisierung auf unbekannte Schreiber. Die wesentlichen Probleme ergeben sich dadurch, daß zum einen viele Extrema keine gewünschten Segmentierungspunkte repräsentieren und daher entfernt werden müssen bzw. in manchen Fällen ein eigentlich erwartetes Extremum fehlt. Zum anderen hängt die Existenz der Extrema stark von dem Schreibstil ab. In dieser Arbeit werden die lokalen Extrema daher lediglich dazu genutzt, die Schreiblinien der Eingabe zu bestimmen. Eine Segmentierungsentscheidung anhand solcher Extrema wird aufgrund der eben genannten Probleme vermieden. Die Erkenntnis, daß eine gute Segmentierung bereits die Erkennung der Eingabe voraussetzt, führte zu den analytischen Ansätzen mit impliziter Segmentierung. Diesem Ansatz folgt auch das in dieser Arbeit vorgestellte System.

3.3 Analytische Ansätze mit impliziter Segmentierung

Die Ergebnisse der kontinuierlichen Spracherkennung haben gezeigt, daß eine explizite Segmentierung des Sprachsignals in Phoneme, Buchstaben oder Wörter äußerst problematisch und sehr unzuverlässig ist. Dies führte dort früh zu Ansätzen, die auf eine explizite Segmentierung verzichten, d.h. die Segmentierung gleichzeitig mit der Erkennung durchführen. Letztendlich interessieren nicht die gefundenen Buchstabengrenzen, sondern das gesamte erkannte Wort bzw. der gesamte Satz. Seit einigen Jahren wurde dieser Ansatz nun auch sowohl für die On-line als auch Off-line Handschrifterkennung adaptiert. Die folgenden Abschnitte stellen einige dieser Systeme vor.

3.3.1 Ansätze mit neuronalen Netzen

Eines der ersten Systeme, das neuronale Spracherkennungstechniken für die Online Erkennung von Einzelzeichen einsetzte, wurde 1991 von **Guyon et al.** vorgestellt [GAL+91, GHA+92]. In diesem System ist bereits zu sehen, wie anstelle einer 2-dimensionalen bildlichen Repräsentation der Eingabe die zeitliche Information für die Erkennung genutzt werden kann. Die Trajektorie eines Zeichens wird neu abgetastet, so daß sie für alle Zeichenklassen gleichermaßen aus 81 Datenpunkten dargestellt wird. Jeder dieser Datenpunkte wird anschließend durch einen Merkmalsvektor mit

der Position des Punktes, der Schreibrichtung bzw. Krümmung der Trajektorie in diesem Punkt und dem aktuellen Stiftstatus ersetzt. Diese Folge der Merkmalsvektoren dient als Eingabe für ein *Time Delay Neural Network* (TDNN) [WHH+87, WHH+89] mit 5 Schichten (Abbildung 3.3). Dieses TDNN wurde mit insgesamt 12 000 Ziffern

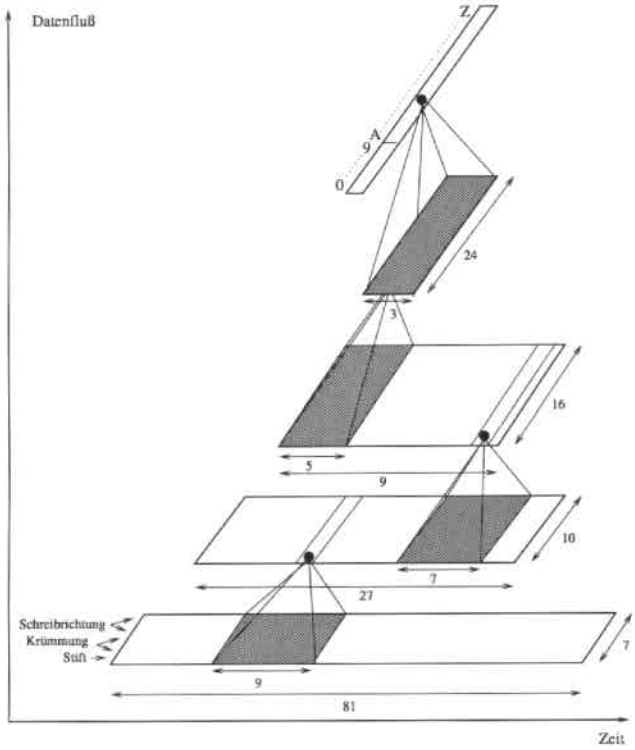


Abbildung 3.3: Time Delay Neural Network für die Einzelzeichenerkennung [GAL+91]

und Großbuchstaben von einer großen Zahl unterschiedlicher Schreiber trainiert und anschließend auf 2 500 Beispielen einer disjunkten Schreibermenge getestet. Dabei wurde eine Buchstabenerkennungsrate von 96.6% erzielt.

Die von Guyon et al. vorgestellte Kombination einer zeitlichen Abfolge von Merkmalsvektoren als Eingaberepräsentation und eines TDNN zur Berechnung von Buchstabenhypothesen wurde mittlerweile in mehreren anderen Systemen in erweiterter Form übernommen und ist auch in dieser Arbeit wiederzufinden. Ein Beispiel aus neuerer Zeit ist der Einzelworterkenner von **Seni** von 1995 [Sen95]. Die prinzipielle Eingaberepräsentation als zeitlich geordnete Folge von Merkmalsvektoren entspricht der des obigen Systems von Guyon et al. Die Merkmalsvektoren enthalten hier je-

doch für jeden Punkt nur 3 Informationen: die Schreibrichtung, die Krümmung und die vertikale Position des Punktes. Ein TDNN mit 26 Ausgabeklassen berechnet aus dieser Folge für jeweils einen begrenzten Bereich der Eingabe die a posteriori Wahrscheinlichkeiten, daß ein bestimmter Buchstabe bzw. ein Teil dieses Buchstabens in diesem Bereich sichtbar ist (siehe auch Kapitel 6 für eine detailliertere Einführung in die TDNN Architektur für die Handschrifterkennung). Aus der zeitlichen Abfolge

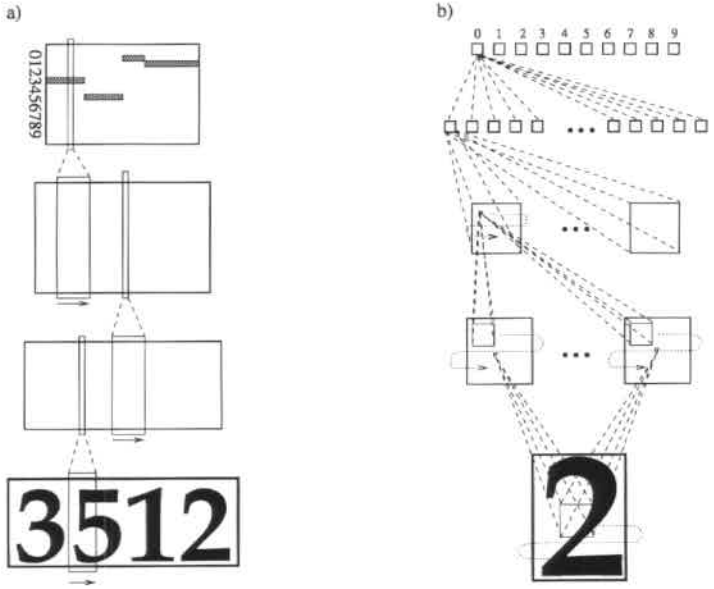


Abbildung 3.4: Neuronale Netzarchitekturen für die Off-line Handschrifterkennung von Matan et al. (a) und LeCun et al. (b)

der Ausgabevektoren des TDNN wird dann die wahrscheinlichste Buchstabensequenz ermittelt. Diese Buchstabensequenz wird abschließend mit einer Liste von einigen wenigen Worthypothesen verglichen, die zuvor aus einem 21 000 Wörter umfassenden Vokabular ausgewählt wurden. Die Auswahl dieser Wortliste findet aufgrund der generellen Struktur der Eingabe statt. Die Trajektorie wird in einzelne Trajektoriensegmente, begrenzt durch lokale y-Extrema, zerlegt, die jeweils anhand ihrer Eigenschaft in eine von 9 Segmentklassen klassifiziert werden. Die zeitliche Abfolge dieser Segmente wird nun mit der theoretischen, manuell bestimmten Abfolge der 21 000 Wörter des Vokabulars verglichen, und so werden die Wörter bestimmt, die optisch der Eingabe am meisten ähneln. Die Editierdistanz [WF74] zwischen der durch das TDNN bestimmten Buchstabensequenz und den so ermittelten Worthypothesen entscheidet letztendlich über die endgültige Ausgabehypothese des Gesamtsystems. Trainiert auf 2 443 Wörtern von 55 Schreibern und getestet auf 466 Wörtern von 9

weiteren Schreibern, erreicht dieses System eine Worterkennungsrate von 62.4%.

Eine der Haupteigenschaften der TDNN-Architektur ist ihre zeitliche Verschiebungsinvarianz. Zusammenhänge aufeinanderfolgender Merkmalsvektoren werden unabhängig von ihrer zeitlichen Position im Eingabesignal erkannt. In der Off-line Erkennung findet man ähnliche neuronale Ansätze, die jedoch keine zeitliche, sondern eine räumliche Verschiebungsinvarianz aufweisen, da dort die zeitlichen Informationen nicht vorhanden sind. Unter der Bezeichnung *Space Displacement Neural Network* (SDNN) stellten z.B. **Matan et al.** eine TDNN-ähnliche neuronale Netzarchitektur für die Erkennung von Ziffernfolgen vor [MBLD92]. Hier wird mit dem 4-schichtigen SDNN (Abbildung 3.4a) jeweils für einen begrenzten räumlichen, die ganze Höhe umfassenden Bereich des Grauwertbildes der Eingabe ein Ausgabevektor mit Ziffernhypothesen generiert. Der Viterbi-Algorithmus ermittelt aus dieser Folge der Ausgabevektoren die wahrscheinlichste Ziffernfolge. Während der Ansatz von Matan et al. nur in der horizontalen Richtung verschiebungsinvariant ist, ist die Netzarchitektur von **LeCun et al.** zur Erkennung einzelner Ziffern zusätzlich invariant gegenüber Verschiebungen in vertikaler Richtung [LBD+89]. D.h. der jeweils betrachtete Bereich des Grauwertbildes ist nicht nur in horizontaler, sondern zusätzlich auch in vertikaler Richtung beschränkt (Abbildung 3.4b). Einen ähnlichen, erweiterten Ansatz mit Verschiebungsinvarianz in beiden Richtungen stellen auch **Keeler et al.** in [KR91] für die Erkennung von Ziffernfolgen vor.

3.3.2 Ansätze mit Hidden Markov Modellen

Mittlerweile haben sich Hidden Markov Modelle neben neuronalen Netzen in der On-line Handschrifterkennung als zweite Klassifikationstechnik etabliert. Dieser Abschnitt stellt einige der auf HMMs basierenden Systeme vor.

Bellegarda et al. beschreiben in [NBNB93, BBNN94] einen HMM-basierten Online Erkenner für Einzelzeichen mit 81 Zeichenklassen (Ziffen, Buchstaben und Sonderzeichen). In diesem System durchläuft die Eingabe eine einfache Vorverarbeitung, die nach einer Neuabtastung des originalen Eingabesignals für jeden Datenpunkt einen Merkmalsvektor mit Informationen über die lokale Position des Punktes und die Krümmung der Trajektorie berechnet. Jede Zeichenklasse wird durch eine Menge einfacher HMMs mit zwei Zuständen modelliert, die jeweils die unterschiedlichen Schreibweisen der Zeichen repräsentieren (Abbildung 3.5a). Ein schreiberabhängiger Test des Systems auf 6400 Einzelzeichen von 8 Schreibern ergab eine Erkennungsrate von 86.6%. Das System wurde hierbei mit 12800 Zeichen derselben Schreiber trainiert. Getestet auf 3200 Zeichen von 4 weiteren Schreibern, ergab sich eine schreiberunabhängige Erkennungsrate von 80.9%. Eine Erweiterung dieses Systems für die Erkennung ganzer Sätze mit Vokabularen bis zu 20000 Wörtern beschreiben **Ratzlaff et al.** in [RNM96]. Das Vokabular wird hier durch eine Struktur repräsentiert, in der gemeinsame Präfixe und Postfixe der Wortmodelle zusammengefaßt werden. Diese Struktur wird bei der Erkennung mit einer Strahlensuche, die den Suchraum einschränkt, durchlaufen. Getestet auf 75 Sätzen von 75 unterschiedlichen Schreibern wird eine schreiberunabhängige Erkennungsrate von etwa 63% erzielt. Die Erken-

nungsrate liegt damit unter vergleichbaren Bedingungen mehr als 20% unter dem besten Ergebnis des hier vorgestellten hybriden Ansatzes aus neuronalem Netzwerk und HMM.

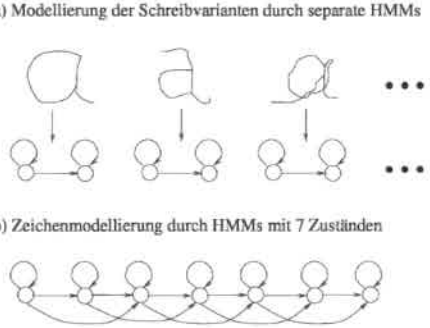


Abbildung 3.5: Modellierung von Einzelzeichen in [BBNN94, NBNB93] (a) und [MSSC94, SMSC94] (b)

Ein interessantes Experiment führten **Starner et al.** in [SMSC94, MSSC94] durch. Sie entwickelten kein spezielles On-line Erkennungssystem für Handschrift, sondern gingen der Frage nach, ob ein Spracherkennungssystem ohne Modifikation für die Schrifterkennung eingesetzt werden kann. Zu diesem Zweck verwendeten sie ihr auf HMMs basierendes, für die kontinuierliche Spracherkennung entwickeltes BBN Byblos System [KAM+94]. Für ein erstes schreiberabhängiges Experiment sammelten sie insgesamt 475 Sätze von einem Schreiber, wovon 381 zum Training und 94 zum Test des Systems verwendet wurden. Die Sätze wurden aus dem ARPA Airline Travel Information Service (ATIS) Textkorpus ausgewählt und enthielten nur Klein- und Großbuchstaben. Die Vorverarbeitung bestand lediglich aus der Berechnung des Schreibwinkels und der Änderung des Schreibwinkels. Das Ergebnis war also eine zeitliche Sequenz 2-dimensionaler Merkmalsvektoren, die anstelle des ursprünglichen Sprachsignals in das Byblos System gegeben wurde. Alle Zeichen wurden jeweils durch ein HMM mit 7 Zuständen modelliert (Abbildung 3.5b). Mit einem 3050 Wörter umfassenden Vokabular wurde eine Wortakkuratheit[1] von 95.8% erzielt. Durch den Einsatz eines auf 17209 Sätzen des ATIS Textkorpus' berechneten Bigramms konnte diese Wortakkuratheit auf 98.9% verbessert werden. Ein analoges Experiment wurde anschließend mit weiteren 3217 Sätzen von 6 Schreibern durchgeführt. Diesmal wurden die Sätze aus dem ARPA Wall Street Journal (WSJ) Textkorpus gewählt und das Bigramm auf etwa 2 Mio. Sätzen des WSJ berechnet. Zusätzlich zu Buchstaben wurden hier auch Ziffern und Sonderzeichen berücksichtigt. Mit einem Vokabular von 25595 Wörtern und dem Bigramm betrug die schreiberabhängige Erkennungsrate 95.8%. Allerdings repräsentieren die verwendeten Daten nur bedingt realistische kontinuierliche Handschrift, da den Schreibern der Sätze die Instruktion gegeben wurde, innerhalb der

[1] Siehe Kapitel 4.4 für eine Definition der Wortakkuratheit.

Wörter den Stift nicht abzuheben und dementsprechend „i"-Punkte und „t"-Striche erst am Ende eines jeden Wortes einzufügen. Das Segmentierungsproblem wird dadurch auf ein Minimum reduziert, da innerhalb des Satzes jedes Abheben des Stiftes automatisch auch den Neubeginn eines Wortes anzeigt. Die in dieser Arbeit verwendeten Daten hingegen weisen solche Einschränkungen nicht auf. Trotz dieser Einschränkungen waren die Ergebnisse jedoch vielversprechend, insbesondere da keine Änderungen des Spracherkennungssystems durchgeführt wurden.

Das bislang beste auf Hidden Markov Modellen basierende System wird von **Dolfing et al.** in [DHU97] präsentiert. Auf Testdaten, die mit den in dieser Arbeit verwendeten Daten vergleichbar sind, erzielen sie mit einem 20 000 Wörter umfassenden Vokabular eine schreiberunabhängige Worterkennungsrate von bis zu 88.8%.

Einfache Versuche mit HMMs für die On-line Erkennung wurden auch von **Nag et al.** bereits im Jahr 1986 durchgeführt [NWF86]. Sie entwickelten einen einfachen HMM Erkenner mit einem kleinen Vokabular für kursive Handschrift. Einen guten Überblick über Systeme zur Off-line Erkennung mit HMM Techniken gewinnt man aus der Arbeit von **Elms** [Elm96]. Sie widmet sich insbesondere der Darstellung von Texten mit Hidden Markov Modellen.

Alle hier vorgestellten Systeme haben gemeinsam, daß bei ihrer Entwicklung das Problem der Vorverarbeitung fast vollständig ausgeklammert wurde. Dies trägt in hohem Maße mit dazu bei, daß im Vergleich zu dem in dieser Arbeit präsentierten System, in dem die Vorverarbeitung einen wichtigen Teil der Gesamterkennung ausmacht, eine deutlich schlechtere Erkennungsleistung erzielt wird.

3.3.3 Hybride Systeme

Hybride Systeme verbinden die konnektionistischen Techniken mit weiteren Techniken, die meist einer zeitlichen Integration der variabel langen handschriftlichen Eingaben dienen. Der in dieser Arbeit vorgestellte Ansatz fällt ebenso in diese Erkennerklasse wie der gleichzeitig von **Schenkel et al.** entwickelte Erkenner [SGH94, Sch95]. Es handelt sich dabei um eine Weiterentwicklung des bereits in Abschnitt 3.3.1 vorgestellten Einzelzeichenerkenners von Guyon et al., so daß nun auch Einzelworte erkannt werden können. Abbildung 3.6, die aus [Sch95] entnommen wurde, gibt einen Überblick über dieses System. Die Eingaberepräsentation für das verwendete Time Delay Neural Network ist wiederum eine Folge von Merkmalsvektoren, die zusätzlich zur Schreibrichtung, Krümmung und dem Stiftstatus um 5 weitere Merkmale, wie Schreibgeschwindigkeit und Positionsangaben, ergänzt wurden. Das TDNN berechnet für jeweils einen begrenzten Kontext dieser Eingabe einen Ausgabevektor mit Buchstabenhypothesen, der die a posteriori Wahrscheinlichkeit angibt, daß ein bestimmtes Zeichen in dem gerade betrachteten Bereich der Eingabe vorhanden ist. Aus diesen zeitlich geordneten Ausgabevektoren des TDNN wird mit Hilfe von Buchstaben-HMMs die wahrscheinlichste Buchstabenfolge extrahiert. Die Beobachtungswahrscheinlichkeiten der HMMs für jeden Zeitschritt sind durch die Ausgabevektoren des TDNN gegeben, die Übergangswahrscheinlichkeiten werden per Hand gesetzt, so daß sie die erwarteten Buchstabenlängen modellieren. Die HMMs schränken nicht die möglichen Buchstabenübergänge ein, d.h. bei der Suche der wahrscheinlich-

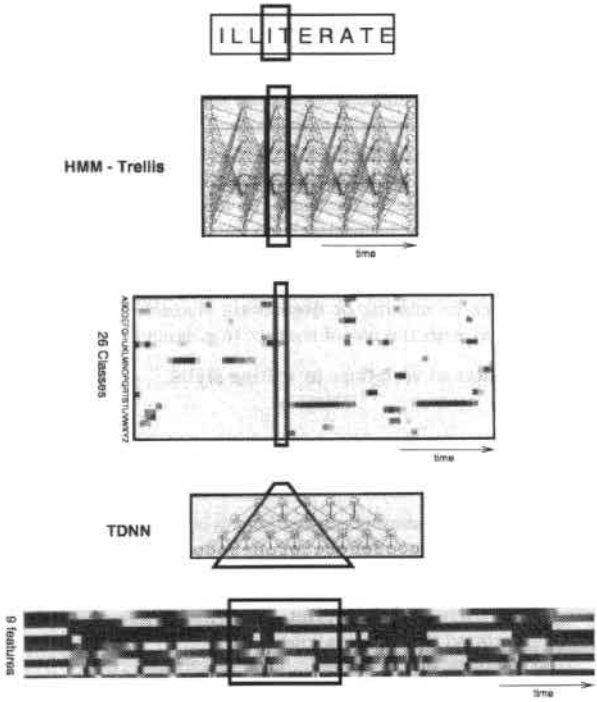


Abbildung 3.6: Der Worterkenner von Schenkel et al. im Überblick

sten Buchstabensequenz kann von jedem Buchstaben zu jedem anderen Buchstaben gewechselt werden, so daß ggf. auch nicht legale Wörter der Sprache als Ausgabehypothese erzeugt werden können. Daher wird als letzter Erkennungsschritt die Ausgabehypothese mit einem 25 000 Wörter umfassenden Vokabular verglichen und das Wort mit der geringsten Editierdistanz als endgültige Hypothese ausgegeben.

Zur Evaluation dieses hybriden Systems wurde das TDNN mit 20 000 Wörtern von 59 Schreibern trainiert und auf zwei unterschiedlichen Testmengen einer disjunkten Schreibermenge mit einem 25 000 Wörter Vokabular getestet. Die erste Testmenge bestand aus Wörtern in Blockschrift, die nur Großbuchstaben enthielten. Auf diesen Wörtern wurde eine Worterkennungsrate von 96% erreicht. Bei der zweiten Testmenge handelte es sich um kursive Wörter mit Klein- und Großbuchstaben, die vergleichbar mit den in dieser Arbeit verwendeten Daten sind. Auf ihr ergab sich eine Worterkennungsrate von 80%.

Das System von Schenkel et al. hat einige Gemeinsamkeiten mit dem in dieser Arbeit vorgestellten Ansatz. In beiden Systemen wird eine Repräsentation der Eingabe durch eine Sequenz von Merkmalsvektoren eingesetzt. Die Merkmalsvektoren in

Schenkel et al. beschränken sich jedoch auf die Kodierung zeitlich lokaler Merkmale der Trajektorie, während in dieser Arbeit zusätzlich auch von der Zeit unabhängige Merkmale betrachtet werden. Ebenfalls bildet in beiden Systemen ein Time Delay Neural Network die Grundlage der Erkennung. In dieser Arbeit wird das TDNN jedoch in einer weiterentwickelten Form verwendet, die eine HMM-ähnliche Modellierung der einzelnen Zeichen durch mehrere Zustände ermöglicht, wodurch die Erkennungsleistung deutlich verbessert wird. HMMs werden in dem System von Schenkel et al. lediglich für die Längenmodellierung der einzelnen Zeichen in die Erkennung integriert. Vokabulare werden erst nach der eigentlichen Erkennung herangezogen. Hier hingegen wird das Vokabular direkt in den Erkennungsprozeß integriert und so werden von vornherein nur legale Wörter des Vokabulars generiert. Die Einbindung globaler Merkmale in die Eingaberepräsentation, die Modellierung einzelner Zeichen durch mehrere Zustände und insbesondere die direkte Integration des Vokabulars in den Erkennungsprozeß sind die wesentlichen Faktoren, die zu der etwa 13% höheren Erkennungsleistung dieses Systems gegenüber dem System von Schenkel et al. auf vergleichbaren Daten beitragen.

3.4 Zusammenfassung

Bereits die wenigen in diesem Kapitel vorgestellten Systeme lassen die wichtigsten Entwicklungstendenzen in der Forschungsliteratur erkennen. Zu beobachten ist, daß sich, aufgrund der äußerst schwierigen expliziten Segmentierung von Wörtern und Sätzen, Ansätze mit impliziter Segmentierung als deutlich überlegen erwiesen haben. Ganzheitliche Ansätze spielen aufgrund der schlechten Skalierbarkeit nur noch für Spezialanwendungen mit kleinen Vokabularen eine Rolle. Die Ähnlichkeit der Problemstellung zwischen Handschrift- und Spracherkennung führt zu einer immer größeren Annäherung der Erkennungstechniken aus beiden Bereichen. Nahezu alle aktuellen Ansätze greifen bei der Erkennung auf syntaktisches Wissen über die zugrundeliegende Sprache in Form von Vokabularen oder Sprachmodellen zurück. Lediglich auf HMMs basierende Erkenner integrieren bislang Vokabulare direkt in den Erkennungsprozeß. Neuronale Ansätze hingegen greifen erst nach der Generierung von Buchstabenhypothesen auf das syntaktische Wissen eines Vokabulars zurück. In dieser Arbeit wird erstmals bei der Handschrifterkennung das Vokabular, modelliert durch HMMs, auch bei einem neuronalen Ansatz direkt in den Erkennungsprozeß eingebunden. Wie die Experimente in Kapitel 7 zeigen, führt dies gegenüber einer nachträglichen Korrektur der Buchstabenhypothesen in diesem System zu einer Verbesserung der Erkennungsleistung von 87.2% auf 93.4% für ein 20 000 Wörter umfassendes Vokabular.

Desweiteren ist zu beobachten, daß sich mittlerweile der Forschungsschwerpunkt von der Einzelzeichenerkennung auf die Erkennung isolierter Wörter verlagert hat. Die Satzerkennung hingegen wurde bislang nur in einigen wenigen HMM-Systemen in Angriff genommen. Insbesondere die vorgestellten neuronalen Ansätze mit nachträglicher Korrektur der Buchstabenhypothesen stoßen hier bei der Erweiterung über die

Einzelworterkennung hinaus an ihre Grenzen. Eine Erweiterung dieser Systeme unter Beibehaltung des Vergleichs mit den Wörtern eines Vokabulars würde eine explizite Segmentierung der Sätze in Einzelworte erfordern. Durch die Integration des syntaktischen Wissens direkt in den Suchprozeß, ist bei dem in dieser Arbeit vorgestellten System eine explizite Segmentierung in Worte nicht erforderlich.

Auffallend in vielen Systemen ist, daß man sich zwar mit der Modellierung der Schrift und der eigentlichen Erkennung intensiv auseinandersetzt, die Vorverarbeitung, insbesondere bei HMM-basierten Systemen, jedoch oftmals nur am Rande beachtet wird, obwohl diese einen großen Teil zur Erkennungsleistung beiträgt, wie auch diese Arbeit zeigt. Häufigste Repräsentationsform der Eingabe für die Erkennung ist eine zeitliche Folge von Merkmalsvektoren, die jeweils nur aus einem begrenzten zeitlichen Kontext berechnet werden. In dieser Arbeit werden erstmals auch Merkmale in die Merkmalsvektoren integriert, die von der Zeit unabhängige Eigenschaften der Trajektorie kodieren.

	Ansatz	Vokabulargröße	Erkennungsrate
Wörter			
Seni [Sen95]	TDNN	21 000	62.4%
Schenkel et al. [SGH94, Sch95]	TDNN	25 000	80.0%
Dolfing et al. [DHU97]	HMM	20 000	88.8%
Manke	MS-TDNN	20 000	93.4%
Sätze			
Ratzlaff et al. [RNM96]	HMM	20 000	63.0%
Manke	MS-TDNN	20 000	86.6%

Tabelle 3.1: Die besten On-line Erkenner im Vergleich mit dem in dieser Arbeit vorgestellten System

Den bisherigen Stand der Forschung in Bezug auf die Erkennungsrate definieren bei der Worterkennung die Systeme von Schenkel et al. und Dolfing et al. und bei der Satzerkennung die Systeme von Starner et al. und Ratzlaff et al. Das System von Schenkel et al. basiert auf einem hybriden Ansatz aus neuronalem Netzwerk und HMM zur Längenmodellierung und erzielt auf kursiver Handschrift mit einem 25 000 Wörter umfassenden Vokabular eine Erkennungsleistung von 80%. Mit einem HMM-basierten Ansatz beträgt die Erkennungsrate des Systems von Dolfing et al. 88.8% bei einem 20 000 Wörter umfassenden Vokabular. Beide Systeme werden durch das in dieser Arbeit präsentierte System mit einer Worterkennungsrate von über 93% für vergleichbare Vokabulargrößen und Daten übertroffen. Das System von Starner et al. zur Satzerkennung hat eine schreiberabhängige Wortakkuratheit von 95.8%. Dieses Ergebnis wurde jedoch nur auf speziellen Daten erzielt, die keine reale Handschrift widerspiegeln. Die Ergebnisse sind somit nicht mit den in dieser Arbeit erzielten vergleichbar. Vergleichbar sind jedoch die Ergebnisse von Ratzlaff et al., die bei einer Vokabulargröße von 20 000 Wörtern mit 63% Wortakkuratheit mehr als 20% unter den Ergebnissen dieser Arbeit liegen. Der Vergleich der Erkennungsrate ist noch einmal in Tabelle 3.1 zusammengefaßt.

Kapitel 4

Systemüberblick und Evaluierung

Dieses Kapitel beschreibt die grundlegenden Prinzipien, die bei der Entwicklung des hier vorgestellten On-line Handschrifterkenners für Einzelzeichen, Wörter und Sätze als Leitfaden dienten, und gibt einen Überblick über das daraus resultierende Gesamtsystem. Dies verdeutlicht den Datenfluß durch die einzelnen Verarbeitungsstufen des Systems, die einzeln in den folgenden Kapiteln detaillierter behandelt werden, und die dabei entstehenden Repräsentationen der Eingabe, ausgehend vom ursprünglichen, punktbasierten Eingabesignal bis hin zu einzelnen Zeichen und Zeichenfolgen.

In dieser Arbeit werden zahlreiche Experimente zur Motivation von Entwurfsentscheidungen und Evaluierung der Leistungsfähigkeit des Handschrifterkenners präsentiert. Für diese Experimente werden Datenbasen benötigt, anhand derer zum einen die Parameter des Systems eingestellt werden und zum anderen abschließend die Erkennungsleistung festgestellt wird. Über die Handschriftdaten hinaus, welche die Informationen über die Gestalt der Schrift beinhalten, wird zur Erzielung einer hohen Erkennungsrate auch syntaktisches Wissen über die zugrundeliegende Sprache hinzugezogen. Im Falle der Worterkennung besteht dieses Wissen aus einer Liste von legalen Wörtern der Sprache (Vokabular), bei der Satzerkennung darüber hinaus ggf. aus Sprachmodellen, welche das Wissen über die Häufigkeit bestimmter Wortfolgen beinhalten. Als Grundlage für das Verständnis der Experimente und Ergebnisse dieser Arbeit geben die an den Systemüberblick anschließenden Abschnitte eine Beschreibung der Art, Herkunft und Größe aller hier eingesetzten Datenbasen, Vokabulare und Sprachmodelle.

Anschließend werden die Größen definiert, anhand derer die Erkennungsleistung des Systems gemessen wird. Den Abschluß dieses Kapitels bilden die wichtigsten mit diesem System auf unterschiedlichen Erkennungsaufgaben erzielten Ergebnisse.

4.1 Überblick

Das in dieser Arbeit vorgestellte System entspricht in seiner Architektur dem klassischen hierarchischen Aufbau eines Handschrifterkenners, der in vielen Off-line und On-line Erkennungssystemen wiederzufinden ist (siehe Abbildung 4.1). Unterschiede

zwischen heutigen Erkennern bestehen in der Regel lediglich in der Konzeption innerhalb der einzelnen Komponenten der Systeme. Das ursprüngliche Eingabesignal, die zeitlich geordnete Folge der Datenpunkte $\{(x(t), y(t), p(t))\}_{t\in\{1\ldots T\}}$ mit $p(t) \in \{0, 1\}$ (Stiftzustand), durchläuft während des Erkennungsprozesses mehrere Verarbeitungsstufen, bestehend aus Vorverarbeitung und eigentlicher Erkennung. Mit jeder Verarbeitungsstufe wird das Eingabesignal hierarchisch von einer punkt-basierten Repräsentation zu Einheiten einer höheren Ebene, wie Zuständen, einzelnen Zeichen und ganzen Wörtern, zusammengesetzt.

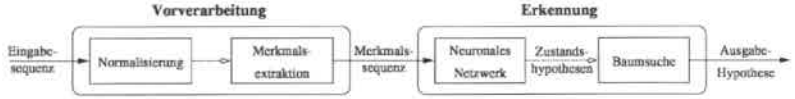


Abbildung 4.1: Die Verarbeitungsstufen des Systems

Die unterschiedlichen bei der Handschrifterkennung auftretenden Probleme, die in Kapitel 1.5 verdeutlicht wurden, werden jeweils durch entsprechenden Entwurf einer oder mehrerer der Verarbeitungsstufen gelöst. Unerwünschte Variabilität des Eingabesignals, verursacht durch die verwendete Hardware oder den Schreiber selbst, wird z.B. soweit wie möglich im Zuge der Vorverarbeitung reduziert. Robustheit gegenüber verschiedenen Schreibstilen wird durch die Realisierung der Erkennungskomponente mittels eines lernenden neuronalen Netzes erreicht. Für Toleranz gegenüber fehlerhaften Eingaben mit degenerierten oder fehlenden Zeichen sorgt die vokabulargesteuerte Baumsuche mit entsprechendem Wissen über die zugrundeliegende Sprache.

Die Architektur des Gesamtsystems und seiner einzelnen Komponenten basiert auf folgenden grundlegenden Prinzipien, die bei der Entwicklung als Leitfaden dienten:

Implizite Segmentierung: Die Erkennungsstrategie des Systems entspricht dem analytischen Ansatz mit impliziter Segmentierung (siehe Kapitel 2.4.4), d.h. die Segmentierung und Erkennung der Eingabe erfolgen im selben Verarbeitungsschritt. Eine explizite Segmentierung der Eingabe in Einzelzeichen ist nicht erforderlich.

Bewahrung der zeitlichen Information: Der zeitliche Charakter des Eingabesignals wird durch alle Verarbeitungsstufen beibehalten. Die Information über die chronologische Abfolge der einzelnen Datenpunkte bildet die Basis für den Erkennungsprozeß [MFW95b].

Erkennung auf Wortebene: Erkennungsentscheidungen und Optimierungen des Systems werden, außer im Fall der Einzelzeichenerkennung, auf der Wortebene durchgeführt. Einzelne Zeichen sind lediglich eine Zwischenrepräsentation während des Erkennungsprozesses.

Integration kontextuellen Wissens: Kontextuelles Wissen über gültige Wörter (Vokabular) und wahrscheinliche Wortfolgen (Sprachmodelle) der zu erkennenden Sprache sind direkt in den Erkennungsprozeß integriert.

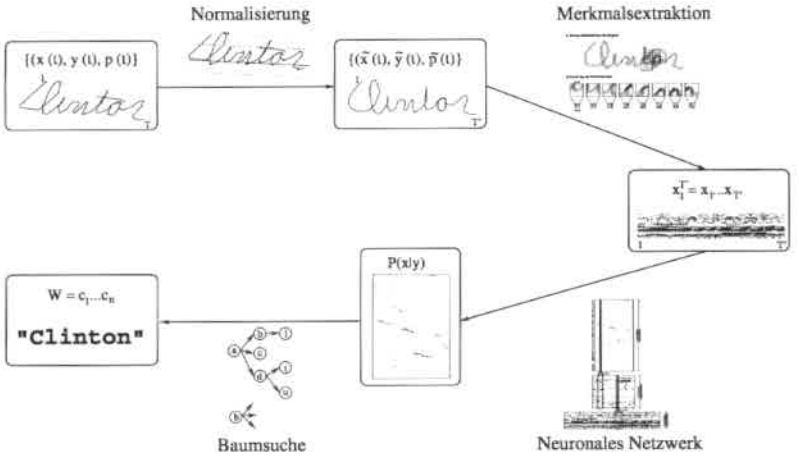


Abbildung 4.2: Das Gesamtsystem im Überblick

Vermeidung früher Entscheidungen: Endgültige Entscheidungen bezüglich der Segmentierung und Erkennung einer Eingabe, die in den nachfolgenden Verarbeitungsschritten nicht mehr korrigiert werden können, sind soweit wie möglich an das Ende des Erkennungsprozesses verlagert. Die in der Vorverarbeitung berechneten Merkmale orientieren sich nahe an der ursprünglichen punktbasierten Eingabe, die Erzeugung komplexerer Merkmale und Repräsentationen bleibt nachfolgenden Verarbeitungsschritten überlassen.

Eine Architektur für alle Aufgaben: Durch den streng hierarchischen Aufbau ist mit ein und derselben Architektur die Erkennung von Einzelzeichen, isolierten Wörtern bis hin zu Sätzen möglich.

Lernen durch Beispiele: Das Wissen über die möglichen Schreibweisen aller Zeichen erlangt das System durch überwachtes Lernen anhand einer großen Datenbasis handschriftlicher On-line Daten.

Abbildung 4.2 gibt einen detaillierteren Überblick über die Verarbeitungsstufen und die dabei entstehenden Zwischenrepräsentationen des vollständigen Systems, ausgehend vom ursprünglichen Eingabesignal bis hin zur Ausgabe. Die Stifttrajektorie eines Zeichens, Wortes oder Satzes wird als Punktsequenz $\{(x(t), y(t), p(t))\}_{t\in\{1\ldots T\}}$ mit $p(t) \in \{0, 1\}$ auf dem Eingabegerät aufgezeichnet. Diese Folge wird während der **Vorverarbeitung** normalisiert und durch Extraktion relevanter Merkmale in die weiterhin zeitlich geordnete Folge $x_1^{T'} = x_1 \ldots x_{T'}$ neuer N-dimensionaler Merkmalsvektoren $x_t = (f_1(t), \ldots, f_N(t))$ transformiert. Die Normalisierung entfernt unerwünschte Variabilität wie Rotation, Zittern, Zeichenneigungen usw. aus dem Signal, die entweder durch das Eingabegerät oder den Schreiber selbst verursacht wird. Die Merk-

malsvektoren kombinieren, über die bloße x- und y-Information der ursprünglichen Punkte hinausgehend, lokale Merkmale, die aus dem Datenpunkt selbst und seinen zeitlichen Nachbarn extrahiert werden, wie z.B. Schreibrichtung oder Winkel in einem Datenpunkt, mit globalen Merkmalen, die in Form eines Grauwertbildes mit sehr geringer Auflösung den räumlichen Kontext des Punktes repräsentieren [MFW94].

Nach der Vorverarbeitung wird, am Anfang der Merkmalssequenz beginnend, jeweils ein zeitlicher Ausschnitt $x_{t_1}^{t_2}$ der Eingabe betrachtet und für alle Zeichenklassen c_i mittels eines **neuronalen Netzes** die a posteriori Wahrscheinlichkeit $P(c_i|x_{t_1}^{t_2})$ geschätzt, daß das jeweilige Zeichen bzw. ein Teil davon in diesem Ausschnitt beobachtet wird. Ist die Berechnung der Wahrscheinlichkeiten für einen Ausschnitt beendet, wird der nachfolgende Ausschnitt $x_{t_1+1}^{t_2+1}$ entsprechend betrachtet, so daß sich letztendlich für jede Ausgabeklasse des neuronalen Netzes eine zeitliche Sequenz von a posteriori Wahrscheinlichkeiten ergibt. Das hier verwendete neuronale Netz ist ein Multi-State Time Delay Neural Network (MS-TDNN), ein mehrschichtiges Netz mit lokalen Verbindungen und gemeinsamen Gewichten über die Zeit [MB94, MFW95a]. MS-TDNNs, ursprünglich für Spracherkennungsaufgaben entwickelt, haben sich als sehr effektiv bei der Verarbeitung sequentieller Signale, wie sie hier vorliegen, erwiesen. Erst eine solche Architektur ermöglicht die Beibehaltung der sequentiellen Natur der Eingabedaten anstelle der in der Off-line Erkennung üblichen bildlichen Repräsentation.

Die durch das MS-TDNN erzeugten a posteriori Wahrscheinlichkeiten werden in klassenbedingte Wahrscheinlichkeiten umgewandelt und als Emissionswahrscheinlichkeiten der Wort-HMMS des Vokabulars interpretiert. Diese Wortmodelle sind durch Zusammenfassung gemeinsamer Präfixe in einer Baumstruktur angeordnet. Aus dieser Baumstruktur wird mittels einer effizienten **Strahlensuche** die wahrscheinlichste Ausgabehypothese des Systems bestimmt [MFW96].

4.2 Datenbasen

Datenbasen mit entsprechenden On-line Handschriftdaten bilden den Grundstein für die Entwicklung und Evaluierung eines On-line Handschrifterkenners. Da im Vergleich zur On-line Erkennung der Off-line Erkennung in den zurückliegenden Jahren erheblich mehr Aufmerksamkeit gewidmet wurde, existieren hier bereits zahlreiche Datenbasen mit umfangreichen Off-line Daten. Anhand dieser standardisierten und für alle frei verfügbaren Datenmenge können verschiedene Off-line Erkenner objektiv miteinander verglichen werden. Für die On-line Erkennung befindet sich im Rahmen des UNIPEN Projekts [GSP+94] auf Initiative des Technical Committee 11 der International Association for Pattern Recognition (IAPR) eine solche standardisierte Datenbasis im Aufbau (siehe Kapitel 10.2.6). Derzeit steht sie erst in einer sehr fehlerhaften und vorläufigen Version zur Verfügung und kann daher noch nicht sinnvoll eingesetzt werden, was die Vergleichbarkeit unterschiedlicher On-line Systeme erschwert.

	UKA	CMU	MIT	Summe
Ziffern:				
# Schreiber	102	198	159	459
# Zeichen	1985	4581	1590	8156
Großbuchstaben:				
# Schreiber	98	198	159	455
# Zeichen	3103	11764	4134	19001
Kleinbuchstaben:				
# Schreiber	103	198	159	460
# Zeichen	3001	11715	4134	18850
Einzelwörter:				
# Schreiber	114	215	159	488
# Wörter	6942	12575	2099	21616
Wortfolgen:				
# Schreiber	-	161	-	
# Sätze	-	8024	-	

Tabelle 4.1: Die Datenbasen im Überblick

Aufgrund dieses Mangels kommen hier für die Entwicklung und Evaluierung drei Datenbasen aus unterschiedlichen Quellen zum Einsatz. Alle drei Datenbasen zusammen enthalten mehr als 46000 Einzelzeichen, 21600 Einzelworte und 8000 Wortfolgen von insgesamt mehr als 500 Schreibern. Diese Datenmenge ist zwar nur klein im Vergleich zu den für analoge Erkennungsaufgaben verwendeten Off-line Datenbasen, ermöglicht aber dennoch eine fundierte Aussage über die Leistungsfähigkeit des Systems. Jede der drei Datenbasen besteht jeweils aus Einzelzeichen und Einzelworten, die Datenbasis CMU zusätzlich aus Wortfolgen bzw. ganzen Sätzen. Tabelle 4.1 gibt einen Überblick über die Verteilung der Daten. Aufgrund der Herkunft und eingesetzten Aufnahme-Hardware repräsentieren diese Datenbasen sowohl nationale Besonderheiten als auch unterschiedliche technische Gegebenheiten. An ihnen können also alle wichtigen Eigenschaften des Systems, wie Schreiberunabhängigkeit, Hardware-Unabhängigkeit usw. überprüft werden.

Die Datenbasen sind schreiberdisjunkt in je eine Trainings-, Kreuzvalidierungs- und Testmenge unterteilt, wobei die Unterteilung für Ziffern, Buchstaben, Wörter und Wortfolgen jeweils getrennt und unabhängig voneinander erfolgte. Da außer im Falle der Datenbasis MIT zwischen den Schreibern große Unterschiede in der Anzahl der gesammelten Wörter bestehen, wurden für die Testmengen überwiegend die Schreiber mit einer ähnlich großen Datenmenge ausgewählt, um ein ausgeglichenes Ergebnis über alle Schreiber zu erhalten. Die Daten der restlichen Schreiber ist den Trainings- und Validierungsmengen zugeordnet, wobei die Validierungsmenge jeweils in etwa die gleiche Größe wie die zugehörige Testmenge aufweist.

Die folgenden Abschnitte beschreiben die für die Datensammlung verwendete Aufnahme-Hardware und stellen die drei Datenbasen und ihre Aufteilung in

Trainings-, Validierungs- und Testmenge im Detail vor.

4.2.1 Aufnahme-Hardware

Die Sammlung der in dieser Arbeit verwendeten Datenbasen wurde mit Hilfe zweier unterschiedlicher, kommerziell verfügbarer Digitalisiertabletts durchgeführt, die hier kurz beschrieben werden.

Das **Wacom SD-421E** ist ein DIN-A4 Grafiktablett mit elektrostatischer Oberfläche, auf der bei Bedarf ein Blatt Papier fixiert werden kann. Die Eingabe erfolgt mit einem kabellosen Stift entweder direkt auf der glatten Oberfläche des Tabletts, wobei die Schrift dann für den Schreiber auf einem externen Bildschirm visualisiert werden muß, oder ggf. auf dem aufgelegten Papier. Beim Schreiben auf Papier kann ein spezieller, schreibender Stift verwendet werden, so daß die Eingabe tatsächlich direkt auf dem Papier sichtbar ist, was ein natürlicheres Schreibgefühl ermöglicht. Die Position und der aktuelle Status des Stiftes (abgehoben oder abgesetzt) wird durch Erzeugung elektro-magnetischer Resonanzen in dem Stift festgestellt. Die Abtastrate kann von 2 bis 200 Punkten pro Sekunde variiert werden. Die Auflösung beträgt 50 Zeilen pro Millimeter bei einer Abtastgenauigkeit von 0.15 Millimetern. Dieses Tablett wurde für die Datenbasen UKA und CMU verwendet, wobei die Sammlung teils mit aufgelegtem Papier erfolgte und teils ohne.

Das **Wacom HD-648A** ist ein Grafiktablett mit integriertem LCD-Bildschirm, das lediglich in kleiner Stückzahl für interessierte Entwickler stiftbasierter Soft- und Hardware produziert wurde. Die Aufnahmetechnik mittels elektromagnetischer Resonanz entspricht der des obigen Wacom SD-421E Grafiktabletts. Die maximale Abtastrate beträgt 192 Punkte pro Sekunde bei einer Auflösung von 50 Zeilen pro Millimeter. Der LCD Bildschirm hat eine Auflösung von 640×480 Punkten bei einer Größe von 21.1×15.8 Zentimetern. Dieses Tablett wurde bei der Sammlung der MIT Datenbasis eingesetzt.

4.2.2 Die UKA Datenbasis

Die UKA Datenbasis wurde gezielt für die Entwicklung des hier vorgestellten Erkennungssystems an der Universität Karlsruhe gesammelt. Sie bestand ursprünglich lediglich aus Einzelzeichen und wurde im Laufe der Zeit um Einzelworte erweitert. Die Aufnahme der Daten erfolgte auf einem Wacom SD-421E Grafiktablett. Als Schreiber stellten sich Studenten und Mitarbeiter der Universität zur Verfügung Zusätzlich wurden auch Sonderveranstaltungen der Universität für Abiturienten zur Datensammlung genutzt.

Von jedem Schreiber wurde eine vorher abgesprochene Datenmenge aufgenommen, die entweder jeweils nur Einzelzeichen oder Einzelworte umfaßte oder aber eine Mischung aus beidem. Instruktionen für die Schreiber waren auf ein Minimum beschränkt, um möglichst realistische Daten zu erhalten. Die Schreiber wurden dazu angehalten, so natürlich wie möglich auf dem ihnen ungewohnten Grafiktablett zu

UKA (Einzelzeichen)		
0...9	# Schreiber	# Zeichen
Training	64	1 605
Validierung	19	190
Test	19	190
Summe	102	1 985
A...Z	# Schreiber	# Zeichen
Training	62	2 167
Validierung	18	468
Test	18	468
Summe	98	3 103
a...z	# Schreiber	# Zeichen
Training	65	2 013
Validierung	19	494
Test	19	494
Summe	103	3 001

Tabelle 4.2: Unterteilung der Einzelzeichen der UKA Datenbasis

schreiben. Insbesondere wurden keine Vorgaben über den zu verwendenden Schreibstil gemacht. Die Datenbasis weist folglich eine zufällige Verteilung von Schreibern mit Druck- und Kursivschrift auf. Die zu schreibenden Einzelworte wurden zufällig aus einem englischen Vokabular mit Wörtern des Wall Street Journals ausgewählt. Sie bestehen nur aus den Kleinbuchstaben „a"...„z" und Großbuchstaben „A"...„Z". Ziffern kommen nur als Einzelzeichen, d.h. nicht innerhalb von Wörtern vor. Deutsche Umlaute oder Sonderzeichen wurden nicht aufgenommen. Jedem Schreiber wurde die Größe der zu sammelnden Datenmenge freigestellt, es wurde aber darauf geachtet, daß eine Mindestanzahl von 30 Einzelworten erreicht wurde, sofern nicht nur Einzelzeichen gesammelt wurden.

Für die Durchführung der Experimente in dieser Arbeit wurden die Schreiber der Einzelworte anschließend manuell nach ihrem Schreibstil der Gruppe der Druckschriftschreiber (PRINTED) oder Kursivschreiber (CURSIVE) zugeordnet. Schreiber mit einer Mixtur aus Druck- und Kursivschrift wurden einheitlich als Kursivschreiber klassifiziert. Für jede Erkennungsaufgabe (Ziffern, Kleinbuchstaben, Großbuchstaben und Einzelworte) wurden die gesammelten Daten schreiberdisjunkt in je eine Trainings-, Kreuzvalidierung- und Testmenge unterteilt. Im Fall der Einzelworterkennung wurde dabei zusätzlich auch nach dem Schreibstil unterschieden. Insgsamt ergibt sich die in den Tabellen 4.2 und 4.3 dargestellte Aufteilung. Da ein und dasselbe Wort oftmals von mehreren Schreibern gesammelt wurde, gibt die letzte Spalte von Tabelle 4.3 die Anzahl der unterschiedlichen Worte an. Etwa 4 000 Einzelworte von Kursivschreibern der Trainingsmenge wurden abschließend für spätere Trainingszwecke auf Buchstabenebene etikettiert, d.h. mit den Buchstabengrenzen markiert (siehe auch Kapitel 6.4).

UKA (Wörter)			
PRINTED	# Schreiber	# Wörter	# verschiedene
Training	13	1653	923
Validierung	5	660	577
Test	5	314	314
Summe	23	2627	
CURSIVE	# Schreiber	# Wörter	# verschiedene
Training	50	2374	2374
Validierung	20	856	851
Test	21	1085	1083
Summe	91	4315	

Tabelle 4.3: Unterteilung der Einzelworte und Sätze der UKA Datenbasis

4.2.3 Die CMU Datenbasis

Die Sammlung dieser Datenbasis wurde unter analogen Bedingungen zur UKA Datenbasis durchgeführt, diesmal jedoch im Auftrag an der Carnegie Mellon University (CMU) in Pittsburgh, USA. Schreiber waren hier amerikanische Studenten und Mitarbeiter des Computer Science Department an der CMU. Es wurden ebenfalls das Wacom SD-421E Grafiktablett zur Aufnahme verwendet und nur minimale Instruk-

CMU (Einzelzeichen)		
0...9	# Schreiber	# Zeichen
Training	103	3728
Validierung	44	386
Test	51	467
Gesamt	198	4581
A...Z	# Schreiber	# Zeichen
Training	102	9580
Validierung	44	999
Test	52	1185
Gesamt	198	11764
a...z	# Schreiber	# Zeichen
Training	102	9509
Validierung	44	1009
Test	52	1197
Summe	198	11715

Tabelle 4.4: Unterteilung der Einzelzeichen der CMU Datenbasis

tionen für die Schreiber gegeben. Im Gegensatz zur UKA Datenbasis wurden fehlerhafte Daten, d.h. solche mit Schreibfehlern, Korrekturen, Durchstreichungen usw., manuell markiert und für spätere Verwendung in der Datenbasis belassen. In dieser

Arbeit werden diese fehlerhaften Daten nicht berücksichtigt (siehe Kapitel 10.2.1). Die Tabellen 4.4 und 4.5 zeigen die Aufteilung der Daten in Trainings-, Validierungs- und Testmenge.

Die Klassifikation der Schreiber nach ihrem Schreibstil wurde wiederum manuell durchgeführt. Wie in der UKA Datenbasis ist zu beobachten, daß die Anzahl der Kursivschreiber die der Druckschriftschreiber um den Faktor 3 überschreitet. Etwa 2 000 Einzelworte von Kursivschreibern der Trainingsmenge wurden auch hier manuell mit den Buchstabengrenzen versehen. Eine Unterteilung der Schreiber von Sätzen nach ihrem Schreibstil erfolgte nicht, da kaum ein Schreiber über die ganze Länge eines Satzes Druckschrift verwendet.

CMU (Wörter)			
PRINTED	# Schreiber	# Wörter	# verschiedene
Training	27	1 621	1 573
Validierung	12	399	396
Test	15	347	344
Summe	54	2 367	
CURSIVE	# Schreiber	# Wörter	# verschiedene
Training	98	8 628	7 367
Validierung	28	680	670
Test	35	900	882
Summe	161	10 208	
CMU (Sätze)			
CURSIVE	# Schreiber	# Sätze	# Wörter
Training	123	1 619	11 291
Validierung	40	138	1 930
Test	50	161	2 322
Summe	213	1 918	15 543

Tabelle 4.5: Unterteilung der Einzelworte und Sätze der CMU Datenbasis

4.2.4 Die MIT Datenbasis

Die MIT Datenbasis wurde zu Forschungszwecken am Massachusetts Institute of Technology (MIT) gesammelt, um verschiedene Erkennungsverfahren miteinander zu vergleichen [Kas95]. Nach Abschluß der Untersuchungen wurde die Datenbasis allen Interessenten im Internet frei zur Verfügung gestellt. Sie enthält Daten von insgesamt 159 Schreibern (Studenten des MIT), die gegen Bezahlung an der Sammelaktion teilgenommen haben. Jeder Schreiber wurde gebeten, auf einem Wacom HD-648A LCD-Tablett die Ziffern „0"...„9", jeweils alle Klein- und Großbuchstaben, 25 vorgegebene Ziffernkolonnen der Länge 5 und 53 Einzelworte, die ebenfalls vorgegeben wurden, in Druckschrift niederzuschreiben. Kursivschrift ist in dieser Datenbasis nicht vorhanden. Die Liste der Ziffernkolonnen und Einzelworte war für alle Schreiber identisch,

d.h. es kommen tatsächlich nur 25 unterschiedliche Ziffernkolonnen bzw. 53 Einzelworte in der Datenbasis vor. Die Liste der Ziffernkolonnen wurde so gewählt, daß alle

MIT (Einzelzeichen)		
0...9	# Schreiber	# Zeichen
Training	99	990
Validierung	20	200
Test	40	400
Gesamt	159	1590
A...Z	# Schreiber	# Zeichen
Training	99	2574
Validierung	20	520
Test	40	1040
Gesamt	159	4134
a...z	# Schreiber	# Zeichen
Training	99	2574
Validierung	20	520
Test	40	1040
Summe	159	4134

Tabelle 4.6: Unterteilung der Einzelzeichen der MIT Datenbasis

Ziffern möglichst gleich häufig auftreten, alle 100 Ziffernpaare berücksichtigt werden und alle Ziffern mindestens einmal am Anfang und einmal am Ende einer Ziffernkolonne stehen. Die Auswahl der 52 Einzelworte erfolgte anhand einer Statistik über die Häufigkeit des Auftretens bestimmter Buchstabenkombinationen in einem Vokabular von 33000 Wörtern. So sollten die häufigsten Kombinationen in den Wörtern vertreten sein und jeder Großbuchstabe mindestens einmal am Wortanfang stehen. Als Wortliste wurde aufgrund dieser Festlegung folgende gewählt:

Accountability	Agonizingly	Announcing	Approaching	Backing
Cafeteria	Commanding	Comparatively	Complex	Declaring
Decompress	Disqualified	Embraces	Fabulously	Frightfully
Fuzz	Geography	Governing	Hugging	Inconsequential
Industrialized	Invulnerable	Justifications	Kidding	Lump
Mate	Menu	Normalization	Omitted	Projections
Puff	Puzzlement	Quizzically	Rejuvenating	Revving
Seeker	Shadow	Skiing	Spoiling	Surrounded
Swab	Sympathetically	Taxi	Transform	Uncomfortably
Unexpected	Unworkable	Vanquish	Volcanic	Wobble
Xylophone	Yearbook	Zero		

Alle Ziffernkolonnen und Einzelworte wurden manuell mit den jeweiligen Ziffern- bzw. Buchstabengrenzen markiert. Da in dieser Arbeit Ziffern nur im Rahmen der Experimente zur Einzelzeichenerkennung berücksichtigt werden, finden die Ziffernko-

MIT (Einzelworte)			
PRINTED	# Schreiber	# Wörter	# verschiedene
Training	99	5247	53
Validierung	20	1060	53
Test	40	2120	53
Summe	159	8427	

Tabelle 4.7: Unterteilung der Einzelworte der MIT Datenbasis

lonnen hier keine Verwendung. Die Tabellen 4.6 und 4.7 zeigen die Aufteilung der Daten nach Schreibern in Trainings-, Kreuzvalidierungs- und Testmenge. Die Daten von 99 der insgesamt 159 Schreiber wurden der Trainingsmenge zugeordnet, 20 Schreiber der Kreuzvalidierungsmenge und die verbleibenden 40 Schreiber der Testmenge, so daß sich wie in den anderen Datenbasen eine schreiberdisjunkte Aufteilung ergibt.

4.3 Verwendete Vokabulare und Sprachmodelle

Für die Erkennung von Einzelworten und Wortfolgen verwendet das hier vorgestellte System Vokabulare, d.h. Listen von Wörtern, auf die die Erkennung eingeschränkt ist. Wörter, die nicht im Vokabular aufgenommen sind, können daher nicht erkannt werden. Da mit der Größe des Vokabulars auch die Verwechselbarkeit der Wörter steigt und somit die Erkennungsrate sinkt, werden bei einigen Experimenten dieser Arbeit vier Vokabulare mit unterschiedlicher Anzahl von Wörtern herangezogen. Die Größe der gewählten Vokabulare beträgt 5000, 10000, 20000 und 50000 Wörter. Tabelle 4.8 zeigt die vier Vokabulare im Überblick mit ihren im folgenden verwendeten Namen, der Anzahl der Wörter und der durchschnittlichen Anzahl von Zeichen pro Wort.

Name des Vokabulars	# Wörter	# Zeichen/Wort
WSJ_5k	5000	7.8
WSJ_10k	10000	8.0
WSJ_20k	20000	8.2
WSJ_50k	50000	8.4

Tabelle 4.8: Verwendete Vokabulare für Wort- und Satzerkennung

Die Auswahl der Wörter für die Vokabulare erfolgte in einem zweistufigen Prozeß. Zunächst wurden die etwa 3500 unterschiedlichen Wörter, die in den Testdaten der obigen Datenbasen vorhanden sind, in das Vokabular aufgenommen, um zu verhindern, daß bei den durchgeführten Experimenten Fehlerkennungen dadurch zustande kommen, daß Wörter sich nicht im Vokabular befinden. Die Erkennung von Wörtern außerhalb des gegebenen Vokabulars wird in dieser Arbeit also nicht berücksichtigt. Die für die gewünschte Endgröße des Vokabulars fehlenden Wörter werden danach

aus einem 77 223 Wörter umfassenden Vokabular ausgewählt, das die, nach ihrer Häufigkeit des Auftretens sortierten, verschiedenen Wörter aus 230 223 Sätzen mit insgesamt 4 910 039 Wörtern aus dem Wall Street Journal (WSJ) enthält. Es wurden also jeweils die häufigsten Wörter aus den Sätzen des Wall Street Journals zu dem endgültigen Vokabular hinzugenommen. Aufgrund dieser Aufteilung ist jedes Vokabular vollständig in dem nächstgrößeren Vokabular enthalten. Die Sätze wurden aus dem Wall Street Journal Continuous Speech Recognition Corpus des Linguistic Data Consortiums von 1994 entnommen.

Wie aus Tabelle 4.8 zu ersehen, steigt die durchschnittliche Anzahl der Zeichen pro Wort mit der Größe des Vokabulars. Dies ist darauf zurückzuführen, daß die Wörter nach der Häufigkeit des Auftretens und nicht zufällig aus dem WSJ Vokabular ausgewählt werden. Es ergibt sich hier also die Beobachtung, daß kurze Wörter häufiger in den Sätzen des Wall Street Journals vorkommen als lange Wörter. Zusätzlich wurden auf allen 77 223 Wörtern des Wall Street Journal Vokabulars auf Buchstabenebene das Bigramm WSJ_BIGRAM und Trigramm WSJ_TRIGRAM berechnet, das für einen Test des Erkenners ohne Vokabular verwendet wird.

Bei der Erkennung von Sätzen kann die Erkennungsrate durch den Einsatz eines Sprachmodells, welches die Häufigkeit des Auftretens von bestimmten Wortfolgen berücksichtigt, verbessert werden. Das in dieser Arbeit verwendete Sprachmodell WSJ_LM (Bigramm) mit einem Verzweigungsgrad (Perplexität) von 112 wurde ebenso wie obige Vokabulare aus den 230 223 Wall Street Journal Sätzen erzeugt.

4.4 Leistungsmessung des Systems

Das Treffen von Entwurfsentscheidungen und der Vergleich dieses Ansatzes mit anderen Systemen erfordern die Berechnung der Erkennungsleistung für die drei unterschiedlichen Aufgaben Einzelzeichen-, Einzelwort- und Satzerkennung. An vielen Stellen der folgenden Kapitel sind nach diesen drei Bereichen getrennt Erkennungsergebnisse des Systems angegeben. Sofern nicht anders angegeben, sind diese Ergebnisse jeweils für alle Aufgaben unter den gleichen Bedingungen erzeugt worden. Das Training des neuronalen Netzes erfolgt über alle drei verfügbaren Datenbasen hinweg, d.h. die zu einer Erkennungsaufgabe gehörenden Trainingsmengen der Datenbasen werden zu einer einzigen Trainingsmenge zusammengefaßt. Die Erkennungsleistung auf den Testmengen ist jeweils sowohl getrennt nach den Datenbasen als auch als Gesamterkennungsrate angegeben. So wird ersichtlich, ob sich bestimmte Änderungen auf alle Datenbasen gleichermaßen auswirken. Die angegebenen Erkennungsraten für Einzelworte sind für jede Datenbasis noch einmal unterteilt nach den Schreibstilen Druck- (PRINTED) und Kursivschrift (CURSIVE).

Die Erkennungsrate auf Einzelzeichen und -worten wird üblicherweise direkt als *Buchstaben-* bzw. *Worterkennungsrate*

$$\text{BE (bzw. WE)} = \frac{N_{\text{Korrekt}}}{N_{\text{Gesamt}}}$$

angegeben, wobei N_{Korrekt} die Anzahl der vom System korrekt klassifizierten und N_{Gesamt} die Gesamtzahl der getesteten Zeichen bzw. Wörter angibt. Im Fall der Worterkennung wird ein Wort als falsch gezählt, wenn bereits ein einziges Zeichen dieses Wortes falsch erkannt wurde.

Bei der kontinuierlichen Handschrifterkennung, d.h. der Erkennung von Wortfolgen, treten wegen Segmentierungsfehlern neben Wortverwechslungen zusätzlich noch Einfügungen und Auslassungen als Fehlerquelle auf. Daher wird, wie in der kontinuierlichen Spracherkennung üblich, bei Sätzen anstelle der Worterkennungsrate die *Wortakkuratheit* (word accuracy)

$$\text{WA} = \frac{N_{\text{Gesamt}} - N_{\text{Einfügungen}} - N_{\text{Auslassungen}} - N_{\text{Verwechslungen}}}{N_{\text{Gesamt}}}$$

als Erkennungsleistung für einen Satz angegeben. N_{Gesamt} ist die Anzahl aller Wörter in dem Satz, $N_{\text{Einfügungen}}$ gibt die Anzahl der Worteinfügungen an, $N_{\text{Auslassungen}}$ die Anzahl der Auslassungen und $N_{\text{Verwechslungen}}$ die Anzahl der Verwechslungen. Die Summe $N_{\text{Einfügungen}} + N_{\text{Auslassungen}} + N_{\text{Verwechslungen}}$ wird allgemein als *Editierdistanz* bezeichnet [WF74]. Sie spezifiziert die minimale Anzahl der benötigten Editieroperationen Einfügen, Löschen und Vertauschen, um die vom Erkenner produzierte Hypothese in den Referenzsatz zu überführen. Analog zur Wortakkuratheit WA wird die *Buchstabenakkuratheit* BA für die Einzelworterkennung definiert, wenn kein Vokabular beim Test verwendet wird.

Da bei der Berechnung der Wortakkuratheit (Buchstabenakkuratheit) im Falle der Einzelworterkennung (Einzelzeichenerkennung) keine Auslassungen oder Einfügungen auftreten können und damit WA=BE (BA=BE) ist, wird in dieser Arbeit durchgehend, unabhängig von der Erkennungsaufgabe, die Erkennungsrate als Wortakkuratheit (Buchstabenakkuratheit) angegeben. Die Begriffe Akkuratheit und Erkennungsrate werden hier also synomym gebraucht. Als Standardvokabular der Wort- und Satzerkennung wird das Vokabular WSJ_20K verwendet.

4.5 Erkennungsergebnisse des Systems

In den nachfolgenden Kapiteln werden an zahlreichen Stellen Ergebnisse von unterschiedlichen Erkennungsaufgaben und unter unterschiedlichen Bedingungen präsentiert. Die dort durchgeführten Experimente dienen unter anderem dazu, die beste Konfiguration der Parameter des Systems zu bestimmen. Für die einzelnen Erkennungsaufgaben ergeben sich in manchen Fällen unterschiedliche Einstellungen, die in einer geringfügigen Verbesserung der Ergebnisse für diese Aufgabe resultieren. Dies gilt in besonderem Maße für die Einzelzeichenerkennung. Da jedoch die Gesamtleistung des Systems über alle Aufgaben hinweg im Mittelpunkt steht, werden für die Evaluierung des Systems für die Einzelzeichen-, Wort- und Satzerkennung jeweils die wichtigsten Einstellung gleichgehalten. Da die On-line Einzelzeichenerkennung nur bedingt von praktischem Interesse ist, sind dies diejenigen Einstellungen, die bei Einzelworten jeweils das beste Erkennungsergebnis erzielt haben.

Einzelzeichen	UKA	CMU	MIT	Gesamt
Ziffern	97.9%	97.0%	95.3%	**96.5%**
Großbuchstaben	93.4%	90.5%	94.9%	**92.7%**
Kleinbuchstaben	89.7%	89.8%	93.4%	**91.1%**
Einzelworte	UKA	CMU	MIT	Gesamt
	PRI / CUR	PRI / CUR	PRI	
BA ohne Vokabular	85.0%/70.6%	80.1%/74.3%	83.3%	**79.0%**
WA ohne Vokabular	37.3%/14.2%	24.7%/17.8%	36.0%	**26.8%**
BA mit Vokabular WSJ_20K	99.0%/95.4%	98.2%/96.8%	97.3%	**97.0%**
WA mit Vokabular WSJ_20K	97.1%/90.8%	92.8%/92.2%	94.7%	**93.4%**
Sätze mit WSJ_20K	UKA	CMU	MIT	Gesamt
WA ohne Sprachmodell	-	81.7%	-	**81.7%**
WA mit Bigramm	-	86.6%	-	**86.6%**

(PRI = PRINTED, CUR = CURSIVE)

Tabelle 4.9: Beste Erkennungsraten des Systems für die Einzelzeichen-, Wort- und Satzerkennung

Tabelle 4.9 faßt nun die besten mit dieser Einstellung erreichten Ergebnisse auf den Testmengen für die wichtigsten Erkennungsaufgaben zusammen. Diese Ergebnisse werden in den folgenden Kapiteln jeweils zum Vergleich herangezogen, um eine Bewertung einzelner Entwurfsentscheidungen zu ermöglichen. Sie geben die letztendlich erreichte tatsächliche Erkennungsleistung an. Über alle Datenbasen hinweg, die unterschiedliche Aufnahmebedingungen, Nationalitäten und insbesondere eine große Vielfalt unterschiedlicher Schreibstile repräsentieren, werden hervorragende Erkennungsraten erzielt, selbst bei großen Vokabularen mit 20 000 Wörtern.

Eine detailliertere Analyse dieser Ergebnisse ist in Kapitel 9 zu finden. Für diese Ergebnisse wurde das System jeweils auf allen verfügbaren Trainingsdaten der zugehörigen Erkennungsaufgabe trainiert.

Kapitel 5

Vorverarbeitung

5.1 Einleitung

Die Vorverarbeitung des ursprünglichen Eingabesignals ist die erste Aufgabe eines Handschrifterkenners und hat sowohl auf die Architektur der nachfolgenden Erkennungsschritte als auch die Erkennungsrate des Gesamtsystems entscheidenden Einfluß. Dennoch wird in vielen Systemen diesem Verarbeitungsschritt wenig Aufmerksamkeit gewidmet, da die Entwicklung einer robusten Vorverarbeitung ein aufwendiger Prozeß ist, der durch Treffen vieler Entwurfsentscheidungen und die manuelle Optimierung einer großen Anzahl unterschiedlicher Parameter geprägt ist [GP93]. Daß sich der Aufwand dieser Entwicklung jedoch lohnt, zeigt die große Varianz der Erkennungsergebnisse, wenn lediglich ein einziger Parameter der Vorverarbeitung, wie z.B. die Abtastrate des Signals, variiert wird. Auch in anderen Systemen konnte durch einfache Verbesserungen der Vorverarbeitung und Beibehaltung der übrigen Erkennungsschritte eine deutliche Reduzierung der Fehlerrate erreicht werden [DHU97]. Bedingt durch die große Anzahl muß die Optimierung[1] dieser Parameter allerdings in vielen Fällen suboptimal durchgeführt werden, da nur wenige Parameter unabhängig von den anderen Parametern sind. Bei der Optimierung der Parameter wird daher meist eine eigentlich nicht vorhandene Unabhängigkeit angenommen, d.h. es werden nicht alle tatsächlich möglichen Parameterkombinationen getestet.

Die einzelnen Schritte der Vorverarbeitung werden entsprechend ihrem Ziel in *Normalisierung* und *Merkmalsextraktion* unterteilt. Die Normalisierung beinhaltet Schritte zur Entfernung oder zumindest Verringerung aufnahmebedingten Rauschens und schreiberabhängiger Variabilität des Eingabesignals, die den Erkennungsprozeß erschweren. Die Merkmalsextraktion befaßt sich mit der Berechnung relevanter Merkmale aus dem normalisierten Signal unter Verwendung a priori Wissens über das zugrundeliegende Problem, um die Erkennungsaufgabe zu vereinfachen. Wichtigstes Prinzip hinter allen diesen Verarbeitungsschritten ist es, zum einen den zeitlichen Charakter des ursprünglichen Signals beizubehalten und zum anderen Fehlentschei-

[1] „Optimierung" bedeutet hier das Testen der Auswirkungen einer Parametereinstellung auf die Erkennungsrate des Systems und Wahl des sich daraus ergebenden besten Wertes für den Parameter.

dungen, die in nachfolgenden Verarbeitungsschritten nicht mehr korrigiert werden können, zu vermeiden.

Obwohl bereits zahlreiche Vorverarbeitungstechniken entwickelt wurden, existiert noch keine allgemein gültige Lösung für dieses schwierige Problem. Viele der hier vorgestellten Normalisierungstechniken und Merkmale finden in ihrer prinzipiellen Form auch in anderen Systemen ihren Einsatz, sind jedoch ggf. für die speziellen Ziele dieses Systems modifiziert und optimiert. Mit der vorliegenden Arbeit werden erstmals die wichtigsten im Eingabesignal auftretenden Effekte bei der Normalisierung vollständig berücksichtigt. In bisherigen Systemen wurde jeweils nur eine kleine Teilmenge dieser Effekte betrachtet und das Hauptaugenmerk auf die nachfolgenden Erkennungsschritte gerichtet. Die in Abschnitt 5.3.3 vorgestellte nicht-lineare Interpolation durch Bézierkurven zur Kompensation von Abtastfehlern wird in dieser Arbeit erstmals für die Handschrifterkennung eingesetzt. Erstmals werden hier auch Merkmale präsentiert, die über die in anderen Systemen üblichen, zeitlich lokalen Merkmale hinausgehen. So werden von der Zeit unabhängige Informationen über die Trajektorie mit in die Eingaberepräsentation einbezogen. Dadurch wird auf Einzelworten eine relative Fehlerreduktion von etwa 15% erreicht.

Die folgenden Abschnitte beschreiben die Normalisierung und Merkmalsextraktion im Detail am Beispiel der Worterkennung. Die Spezialfälle der Normalisierung und Merkmalsextraktion für Einzelzeichen und Satzerkennung sind am Ende der Abschnitte beschrieben. Aufgrund der bereits erwähnten großen Anzahl relevanter Parameter kann dabei nicht jeder Einfluß einer Entwurfsentscheidung auf die Erkennungsrate getestet und hier angegeben werden. Die wichtigsten Entscheidungen sind jedoch an entsprechender Stelle durch Experimente belegt.

5.2 Die Vorverarbeitung im Überblick

Ausgangspunkt der Vorverarbeitung ist das originale Eingabesignal als zeitliche Folge 3-dimensionaler Merkmalsvektoren $\{(x(t), y(t), p(t))\}_{t\in\{1\ldots T\}}$ mit $p(t) \in \{0, 1\}$, die die aufgezeichneten Datenpunkte repräsentieren. Jeder dieser Vektoren spezifiziert für einen bestimmten Zeitpunkt t, an welchem Punkt $(x(t), y(t))$ sich der Stift befunden hat und ob dort der Stift abgehoben $(p(t) = 0)$ oder aufgesetzt $(p(t) = 1)$ war. Dieses Signal wird nun in der Vorverarbeitung schrittweise in eine Folge $x_1^{T'} = x_1 \ldots x_{T'}$ neuer N-dimensionaler Merkmalsvektoren $x_t = (f_1(t), \ldots, f_N(t))$ transformiert, die die Eingabe für die nachfolgenden Erkennungsschritte bildet (Abbildung 5.1). Während der Normalisierung wird die Dimension der Merkmalsvektoren nicht verändert, sondern lediglich die Anzahl der Datenpunkte (z.B. durch Neuabtastung und Löschen verzögerter Striche) und deren Position (z.B. durch Basisliniennormalisierung und Glättung) modifiziert, so daß sich die neue Folge $\{(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))\}_{t\in\{1\ldots T'\}}$ ergibt. Erst während der Merkmalsextraktion werden zu jedem nach der Normalisierung verbleibenden Vektor zusätzliche Merkmale hinzugefügt bzw. die bestehenden Merkmale ersetzt und somit die Dimension der Vektoren auf N erhöht, um die anschließende

Erkennung zu erleichtern [MFW94].

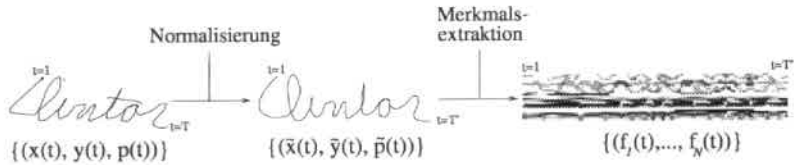


Abbildung 5.1: Die Repräsentationen der Vorverarbeitung

5.3 Normalisierung

Das aufgenommene Eingabesignal weist in der Regel eine große Anzahl unerwünschter Variabilitäten auf, die sich auf die nachfolgende Merkmalsextraktion auswirken und den Erkennungsprozeß erschweren. Selbst wenn derselbe Schreiber auf derselben Hardware mehrfach die gleiche handschriftliche Eingabe tätigt, unterscheiden sich die jeweiligen Eingabesignale teilweise erheblich. Noch größer werden diese Unterschiede bei unterschiedlichen Schreibern unter Verwendung unterschiedlicher Hardware. Folgende auftretende Effekte im Eingabesignal, die durch die Hardware, den Schreiber selbst oder beide zusammen verursacht werden, sind dabei zu beobachten:

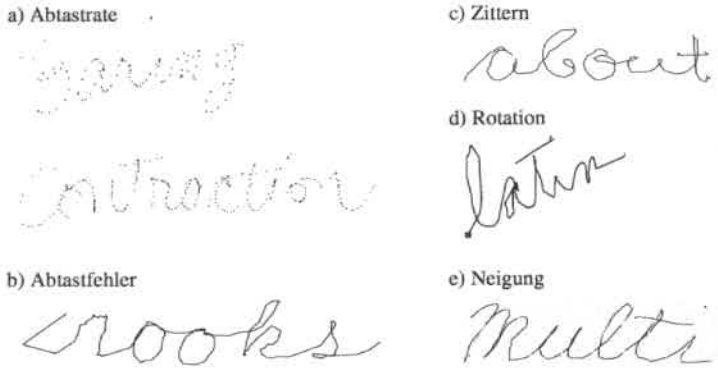


Abbildung 5.2: Auftretende Effekte im Eingabesignal

Unterschiedliche Punktabstände: Unterschiedliche Schreibgeschwindigkeiten und von der jeweiligen Hardware abhängige Abtastraten resultieren in stark variierenden Abständen zwischen aufeinanderfolgenden Datenpunkten. Obwohl unter Umständen der Stift mehrfach derselben Trajektorie gefolgt ist, wird dadurch die zu erkennende Eingabesequenz für diese Trajektorie durch unterschiedlich viele Datenpunkte repräsentiert. Dieser Effekt tritt dabei nicht

nur zwischen verschiedenen Wörtern auf, sondern auch innerhalb eines einzigen Wortes, wenn durch Zögern beim Schreiben die Schreibgeschwindigkeit variiert, wie Abbildung 5.2a an zwei Beispielen aus den Datenbasen zeigt

Abtastfehler: Selbst bei konstanter Abtastrate der eigentlichen Aufnahmehardware, ist bei vielen Hardware-Konfigurationen, z.B. durch Übertragung der Eingabe über eine serielle Schnittstelle zum Rechner und gleichzeitig starke CPU-Belastung, kein tatsächlich zeitlich konstanter Fluß der Datenpunkte gewährleistet. Die Folge sind Abtastlücken, die im Extremfall ganze Teile eines Zeichens umfassen und dieses unleserlich werden lassen. Abbildung 5.2b zeigt ein Beispiel aus den Datenbasen, bei dem aufgrund solcher Abtastfehler der erste Buchstabe nur durch insgesamt 4 Datenpunkte repräsentiert wird.

Zittern: Abtastfehler, zu geringe Auflösung der Hardware und das ungewohnte Schreiben auf der zu glatten Oberfläche der Aufnahmehardware sind Ursache für das oftmals sehr zittrige Schriftbild der aufgenommen Daten (siehe z.B. Abbildung 5.2c).

Rotation: Bei den meisten handschriftlichen Eingaben, bei denen nicht explizit Schreiblinien vorgegeben wurden, treten Rotationen, d.h. Abweichungen der Schreibrichtung von der Horizontalen, in der einen oder anderen Richtung auf. In den Datenbasen sind Rotationen bis nahezu 45° zu finden (Abbildung 5.2d).

Größenunterschiede: Ebenso wie die Schreibrichtung hängt auch die Schreibgröße stark von den persönlichen Gewohnheiten eines Schreibers ab, sofern keine Schreiblinien vorgegeben wurden. Im allgemeinen ist die Schrift auf der glatten Fläche der Aufnahmehardware größer als auf Papier.

Neigungswinkel: Insbesondere bei kursiver Handschrift auf glatter Oberfläche sind häufig Abweichungen des Neigungswinkels der einzelnen Buchstaben von der Vertikalen zu beobachten (Abbildung 5.2e). Die Neigungsrichtung ist unter anderem davon abhängig, ob die Eingabe von einem Links- oder Rechtshänder stammt. Im ersten Fall sind eher Neigungen nach links, im zweiten Fall eher nach rechts zu beobachten. Darüber hinaus werden diese Abweichungen auch erst durch die Rotationsnormalisierung verursacht, wenn die einzelnen Zeichen zwar senkrecht geschrieben wurden, jedoch die ganze Eingabe von der Horizontalen abweicht. Beim Korrigieren der Eingabe in eine horizontale Lage werden die ursprünglich senkrechten Zeichen nun mit einer Neigung in die eine oder andere Richtung versehen.

Verzögerte Striche: Ein lediglich bei der On-line Erkennung auftretendes Problem sind Abweichungen von der üblichen strikten Schreibweise von links nach rechts. Abgesehen von nachträglichen Korrekturen bereits geschriebener Eingabe, die in dieser Arbeit nicht berücksichtigt wird, treten solche Abweichungen in vielen Wörtern auf, in denen „i"-Punkte oder „t"-Striche u.ä. geschrieben werden müssen. Kein Problem bereiten diese verzögerten Punkte oder Striche, wenn

sie unmittelbar hinter dem zugehörigen Zeichen geschrieben wurden, also die strikte Schreibordnung nicht verletzt wurde. Gesondert berücksichtigt werden müssen sie jedoch, wenn sie erst zu einem späteren Zeitpunkt, z.B. nach Beendigung des gesamten Wortes, eingefügt werden und damit in der zeitlichen Abfolge an der „falschen" Stelle stehen.

Ziel der Normalisierung ist es, diese unerwünschten Effekte soweit wie möglich auf ein Minimum zu reduzieren. Abbildung 5.3 zeigt die einzelnen Normalisierungsschritte, die im nachfolgenden beschrieben werden, am Beispiel eines Wortes der Datenbasis.

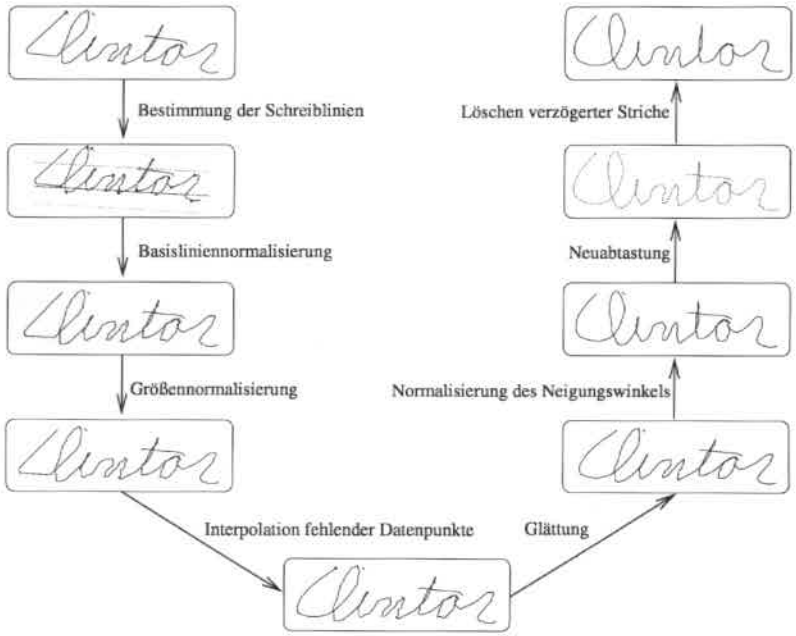


Abbildung 5.3: Normalisierung des Wortes „Clinton"

5.3.1 Bestimmung der Schreiblinien und Basisliniennormalisierung

Viele Erkennungsmethoden, insbesondere solche, die bildliche Repräsentationen der Eingabe oder andere geometrische Informationen zur Erkennung heranziehen, setzen voraus, daß die Größe, Lage und Orientierung der einzelnen Zeichen innerhalb ihrer Zeichenklasse für alle geschriebenen Textsegmente zumindest annähernd gleich

sind. Durch Normalisierungen der Eingabe auf der Ebene von Wörtern kann diese Voraussetzung erfüllt werden, sofern die Basislinie und Kernhöhe des Wortes identifiziert werden.[2] Die Kernhöhe ist dabei definiert als die Größe aller Kleinbuchstaben ohne Über- und Unterlängen, wie z.B. „a", „e" oder „o". Die Basislinie ist eine der vier Schreiblinien, die die drei Schreibzonen eines Wortes begrenzen (Abbildung 5.4). Sie bildet die Grenze zwischen dem Kern der Klein- und Großbuchstaben ohne Unterlängen und den Unterlängen anderer Buchstaben. Die Kernlinie grenzt den Kern nach oben hin gegen die Oberlängen ab. Unter- und Oberlinie bestimmen jeweils das Ende der Unter- bzw. Oberlängen. Die Kernhöhe kann direkt aus dem Abstand der Kernlinie von der Basislinie abgelesen werden, die Lage und Orientierung des Wortes ergeben sich aus der Lage und Orientierung der zugehörigen Basislinie. Die Basis- und Kernlinie sind die hauptsächliche Informations-, die Bestimmung derselben aber auch Fehlerquelle für die Normalisierung. Eine ungenaue Bestimmung der Schreiblinien wirkt sich unmittelbar auf alle weiteren Verarbeitungsschritte aus und hat großen Einfluß auf die Erkennungsrate des Systems. Eine möglichst genaue Bestimmung ist also Voraussetzung für ein robustes Erkennungssystem.

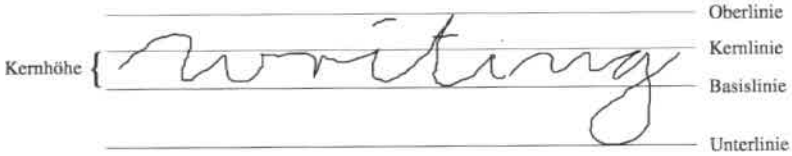


Abbildung 5.4: Definition der Schreiblinien und Kernhöhe

Viele der derzeit eingesetzten Algorithmen zur Bestimmung der Basis- und Kernlinie haben gemeinsam, daß ihre Berechnung lediglich auf den lokalen y-Minima und y-Maxima der Stifttrajektorie beruht und nicht auf der gesamten vorhandenen Punktmenge der Eingabe. Die Beschränkung auf diese Extremalpunkte ist dadurch gerechtfertigt, daß zum einen fast alle Zeichen mindestens ein Minimum und Maximum besitzen, diese also innerhalb eines Wortes in regelmäßigen Abständen auftreten, und sich die Extrema in der Regel nur in unmittelbarer Nähe der Schreiblinien befinden.

Kann ein Extremum keiner der Schreiblinien zugeordnet werden, handelt es sich meist entweder um einen Endpunkt, d.h. einen Punkt, an dem der Stift abgenommen wurde, oder um einen inneren Extremalpunkt, der für die gegebene x-Koordinate nicht auch zugleich absolutes Minimum bzw. Maximum ist. Daher bleiben solche Extrema bei der Bestimmung der Schreiblinien in dieser Arbeit unberücksichtigt. Eine weitere Beobachtung ist, daß Minima überwiegend im Bereich der Basis- und Unterlinie auftreten, Maxima hingegen im Bereich der Kern- und Oberlinie.

Ein häufig verwendeter Ansatz in der Literatur, um aus dieser Menge von y-Extrema die Basis- und Kernlinie zu bestimmen, ist die Verwendung linearer Regres-

[2] In manchen Systemen werden jedoch auch Schreiblinien vorgegeben, die beim Schreiben in etwa eingehalten werden müssen (siehe z.B. [Rao95]).

sionen durch die Extremalpunkte [Sch95, Sen95]. Dabei wird die Regressionsgerade durch die lokalen Minima berechnet und als Basislinie bestimmt. Die Regressionsgerade durch die lokalen Maxima legt entsprechend die Kernlinie fest. Nachteil dieses Verfahrens ist, daß die Basis- und Kernlinie unabhängig voneinander bestimmt werden und daher in ihrer Orientierung teils erheblich voneinander abweichen, insbesondere bei kurzen Wörtern oder Wörtern, die lediglich am Anfang oder Ende Unter- bzw. Oberlängen haben. In solchen Fällen stellt sich die Frage, welche der beiden Linien eine exaktere Annäherung der tatsächlichen Orientierung des Wortes ist. Außerdem bleiben aufgrund der Linearität ggf. Krümmungen des Wortes unberücksichtigt.

Diese Nachteile werden durch den hier verwendeten Ansatz, der auf dem in [LB94] vorgestellten Algorithmus basiert, vermieden. Grundidee dieses Algorithmus ist es, mittels des sog. „Elastic Matchings" nicht nur die eigentlich benötigte Basis- und Kernlinie zu bestimmen, sondern alle vier Schreiblinien gleichzeitig und in Abhängigkeit voneinander, und zusätzlich auch Krümmungen zu berücksichtigen. Die vier Schreiblinien bilden ein abstraktes Modell eines Wortes, welches es an das aktuelle Wort anzupassen gilt. Dazu wird eine Energiefunktion bestehend aus 2 Komponenten entwickelt. Die erste Komponente beschreibt die Qualität der Anpassung des Modells auf das Wort, d.h. wie gut das Modell auf die vorhandenen Minima und Maxima paßt. Die zweite Komponente hingegen beschreibt die interne Spannung des Modells, die angibt, wie stark das Modell gedehnt werden muß, um auf die Daten zu passen. Um diese Energiefunktion zu minimieren, wird der sog. „Expectation-Maximization" (EM) Algorithmus verwendet [DLR77]. Das Modell wird hier betrachtet, als ob es die gegebenen Daten durch einen stochastischen Prozeß erzeugen würde. Die Energiefunktion wird interpretiert als die negative *log*-Wahrscheinlichkeit, daß die Daten durch das gegebene Modell erzeugt werden.

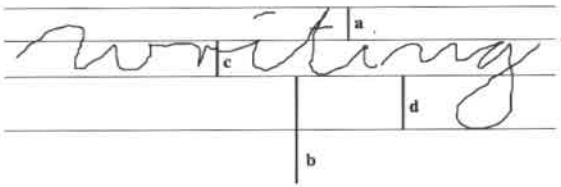


Abbildung 5.5: Das Wortnormalisierungsmodell

Das hier verwendete geometrische Modell der Wortstruktur basiert auf den vier Schreiblinien. Die Energiefunktion beschreibt, wie gut die beiden unteren Linien auf die lokalen Minima und die beiden oberen Linien auf die lokalen Maxima passen. Die Schreiblinien sind definiert als Polynome zweiten Grades der folgenden Form:

$$f_i(x) \;=\; k\,(x-\bar{x})^2 + s\,(x-\bar{x}) + y_i \qquad (i=0,\ldots,3)$$

Der Parameter k beschreibt die Krümmung, s die Rotation und $\bar{x}$ den horizontalen Schwerpunkt der Extrema der Schreiblinien. Diese drei Parameter sind für alle vier

Linien identisch, d.h. eine Krümmung oder Rotation wirkt sich auf alle Linien gleichermaßen aus. Jede Linie hat hingegen ihren eigenen vertikalen Verschiebungsparameter y_i, welcher gegeben ist durch:

$$\begin{aligned} y_0 &= b - d \\ y_1 &= b \\ y_2 &= b + c \\ y_3 &= b + c + a \end{aligned}$$

Die Parameter a, b, c und d sind wie in Abbildung 5.5 definiert. a ist der Abstand der Kernlinie von der Oberlinie, b der Abstand der Basislinie von der x-Achse, c die Kernhöhe und d der Abstand der Basislinie von der Unterlinie.

Abbildung 5.6: Die Rotationsnormalisierung

Sind die Schreiblinien mit obigem Verfahren bestimmt, gibt der gemeinsame Parameter s der vier Schreiblinien die Rotation des Wortes, d.h. dessen Abweichung von einer horizontalen Orientierung, an. Anhand einer einfachen geometrischen Rotation des Wortes wird die gewünschte horizontale Orientierung hergestellt, wie in Abbildung 5.6 gezeigt ist. Welchen Einfluß die Basisliniennormalisierung auf die Erkennungsrate hat, zeigt Tabelle 5.1 am Beispiel der Worterkennung. Hier ist nur eine leichte Verbesserung der Gesamterkennungsrate zu beobachten, da insgesamt nur einige wenige Worte in der Datenbasis größere Rotationen aufweisen. Das Ergebnis zeigt auch, daß die neuronale Erkennerarchitektur bereits robust gegenüber kleineren Rotationen der Eingabe ist.

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
ohne	98.7%	90.4%	94.5%	91.7%	93.3%	**92.8%**
mit	97.1%	90.8%	92.8%	92.2%	94.7%	**93.4%**

Tabelle 5.1: Erkennungsraten mit und ohne Basisliniennormalisierung der Trainings- und Testdaten

In der in [LB94] vorgestellten Wortnormalisierung wurde vor deren Anwendung die Eingabe bereits in eine annähernd horizontale Orientierung gebracht, da sonst bei großen Rotationen die Verteilung der Extrema auf die Schreiblinien durch den EM Algorithmus nicht gefunden werden konnte. Die Bestimmung der Vorabrotation wurde durch eine lineare Regression durch die lokalen Minima ermittelt. Ein Großteil der

Rotationsnormalisierung wurde also bereits durch zusätzliche Verfahren gelöst. Nach Bestimmung der Schreiblinien wurde dann die Orientierung der Eingabe ein zweites Mal anhand des ermittelten Parameters s korrigiert. In dieser Arbeit hingegen ist keine vorhergehende Rotationsnormalisierung der Eingabe erforderlich. Statt dessen werden ebenfalls durch lineare Regressionen die ungefähre Rotation und Position der Schreiblinien ermittelt und damit vor Anwendung des EM Algorithmus die Modellparameter s, a, b, c, d bereits so initialisiert, daß das Modell in etwa auf die Eingabe paßt. Der EM Algorithmus führt anschließend die Feinabstimmung der Parameter durch. So können auch trotz Rotationen mit 40 Grad und mehr die Schreiblinien durch diesen Ansatz bestimmt werden. Die hier vorgestellte Wortnormalisierung muß die Eingabe nicht zweimal korrigieren und läuft daher schneller ab als das Verfahren aus [LB94].

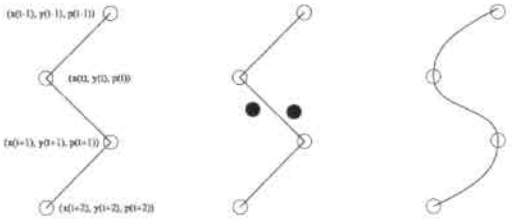
5.3.2 Größennormalisierung

Ziel der Größennormalisierung ist es, über alle handschriftlichen Eingaben hinweg eine soweit wie möglich einheitliche Zeichengröße innerhalb der jeweiligen Zeichenklassen zu erhalten. Unabhängig davon, in welchem Kontext ein bestimmtes Zeichen steht, d.h. mit welchen anderen Zeichen es in einer Folge steht, soll es immer dieselbe Größe haben. Eine einfache Skalierung der Eingabe anhand der Gesamthöhe auf eine vorgegebene Größe erfüllt hier nicht den gewünschten Zweck, da die Gesamthöhe selbst davon abhängig ist, ob die Eingabe Unter- oder Oberlängen bzw. beides aufweist. Es würden sich folglich drei verschiedene Zeichengrößen ergeben. Helfen kann hier die bereits berechnete Kernhöhe h des Wortes, die unabhängig von der Existenz von Unter- und Oberlängen ist. Skaliert wird also derart, daß alle Wörter nach der Größennormalisierung dieselbe Kernhöhe $\tilde{h}$ haben (Abbildung 5.7), also zumindest alle Zeichen ohne Unter- und Oberlängen die gleiche Größe. Die Gesamthöhe des Wortes variiert weiterhin von Wort zu Wort.

Abbildung 5.7: Normalisierung der Schreibgröße anhand der Kernhöhe

5.3.3 Nicht-lineare Interpolation fehlender Datenpunkte

Ist die Abtastrate der Aufnahmehardware zu gering, wird sehr schnell geschrieben oder treten Abtastfehler auf, durch die Datenpunkte verloren gehen, ergibt sich in einigen Fällen eine sehr ungenaue Trajektorie, bei der manche Zeichen nur durch sehr wenige Datenpunkte repräsentiert werden. Mit diesen wenigen Punkten ist eine glatte und der tatsächlichen Trajektorie entsprechende Darstellung der Eingabe



a) Lineare Interpolation b) Kontrollpunkte c) Bézier-Polygonzug

Abbildung 5.8: Interpolation fehlender Datenpunkte mittels Bézierkurven

nicht möglich, wie die Beispiele in Abbildung 5.9b oben zeigen. Bei der Neuabtastung, die im nächsten Abschnitt beschrieben ist, werden zwar bei weit auseinanderliegenden Punkten Zwischenpunkte eingefügt, jedoch nur mittels einer linearen Interpolation, die die natürliche Trajektorie nur unzureichend reproduzieren kann (Abbildung 5.8a). Daher werden in diesen Fällen bereits vor der Neuabtastung zwischen Punkten, die einen Höchstabstand voneinander überschreiten, zusätzliche Punkte eingefügt. Der verwendete Algorithmus betrachtet jeweils zwei benachbarte Datenpunkte $(x(t), y(t), p(t))$ und $(x(t+1), y(t+1), p(t+1))$ und überprüft den Abstand. Ist der Abstand $\sqrt{x(t)^2 + x(t+1)^2}$ größer als ein vorgegebener Schwellwert, werden zusätzlich sog. Kontrollpunkte zwischen diesen Punkten berechnet (Abbildung 5.8b). Die Berechnung der Kontrollpunkte beruht u. a. auf der Richtung, aus der die Trajektorie in den Punkt $(x(t), y(t), p(t))$ hineinläuft, und der Richtung, in der der Punkt $(x(t+1), y(t+1), p(t+1))$ verlassen wird. Durch die ursprünglichen zwei Punkte und die neuen Kontrollpunkte wird nun abschließend eine Bézier-Kurve gelegt (Abbildung 5.8c) [AM91], entlang derer die neuen zusätzlichen Punkte der Trajektorie gewählt werden. Abbildung 5.9a zeigt das Wort „Clinton" vor und nach dem Einfügen der zusätzlichen Punkte. Weitere Beispiele der Rekonstruktion sind in Abbildung 5.9b zu sehen. Das hier verwendete Verfahren stammt aus dem Bereich der geometrischen Modellierung und kommt in dieser Form z.B. auch in technischen Zeichenprogrammen zum Einsatz, um aus einer Sequenz gegebener Punkte glatte Kurven zu erstellen. Tabelle 5.2 zeigt den Effekt dieser nicht-linearen Kurvenrekonstruktion auf die Erkennungsrate.

Diese Art der Normalisierung wird in dieser Arbeit zum ersten Mal in einem Handschrifterkenner eingesetzt. Auf den hier verwendeten Datenbasen ist der Effekt dieser Normalisierung gering, da dort gravierende Abtastfehler, wie in Abbildung 5.9 gezeigt, nur selten auftreten. Deutlicher zeigt sich der Effekt dieser Normalisierung anhand einer kleinen Datenbasis, die bei Demonstrationen dieses Handschrifterkenners auf einem drucksensitiven Bildschirm aufgezeichnet wurden. In dieser Datenbasis, der auch die Beispiele in Abbildung 5.9 entstammen, ist die Anzahl derartiger Abtastfehler deutlich höher. Auf dieser Datenbasis ergibt sich durch die nicht-lineare Interpolation

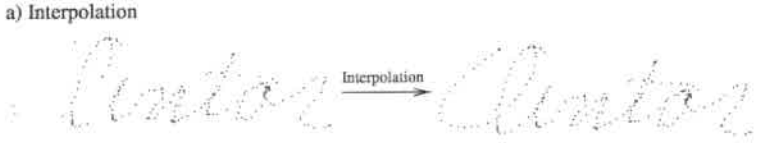


Abbildung 5.9: Interpolation fehlender Datenpunkte mittels Bézierkurven

eine Verbesserung der Erkennungsrate von 66% auf 83% bei einem 20 000 Wörter umfassenden Vokabular.

5.3.4 Glättung

Um leichtes Zittern, verursacht durch Abtastfehler oder zittrige Schreibweise, auszugleichen, wird eine einfache Glättung der Stifttrajektorie vorgenommen. Die neuen

Abbildung 5.10: Die Glättung

Koordinaten $(x_{\text{neu}}(t), y_{\text{neu}}(t))$ eines Datenpunktes $(x_{\text{alt}}(t), y_{\text{alt}}(t))$ ergeben sich dabei aus dem Mittelwert der jeweils N zeitlich vorausgehenden und nachfolgenden Punkte und dem gewichteten Punkt selbst:

$$x_{\text{neu}}(t) = \frac{x_{\text{alt}}(t-N) + \cdots + x_{\text{alt}}(t-1) + \alpha x_{\text{alt}}(t) + x_{\text{alt}}(t+1) + \cdots + x_{\text{alt}}(t+N)}{2N+\alpha}$$

$$y_{\text{neu}}(t) = \frac{y_{\text{alt}}(t-N) + \cdots + y_{\text{alt}}(t-1) + \alpha y_{\text{alt}}(t) + y_{\text{alt}}(t+1) + \cdots + y_{\text{alt}}(t+N)}{2N+\alpha}$$

Der Gewichtungsfaktor α wird für jeden Punkt heuristisch in Abhängigkeit des Winkels zwischen dem vorhergehenden und nachfolgenden Kurvensegment berechnet und vermindert das unerwünschte Abrunden wichtiger Spitzen oder enger Kurven in der Trajektorie, d.h. wenn sich abrupt die Schreibrichtung ändert. Je kleiner der Winkel

	UKA	CMU	MIT	Gesamt
0...9:				
ohne	97.8%	97.0%	95.1%	**96.4%**
mit	97.9%	97.0%	95.3%	**96.5%**
A...Z:				
ohne	92.7%	90.0%	94.8%	**92.3%**
mit	93.4%	90.5%	94.9%	**92.7%**
a...z:				
ohne	87.7%	89.4%	92.5%	**90.3%**
mit	89.7%	89.8%	93.4%	**91.1%**
Wörter:	PRINTED/CURSIVE	PRINTED/CURSIVE	PRINTED	
ohne	97.2%/90.8%	92.4%/92.3%	93.6%	**92.7%**
mit	97.1%/90.8%	92.8%/92.2%	94.7%	**93.4%**

Tabelle 5.2: Erkennungsraten mit nicht-linearer Interpolation und ohne

in dem Punkt ist, desto größer wird α gewählt. Abbildung 5.10 zeigt das Wort „Clinton" vor und nach der Glättung. Aus Tabelle 5.3 sind die Auswirkungen des Glättens auf die Erkennungsrate verschiedener Erkennungsaufgaben zu ersehen.

	UKA	CMU	MIT	Gesamt
0...9:				
ohne	97.8%	97.1%	95.1%	**96.4%**
mit	97.9%	97.0%	95.3%	**96.5%**
A...Z:				
ohne	92.5%	90.4%	93.6%	**92.0%**
mit	93.4%	90.5%	94.9%	**92.7%**
a...z:				
ohne	89.5%	89.1%	90.5%	**89.7%**
mit	89.7%	89.8%	93.4%	**91.1%**
Wörter:	PRINTED/CURSIVE	PRINTED/CURSIVE	PRINTED	
ohne	98.1%/90.7%	93.1%/91.9%	93.7%	**93.0%**
mit	97.1%/90.8%	92.8%/92.2%	94.7%	**93.4%**

Tabelle 5.3: Erkennungsraten mit und ohne Glättung der Trainings- und Testdaten

5.3.5 Normalisierung des Neigungswinkels

Nicht alle später berechneten Merkmale sind invariant gegenüber einer eventuell vorhandenen Neigung der Zeichen eines Wortes nach links oder rechts, die entweder den Schreiber selbst oder die Basisliniennormalisierung als Quelle hat. Dies betrifft insbesondere Merkmale, die z.B. auf der Schreibrichtung oder einer bildlichen Repäsen-

tation der Eingabe basieren. Für eine Korrektur des Neigungswinkels muß dieser zunächst bestimmt werden. Dazu wird die Trajektorie der Eingabe durchlaufen und jeweils der Winkel zwischen der Geraden durch zwei aufeinanderfolgende Punkte und der Horizontalen gemessen und in ein Winkelhistogramm eingetragen [Sch95, Bur97]. Die Winkel werden zusätzlich mit dem Abstand der zugehörigen Punkte gewichtet, so daß sich längere Geraden stärker im Histogramm auswirken als kurze. Zusätzlich wird das Histogramm geglättet.

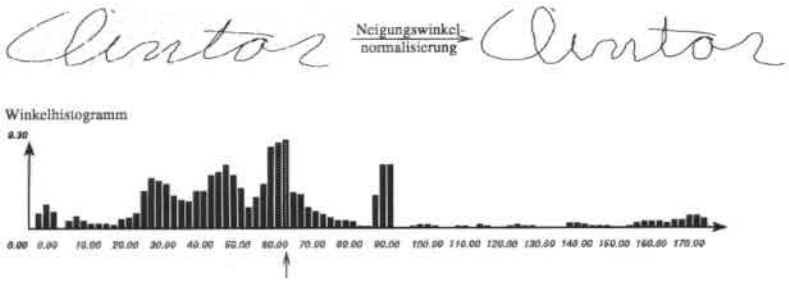


Abbildung 5.11: Beispiel der Neigungswinkelnormalisierung

Charakteristischerweise ist im Bereich des gesuchten Neigungswinkels eine Häufung im Histogramm zu beobachten. Allerdings treten diese Häufungen teilweise auch bei anderen Winkeln, insbesondere im Bereich von ±45 Grad des Neigungswinkels auf. Um zu vermeiden, daß aufgrund fehlerhaft bestimmter Winkel erst durch diese Normalisierung eine Neigung der Eingabe eingebracht wird, wird der gesuchte Neigungswinkel eher vorsichtig gewählt. Ausgehend von 90 Grad wird das Histogramm jeweils in beide Richtungen nach dem ersten Auftreten eines lokalen Maximums durchsucht. Von diesen beiden Maxima wird entsprechend das größere als Neigungswinkel gewählt. Es werden allerdings nur Neigungen im Bereich von 60 bis 120 Grad berücksichtigt, da größere Neigungen nach links oder rechts kaum auftreten und damit auch die oben erwähnten Häufungen durch die Diagonalen ausgeschlossen werden. Abbildung 5.11 zeigt die Neigungswinkelnormalisierung des Wortes „Clinton" mit zugehörigem Histogramm und gewähltem Neigungswinkel. Eine Gegenüberstellung der Erkennungsraten des Systems mit und ohne Neigungswinkelnormalisierung enthält Tabelle 5.4

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
ohne	97.5%	89.9%	93.9%	90.9%	93.9%	**92.6%**
mit	97.1%	90.8%	92.8%	92.2%	94.7%	**93.4%**

Tabelle 5.4: Erkennungsraten mit und ohne Neigungswinkelnormalisierung der Trainings- und Testdaten

Besonders deutlich wird der Effekt der Neigungswinkelkorrektur bei Daten, die eine Neigung nach links aufweisen. Dies ist häufig bei Linkshändern der Fall. Bei solchen Schreibern sind die größten Verbesserungen zu beobachten, da die Mehrheit aller Worte der Trainingsmenge eher eine Neigung nach rechts hat und somit Linksneigungen stark unterrepräsentiert sind. In Abbildung 5.12 sind Daten eines Linkshänders abgebildet, bei dem die Erkennungsrate durch die Neigungswinkelkorrektur von 81% auf 92% gesteigert werden konnte.

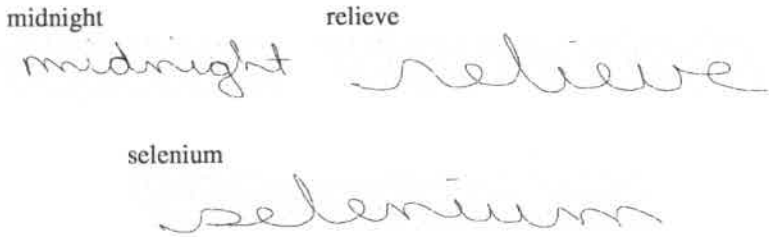


Abbildung 5.12: Beispiele mit Linksneigung

5.3.6 Neuabtastung

Die Anzahl der Datenpunkte für eine bestimmte Eingabe hängt zum einen von der Abtastrate der verwendeten Hardware und zum anderen von der Schreibgeschwindigkeit des Schreibers ab. Die Schreibgeschwindigkeit kann dabei auch innerhalb eines Zeichens oder Wortes variieren. Um eine Invarianz gegenüber diesem Effekt zu erreichen, d.h. eine möglichste gleiche Anzahl von Punkten für Zeichen mit gleicher Form, werden die originalen Punkte $\{(x(t), y(t), p(t)\}_{t\in\{1..T\}}$ der Eingabe, die ursprünglich gleiche Abstände in der Zeit hatten, durch neue Punkte $\{(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t)\}_{t\in\{1..T'\}}$ ersetzt, die den gleichen räumlichen Abstand aufweisen. Je nach der Abtastrate der Hardware entstehen dadurch mehr oder weniger Datenpunkte als in der Ausgangsfolge. Diese Art der Normalisierung wird in nahezu allen Systemen zur On-line Erkennung durchgeführt (siehe z.B. [GAL+91, Sen95, Sch95]). Für die Berechnung der

Abbildung 5.13: Die Neuabtastung

neuen Punkte wird eine einfache lineare Interpolation entlang der Bogenlänge der Trajektorie verwendet. Hierbei werden auch alle Strecken zwischen dem Absetzen und Wiederaufsetzen des Stiftes interpoliert. Der neue Abstand der aufeinanderfolgenden Datenpunkte ist in Abhängigkeit der Kernhöhe h als $\frac{h}{13}$ gewählt. D.h. wenn

eine senkrechte Linie von der Kernlinie zur Basislinie bestünde, so würde diese durch 13 Datenpunkte dargestellt. Abbildung 5.13 zeigt das Wort „Clinton" als Punktfolge vor und nach der Neuabtastung. Die deutliche Verbesserung der Erkennungsleistung durch die Neuabtastung ist in Tabelle 5.5 dargestellt.

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
ohne	95.2%	84.6%	83.0%	87.2%	89.6%	**87.9%**
mit	97.1%	90.8%	92.8%	92.2%	94.7%	**93.4%**

Tabelle 5.5: Erkennungsraten mit und ohne Neuabtastung der Trainings- und Testdaten

Die optimale Anzahl der Punkte a pro Kernhöhe zur Bestimmung des neuen Abstandes $\frac{h}{a}$ zweier Punkte wurde experimentell ermittelt. Für $a = 8 \ldots 18$ wurde jeweils ein Erkenner für Einzelworte trainiert und auf den Testmengen der Datenbasen UKA, CMU und MIT evaluiert. Abbildung 5.14 zeigt anhand der Gesamterkennungsrate auf diesen Datenbasen, daß für eine gegebene Netzarchitektur mit $a = 13$ tatsächlich die beste Erkennungsleistung erzielt wird (siehe auch Kapitel 6.5).

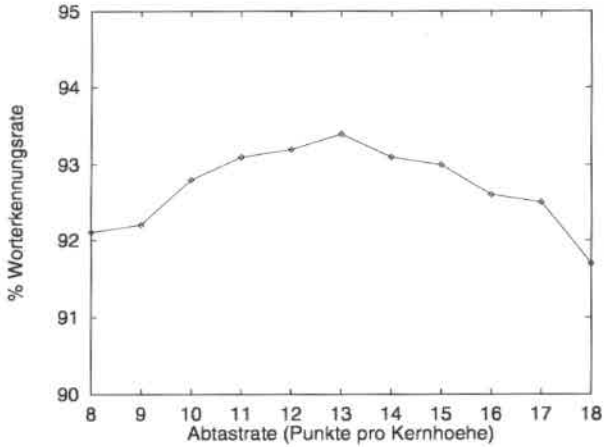


Abbildung 5.14: Worterkennungsraten für unterschiedliche Abtastraten

5.3.7 Detektion verzögerter Striche

Das Auffinden und Entfernen verzögerter Striche ist ein heuristischer Prozeß, der auf dem Wissen über die zugrundeliegende Erkennungsaufgabe beruht. So funktioniert diese Heuristik auch nur für die englische Sprache, da hier in der Regel verzögerte

Striche, sofern es sich nicht um sonstige nachträgliche Ergänzungen oder Korrekturen bereits geschriebener Wortteile handelt, nur bei den Buchstaben „i", „j" und „t" auftreten. In anderen Sprachen, z.B. Deutsch oder Französisch, würden in vielen Fällen auch bedeutungstragende verzögerte Striche, wie die Punkte bei Umlauten oder Akzente, gefunden und später aus der Eingabe entfernt. Im Gegensatz zu „i"-Punkten und „t"-Strichen geben diese verzögerten Striche den dazugehörigen Zeichen tatsächlich eine andere Bedeutung. „i", „j" und „t" hingegen sind auch ohne Punkt und Strich meist als solche zu erkennen. In den meisten heutigen Systemen mit integrierter Segmentierung und Erkennung wird das Problem der verzögerten Striche bislang ebenfalls durch Entfernung gelöst (siehe z.B. [GP93, HOKK93, Sch95, Sen95]). In Systemen, die eine explizite Segmentierung der Eingabe durchführen, ist ggf. auch eine Umsortierung der einzelnen Segmente möglich, so daß die zeitliche Ordnung wieder hergestellt wird [FNCB93].

Verzögerte Striche sind dadurch charakterisiert, daß sie sich meist in der oberen Schreibzone oder im Bereich der Kernlinie befinden. Darüber hinaus handelt es sich um eine sehr kurze Sequenz von Datenpunkten, die durch ein Abheben des Stiftes und eine Rückwärtsbewegung in einen bereits existierenden Teil der Eingabe eingeleitet wird. Die Rückwärtsbewegung muß dabei mindestens so weit sein, wie die Kernhöhe der Eingabe beträgt, um zu gewährleisten, daß der Strich nicht zu dem zuletzt geschriebenen Zeichen gehört, sondern zu einem tatsächlich früher bereits beendeten Zeichen. Es werden also die Eingabe nach der Existenz von kurzen Punktsequenzen mit diesen Merkmalen untersucht und die entsprechenden Punkte aus der Eingabe entfernt. Um die Information über die Existenz der gelöschten Striche nicht gänzlich zu verlieren, werden alle Bereiche der Eingabe, die sich unter einem gelöschten Strich befanden, markiert. Diese Markierungen werden bei der Merkmalsextraktion in einem zusätzlichen Merkmal verwertet. Tabelle 5.6 stellt die Erkennungsraten des Systems mit und ohne Entfernung der verzögerten Striche gegenüber. Die Ergebnisse bestätigen die Vermutung, daß verzögerte Striche bei Schreibern mit Kursivschrift eine größeres Problem darstellen als bei Druckschrift. Bei Druckschrift wird offensichtlich die übliche Schreibreihenfolge von links nach rechts seltener unterbrochen, so daß sich hier nur ein geringfügiger Effekt durch das Löschen der verzögerten Striche bemerkbar macht.

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
ohne	97.0%	90.5%	92.5%	88.1%	95.0%	**93.0%**
mit	97.1%	90.8%	92.8%	92.2%	94.7%	**93.4%**

Tabelle 5.6: Erkennungsraten mit und ohne Löschen verzögerter Striche in den Trainings- und Testdaten

5.3.8 Spezialfälle der Normalisierung

Die vorhergehenden Abschnitte beschreiben die einzelnen Normalisierungsschritte der Vorverarbeitung, wie sie für die Erkennung von Einzelworten verwendet wird. Für die Erkennung von Einzelzeichen und längeren Textsegmenten ergeben sich ein paar Spezialfälle, die hier kurz dargestellt werden.

Eine zuverlässige Bestimmung der Schreiblinien und der Neigung ist bei Einzelzeichen aufgrund der zu kurzen Eingabe in der Regel nicht möglich. Eine Normalisierung des Zeichens basierend auf diesen Schreiblinien und dem gefundenen Neigungswinkel hätte in den meisten Fällen also eher unerwünschte negative Effekte. Daher wird bei Einzelzeichen auf eine Basislinien- und Neigungswinkelnormalisierung verzichtet. Die Größennormalisierung erfolgt in diesem Fall anhand der absoluten Größe des Zeichens und nicht der Kernhöhe. Glättung, Neuabtastung und Interpolation fehlender Datenpunkte können unverändert durchgeführt werden. Die Existenz verzögerter Striche muß bei Einzelzeichen nicht überprüft werden, da diese nur bei Zeichenfolgen auftreten.

Bei der Erkennung längerer Textsegmente ist zusätzlich die Detektion von Zeilenumbrüchen erforderlich. Dies ist notwendig, da jede Zeile eigene Schreiblinien mit ggf. unterschiedlicher Krümmung oder Neigung besitzt und die Schreiblinienfindung bei Betrachtung mehrerer Zeilen gleichzeitig keine Möglichkeit hat, diese einzeln zu identifizieren. Daher werden das Eingabesignal vor dem ersten Normalisierungsschritt an den gefundenen Zeilenumbrüchen auseinandergeschnitten und die einzelnen Zeilen getrennt und unabhängig voneinander vorverarbeitet (einschließlich Merkmalsextraktion). Erst nach Vollendung der Vorverarbeitung werden die einzelnen Zeilen wieder miteinander zu einer einzigen Merkmalsfolge verknüpft. Die Detektion der Zeilenumbrüche ist hier ein einfacher heuristischer Prozeß, der überprüft, ob nach dem Abheben des Stiftes dieser um etwas nach unten versetzt bis zum Anfang der vorhergehenden Zeile zurückkehrt.

Bei langen handgeschriebenen Zeilen kommt es häufig vor, daß sich im Verlauf des Schreibens die Schreibgröße und ggf. auch die Krümmung verändert. Um diesem Effekt Rechnung zu tragen, wird daher die Schreiblinienfindung nicht über die ganze Zeile durchgeführt, sondern lediglich innerhalb eines vorgegebenen Fensters, das überlappend über die Zeile geschoben wird. Für jedes dieser Fenster existieren also eigene Schreiblinien, die dann zur Normalisierung herangezogen werden können. Lediglich die Rotationsnormalisierung wird anhand von Schreiblinien durchgeführt, die über die gesamte Zeile berechnet wurden.

5.3.9 Normalisierungsreihenfolge

Die Reihenfolge der einzelnen Normalisierungsschritte kann in begrenztem Rahmen frei gewählt werden. Es müssen lediglich einige wenige Reihenfolgen eingehalten werden, um das optimale Ergebnis zu erreichen. So sollte die Normalisierung des Neigungswinkels erst nach der Basisliniennormalisierung und der Glättung durchgeführt

werden, da sich unter Umständen erst durch die Basisliniennormalisierung ein ungewollter Neigungswinkel ergibt und die Glättung für ein gleichmäßigeres Winkelhistogramm sorgt. Zusätzlich sollte die Rekonstruktion fehlender Datenpunkte vor der Neuabtastung stattfinden, da sonst keine nicht-lineare Rekonstruktion mehr möglich ist. Je nach Wahl der Normalisierungsreihenfolge ergibt sich ggf. ein leicht anderes Resultat der Normalisierung, sie sollte also in jedem Fall während der Lernphase des Systems und der anschließenden Tests festgeschrieben werden. In der Regel folgt die Normalisierung in dem hier vorgestellten System daher der bereits in Abbildung 5.3 dargestellten Reihenfolge.

5.4 Merkmalsextraktion

Nach der Normalisierung sollte die Eingabe im wesentlichen frei von unerwünschter Variabilität sein. Unterschiedliche Trajektorien für ein bestimmtes Zeichen sollten sich nun nur durch unterschiedliche Schreibweisen des Zeichens ergeben. Die Extraktion relevanter Merkmale, auf denen die eigentliche Erkennung der Eingabe basiert, ist nach der Normalisierung die zweite Aufgabe der Vorverarbeitung.

Die Wahl der Eingaberepräsentation bestimmt die Architektur aller nachfolgenden Erkennungsschritte. Eine Umwandlung des normalisierten, zeitlichen Signals in ein statisches Grauwertbild ermöglicht zwar die Nutzung des großen Vorrats bereits existierender Algorithmen der Off-line Handschrifterkennung, zieht aber den Verlust wertvoller Informationen nach sich. Daher wird die prinzipielle Struktur des Signals in Form einer Sequenz von Merkmalsvektoren beibehalten.

Jeder Punkt $(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))$ der normalisierten Folge $\{(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))\}$ wird durch einen neuen N-dimensionalen Merkmalsvektor $x_t = (f_1(t), \ldots, f_N(t))$ ersetzt, so daß sich als Ergebnis der Vorverarbeitung die Folge $x_1^{T'} = x_1 \ldots x_{T'}$ ergibt (siehe Abbildung 5.15). Jeder dieser Merkmalsvektoren wiederum besteht aus sogenannten *lokalen Merkmalen* und zusätzlichen *globalen Merkmalen*. Die lokalen Merkmale werden für den jeweiligen Datenpunkt aus einem zeitlich begrenzten Kontext um diesen Punkt, d.h. den zeitlich unmittelbar vorausgehenden und nachfolgenden Punkten, berechnet. Es liegt hier zusätzlich zur zeitlichen auch eine räumliche Lokalität vor, da sich die unmittelbaren zeitlichen Nachbarn in einem begrenzten räumlichen Intervall um den Punkt befinden. Nachteil der zeitlichen Lokalität ist, daß dadurch in den Merkmalen keine Informationen über zeitlich weiter auseinanderliegende Zusammenhänge festgehalten werden können. Sollte sich also der aktuelle Punkt in unmittelbarer räumlicher Nähe von bereits zeitlich weiter zurückliegenden Punkten befinden, z.B. wenn der Stift einen bereits geschriebenen Teil kreuzt oder neben diesem entlang läuft, so bleibt diese Tatsache in den lokalen Merkmalen unberücksichtigt. Daher wurden die Merkmalsvektoren um globale Merkmale erweitert, die zwar weiterhin räumlich lokal sind, aber global in der Zeit [MFW94].

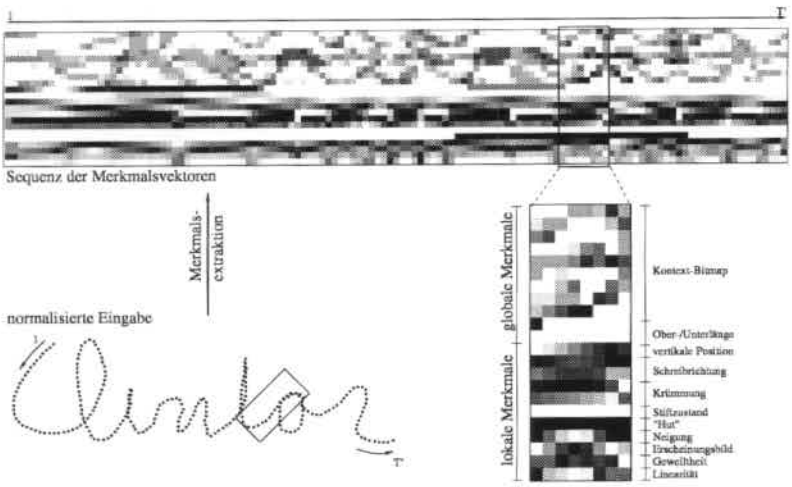


Abbildung 5.15: Merkmalsextraktion

5.4.1 Lokale Merkmale

Lediglich vier Informationsquellen werden bei der Berechnung der lokalen topologischen Merkmale für einen Merkmalsvektor x_t herangezogen: der Datenpunkt $(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))$ selbst, seine unmittelbaren zeitlichen Nachbarn $(\tilde{x}(t-M), \tilde{y}(t-M), \tilde{p}(t-M)), \ldots, (\tilde{x}(t+M), \tilde{y}(t+M), \tilde{p}(t+M))$, die während der Normalisierung identifizierten Schreiblinien und die Information über verzögerte Striche, die im Zuge der Normalisierung entfernt wurden. Ein bestimmtes Merkmal wird nicht in allen Fällen nur durch einen Wert kodiert, sondern teils auch mit zwei Werten. Teilweise beinhalten zwei unterschiedliche Merkmale eine ähnliche Information, sind also nicht unbedingt unabhängig voneinander.

Vertikale Position

Außer im Falle eines Einzelzeichenerkenners verhindern oder erschweren die absoluten Positionsangaben $(x(t), y(t))$ die verschiebungsinvariante Erkennung einzelner Zeichen, da zum einen der x-Wert von der Position des Zeichens im geschrieben Wort und zum anderen der y-Wert von der Existenz von Unterlängen im Wort abhängt. Daher werden diese beiden Werte nicht direkt als Merkmale in den Merkmalsvektor übernommen, sondern durch relative Positionsänderungen (siehe Schreibrichtung unten) und eine zusätzliche Angabe der vertikalen Position in Abhängigkeit der Basislinie ersetzt. Dieses Merkmal gibt den vertikalen Abstand $a(t) = |y(t) - b(x(t))|$ des aktuellen y-Wertes des Punktes $(x(t), y(t))$ von dem y-Wert $b(x)$ der Basislinie an der Position $x(t)$ an. $a(t)$ ist also positiv, wenn sich $(x(t), y(t))$ oberhalb der Basislinie

befindet, und negativ, wenn unterhalb. Um einen gleichmäßigen Wertebereich dieses Merkmals für alle Wörter zu erreichen, folgt abschließend eine Normalisierung der Werte mit der Kernhöhe des Wortes. Normalisiert wird derart, daß $a(t)$ den Wert 1 annimmt, wenn sich der Punkt $(x(t), y(t))$ auf der Kernlinie befindet. Insgesamt ergibt sich dadurch ein Wertebereich von ca. $[-1, +2]$ unter der Annahme, daß üblicherweise Ober- und Unterlängen ähnlich groß wie der Kern sind.

Diese Art der vertikalen Positionsangabe wird erst dadurch ermöglicht, daß bei der Basislinienbestimmung die Krümmung des Wortes berücksichtigt wird. Bei linearer Approximation der Basislinien wie in anderen Systemen würde ansonsten der Abstand durch die Krümmung beeinflußt. In anderen existierenden Systemen ist daher eine solche Kodierung der Position nicht möglich.

Schreibrichtung (Writing Direction)

Die aktuelle Schreibrichtung für den Merkmalsvektor x_t wird, wie in [GAL+91] erstmals vorgeschlagen, durch eine Schätzung des Richtungscosinus der Tangente im Punkt $(x(t), y(t))$ kodiert (siehe Abbildung 5.16a). Diese Parameter können auch als diskrete Annäherung an die erste Ableitungen $\frac{dx}{ds}$ und $\frac{dy}{ds}$ nach der Bogenlänge angesehen werden, wobei $ds = \sqrt{dx^2 + dy^2}$ ist:

$$\begin{aligned} \cos\alpha(t) &= \frac{\Delta x(t)}{\Delta s(t)} \\ \sin\alpha(t) &= \frac{\Delta y(t)}{\Delta s(t)} \end{aligned}$$

Dabei sind $\Delta s(t)$, $\Delta x(t)$ und $\Delta y(t)$ definiert als:

$$\begin{aligned} \Delta s(t) &= \sqrt{\Delta x^2(t) + \Delta y^2(t)} \\ \Delta x(t) &= x(t-1) - x(t+1) \\ \Delta y(t) &= y(t-1) - y(t+1) \end{aligned}$$

Diese Repräsentation wurde einer Kodierung der Schreibrichtung mit nur einem Wert vorgezogen, da sie keine Berechnung einer transzendenten Funktion erfordert und sich für eine glatte Kurve auch glatte Änderungen der Werte ergeben. Darüber hinaus bewegen sich die Werte dieser Merkmale in einem abgeschlossenen Bereich.

Krümmung (Curvature)

Ebenso wie die Schreibrichtung wird auch die Krümmung der Trajektorie an dem aktuellen Punkt durch 2 Werte kodiert, welche den Kosinus und Sinus des Winkels zwischen zwei aufeinanderfolgenden elementaren Kurvensegmenten angeben (siehe Abbildung 5.16b) [GAL+91]. Unter Verwendung trigonometrischer Funktionen können diese Werte einfach aus den obigen Richtungsmerkmalen berechnet werden:

$$\begin{aligned} \cos\beta(t) &= \cos\alpha(t-1) * \cos\alpha(t+1) + \sin\alpha(t-1) * \sin\alpha(t+1) \\ \sin\beta(t) &= \cos\alpha(t-1) * \sin\alpha(t+1) - \sin\alpha(t-1) * \cos\alpha(t+1) \end{aligned}$$

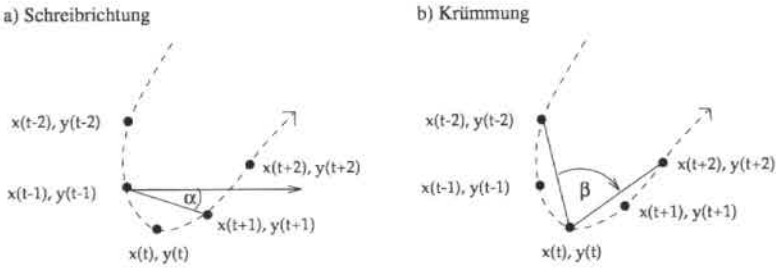


Abbildung 5.16: Berechnung von Schreibrichtung (a) und Krümmung (b)

Stiftzustand (Pen-up / Pen-down)

Dieser Wert gibt an, ob zum Zeitpunkt t der Stift auf der Schreibfläche aufgesetzt oder abgehoben war. Er definiert also den sichtbaren ($p(t) = 1$) und unsichtbaren Bereich ($p(t) = 0$) der Trajektorie. Unsichtbare Teile der Trajektorie ergeben sich durch die lineare Interpolation zwischen den Punkten, an denen der Stift abgehoben und anschließend an anderer Stelle wieder aufgesetzt wurde. Bei Verwendung entsprechender Eingabegeräte kann anstelle der linearen Interpolation die tatsächliche Bewegung des Stifts über der Schreiboberfläche verfolgt werden, sofern sich der Stift nicht zu weit von dieser entfernt. Liefert das Eingabegerät anstelle der binären Information über den Stiftzustand detailliertere Druckinformationen, kann der Wertebereich dieses Merkmals entsprechend angepaßt werden. Da jedoch nicht alle Geräte drucksensitiv sind, beschränken wir uns hier auf die rein binäre Kodierung.

Das „Hut"-Merkmal

Während der Normalisierung wurden aus der Stifttrajektorie verzögerte Striche wie „i"-Punkte und „t"-Striche entfernt, alle horizontalen Bereiche jedoch, die von solch einem verzögerten Strich überlagert waren, festgehalten. Diese Information findet nun in Form des Hut-Merkmals seinen Einzug in den Merkmalsvektor, das binär spezifiziert, ob sich der aktuelle Datenpunkt im horizontalen Bereich eines gelöschten verzögerten Striches befindet oder nicht (siehe Abbildung 5.17) [Sch95].

Erscheinungsbild (Aspect)

Für dieses und die folgenden Merkmale werden die in Abbildung 5.18 definierten Größen verwendet [Sch95]. Diese Merkmale beinhalten eine globalere Sicht des Liniensegments um den aktuellen Datenpunkt. Für die Berechnung wird das Fenster $(\tilde{x}(t-2), \tilde{y}(t-2), \tilde{p}(t-2)), \ldots, (\tilde{x}(t+2), \tilde{y}(t+2), \tilde{p}(t+2))$ von 5 Punkten betrachtet, d.h. der Punkt selbst und jeweils seine zwei zeitlichen Nachbarn.

Das Erscheinungsbild eines Liniensegments gibt dessen Verhältnis der horizontalen

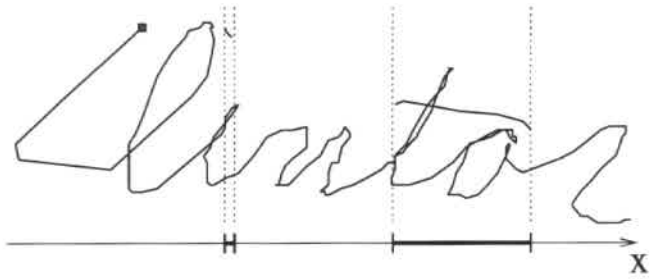


Abbildung 5.17: Das Hut-Merkmal

zur vertikalen Ausdehnung an und wird definiert als

$$A(t) = \frac{2 * \Delta y(t)}{\Delta x(t) + \Delta y(t)} - 1,$$

wobei $\Delta x(t)$ und $\Delta y(t)$ die Breite bzw. Höhe der umspannenden Box der Punkte in dem betrachteten Fenster ist. Aufgrund der Definition ist der Wertebereich dieses Merkmals auf das Interval $[-1, 1]$ beschränkt.

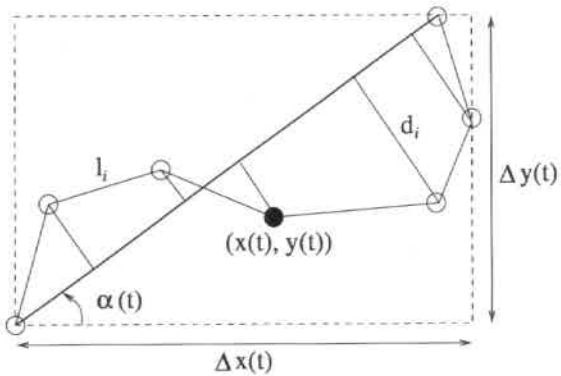


Abbildung 5.18: Definitionen für die Berechnung weiterer Merkmale

Gewelltheit (Curliness)

Dieses Merkmal ist ein Maß dafür, inwieweit sich das Liniensegment in seiner Länge von einer Geraden unterscheidet. Es ist definiert als das Verhältnis zwischen der Bogenlänge des Liniensegments und dem Maximum der Breite und Höhe der umge-

benden Box:

$$C(t) = \frac{(N-1) * l_i}{\max(\Delta x, \Delta y)} - 2,$$

wobei l_i die Länge des Segments und N die Anzahl der Punkte in dem Segment angibt. Der Wertebereich des Merkmals ist $[-1, N-3]$, wird aber selten größer als 1.

Linearität (Lineness)

Dieses Merkmal ist definiert als der gemittelte quadratische Abstand der Punkte des Liniensegments von der direkten Verbindungslinie D zwischen dem ersten und letzten Punkt:

$$L(t) = \frac{1}{N} * \sum_i d_i^2,$$

wobei d_i der Abstand von Punkt i von der Verbindungslinie D ist. Der Wertebereich liegt bei $[0, R]$, mit $R < (\Delta x^2 + \Delta y^2)$.

Neigung (Slope)

Das letzte lokale Merkmal ist der Cosinus der Neigung $\alpha(t)$ der direkten Verbindungslinie zwischen dem ersten und letzten Punkt des betrachteten Liniensegments.

5.4.2 Globale Merkmale

Die lokalen Merkmale repräsentieren die wesentlichen topologischen Gegebenheiten eines jeweils stark begrenzten zeitlichen Kontextes der Trajektorie. Sie erfassen aber keine globaleren Zusammenhänge, wie z.B. Nachbarschaftsbeziehungen zwischen zeitlich weiter auseinanderliegenden Segmenten der Trajektorie. Kreuzt ein Liniensegment ein bereits geschriebenes oder verläuft es in dessen Nähe, bleibt dies in den lokalen Merkmalen ebenso unberücksichtigt wie das Wissen, daß sich der aktuelle Punkt ggf. innerhalb eines Zeichens mit Unter- oder Oberlängen befindet. Bei der Off-line Erkennung auf Basis von Grauwertbildern tritt dieses Problem nicht auf, da bei den üblichen Ansätzen jeweils mehrere vollständige Spalten des Bildes betrachtet werden und damit genau diese Informationen eine große Rolle bei der Erkennung spielen. Um diesen Nachteil der lokalen Merkmale gegenüber Off-line Repräsentationen zumindest partiell auszugleichen, wird in dieser Arbeit der Merkmalsvektor um globale Merkmale ergänzt, die zwar weiterhin lokal im Raum, aber global in der Zeit sind [MFW94]. In keinem anderen heutigen System wurde bislang diese Kombination von lokalen und globalen Merkmalen untersucht und in die Erkennung einbezogen. Alle bekannten Systeme beschränken sich derzeit auf die Extraktion lokaler Merkmale, wie sie im vorigen Abschnitt vorgestellt wurden.

Ober- und Unterlängen

Die Existenz von Ober- und Unterlängen in einem bestimmten Bereich der Eingabe läßt Schlüsse darüber zu, welche Zeichen in diesem Bereich überhaupt wahrscheinlich sind. Sind in einem Bereich keine Ober- und Unterlängen vorhanden, ist auch die Existenz z.B. des Buchstabens „f" oder eines beliebigen Großbuchstabens in diesem Bereich eher unwahrscheinlich. Diese Information kann mittels der lokalen Merkmale jedoch nicht repräsentiert werden. Dies führte zum Hinzufügen zweier zusätzlicher Werte zu dem Merkmalsvektor, die eben diese Information kodieren. Für den jeweiligen Datenpunkt $(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))$ wird innerhalb eines bestimmten horizontalen Intervalls $D = [\tilde{x} - d, \tilde{x} + d]$ um den Datenpunkt festgestellt, wieviele andere Datenpunkte sich in der oberen bzw. unteren Schreibzone befinden (siehe Abbildung 5.19). Die Schreibzonen liegen durch die Bestimmung der Schreiblinien während der Normalisierung bereits vor. Die Summe aller Punkte in der oberen Schreibzone ergibt dann den ersten Wert dieses Merkmals, entsprechend die Summe der Punkte in der unteren Zone den zweiten Wert. Die Breite des Intervalls D entspricht in etwa einem Viertel der Kernhöhe des Wortes. Die jeweiligen Punkte in der oberen/unteren Schreibzone werden noch zusätzlich mit ihrem Abstand zu der Kernlinie/Basislinie gewichtet, d.h. je weiter ein Punkt vom Kern des Wortes entfernt ist, desto größer wird die Wahrscheinlichkeit, daß er nicht zum Kern des Wortes gehört. Dies verhindert, daß sich bereits kleine Überschreitungen der Schreiblinen, wie sie häufig zu beobachten sind, zu stark auf diese Merkmale auswirken. Zusätzlich muß ein Punkt auch einen Mindestabstand b von der jeweiligen Schreiblinie haben, um als zu Ober-/Unterlängen gehörig berücksichtigt zu werden. Damit wird verhindert, daß sich Ungenauigkeiten bei der Bestimmung der Schreiblinien auf diese Merkmale auswirken.

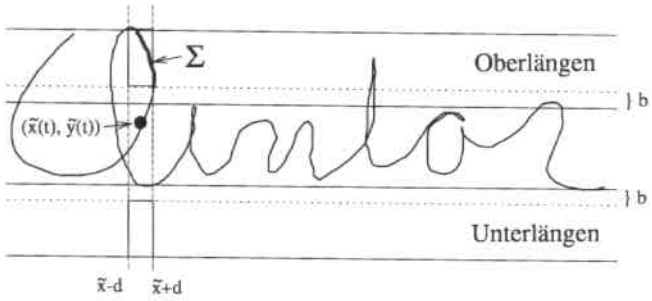


Abbildung 5.19: Berechnung von Ober- und Unterlängen

Kontext-Bitmaps

Bestimmte Ansätze der Off-line Handschrifterkennung, bei denen nacheinander jeweils nur ein beschränkter Ausschnitt aus dem Grauwertbild mit allen darin enthaltenen

Liniensegmenten betrachtet wird [LBD+89, KR91], liefern die Idee für weitere globale Merkmale, die sog. Kontext-Bitmaps [MFW94]. Sie erfassen für jeden Datenpunkt in Form eines Grauwertbildes mit sehr geringer Auflösung einen begrenzten räumlichen Kontext um den Punkt. Die normalisierte Eingabe $\{(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))\}_{t\in\{1..T'\}}$ wird hierzu in ein Grauwertbild $B = \{b(i,j)\}$ transformiert, wobei $b(i,j)$ die Anzahl der Punkte $(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))$ angibt, die in das Pixel (i,j) des Grauwertbildes B fallen (siehe Abbildung 5.20a). Für einen Datenpunkt $(\tilde{x}(t), \tilde{y}(t), \tilde{p}(t))$ ergibt sich die zugehörige Kontext-Bitmap derart, daß aus dem Grauwertbild ein $d \times d$ großer Ausschnitt mit dem zugehörigen Pixel als Mittelpunkt ausgeschnitten wird, wie in Abbildung 5.20 dargestellt ist. Die Größe dieses Ausschnitts entspricht mit $d = 9$ in etwa der Kernhöhe des Wortes. Um die Anzahl der Merkmale im Vergleich zu der Anzahl der lokalen Merkmale annähernd gleich groß zu halten, wird in einem weiteren Schritt dieser Ausschnitt in eine 3×3 Kontext-Bitmap verkleinert (Abbildung 5.20b) und dann in ausgerollter Form dem Merkmalsvektor hinzugefügt. Die

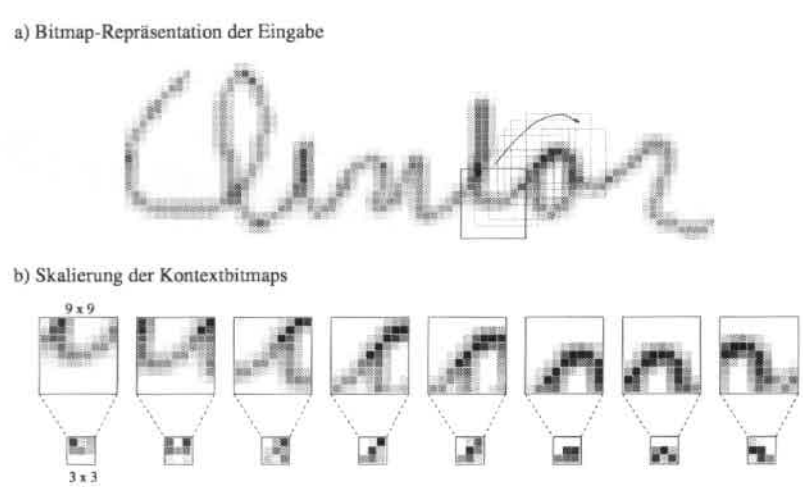


Abbildung 5.20: Berechnung der Kontextbitmaps

Breite und Höhe d des Ausschnittes, der bei der Berechnung der Kontext-Bitmaps betrachtet wird, wurde experimentell anhand der daraus resultierenden Erkennungsrate ermittelt. Ebenso wurde experimentell für die Worterkennung ermittelt, daß eine Vergrößerung der Kontext-Bitmaps von 3×3 auf 5×5 zu keiner Verbesserung der Erkennungsleistung führt, obwohl die Auflösung der Bitmaps dadurch erhöht wird und diese damit mehr Details enthalten.

5.4.3 Spezialfälle der Merkmalsextraktion

Ebenso wie bei der Normalisierung treten auch bei der Merkmalsextraktion ein paar Sonderfälle bei der Einzelzeichen- und Satzerkennung auf. Da Einzelzeichen anhand ihrer absoluten Höhe und nicht ihrer Kernhöhe in der Größe normalisiert werden und die x-Koordinaten nicht von der Position des Zeichens in einem Wort abhängen, wird das Merkmal, das die vertikale Position eines Punkten in Bezug auf die Basislinie angibt, durch den tatsächlichen absoluten x und y Wert ersetzt. Zusätzlich entfallen auch die beiden Merkmale, die die Unter- und Oberlängen in der Eingabe spezifizieren, da diese die Berechnung der Schreiblinien erforden, auf die aber wie bereits oben erwähnt bei Einzelzeichen verzichtet wird. Da bereits bei der Normalisierung die Überprüfung auf verzögerte Striche bei Einzelzeichen nicht durchgeführt wurde, entfällt auch das „Hut"-Merkmal.

Bei der Erkennung von Sätzen wird den oben beschriebenen Merkmalen lediglich ein weiteres hinzugefügt. Dieses zusätzliche Merkmal gibt einen Anhaltspunkt, an welchen Stellen in der Eingabe ein neues Wort beginnen könnte. Die Bestimmung dieses Merkmals wird ebenso wie die Detektion verzögerter Strich heuristisch durchgeführt. Hierzu wird davon ausgegangen, daß üblicherweise ein neues Wort nur nach einem vorausgehenden Abheben des Stiftes und ein wenig rechts der bisherigen Eingabe begonnen wird. Dieses Merkmal erhält also den Wert 0 an allen Punkten, an denen der Stift auf der Schreiboberfläche aufgesetzt war und daher ein Wortübergang unwahrscheinlich ist. An allen anderen Punkten wird der Wert dieses Merkmals aus dem Abstand zu dem letzten Punkt des Stiftabhebens berechnet, d.h. je weiter sich der Stift nach dem Abheben des Stiftes nach rechts bewegt hat, desto größer wird auch dieser Wert. Wichtig ist zu erwähnen, daß an dieser Stelle keine tatsächliche Entscheidung über Wortübergänge getroffen wird, sondern lediglich der Merkmalsvektor um eine zusätzliche Information ergänzt wurde, die ggf. die eigentliche Entscheidung der nachfolgenden Erkennung unterstützt.

5.4.4 Relevanz der Merkmale

In diesem Abschnitt soll nun geklärt werden, welchen Beitrag die lokalen und globalen Merkmale zu der Gesamterkennungsleistung beitragen. Da aufgrund der Vielzahl nicht jede Kombination der unterschiedlichen Merkmale untersucht werden kann, beschränkt sich diese Analyse darauf, den Beitrag der lokalen und globalen Merkmale als Ganzes festzustellen. Die Erfahrungen während der Entwicklung dieses Systems haben gezeigt, daß bereits mit einigen wenigen der in diesem Kapitel vorgestellten Merkmale sehr gute Erkennungsraten erzielt werden können. Nimmt man lediglich die Merkmale für die Schreibrichtung, die Krümmung, die vertikale Position und den Stiftzustand, liegt die Gesamterkennungsrate für Einzelworte bereits bei 90%. Die übrigen Merkmale reduzieren die Fehlerrate dann nochmals um 30%.

Tabelle 5.7 zeigt die Erkennungsraten auf allen Testmengen der Einzelzeichen- und

	UKA	CMU	MIT	Gesamt
0...9:				
lokal	97.9%	97.2%	95.1%	**96.4%**
global	97.4%	94.6%	94.3%	**95.0%**
global+lokal	97.9%	97.0%	95.3%	**96.5%**
A...Z:				
lokal	91.7%	90.8%	94.0%	**92.2%**
global	89.7%	88.4%	93.5%	**90.6%**
global+lokal	93.4%	90.5%	94.9%	**92.7%**
a...z:				
lokal	89.1%	89.0%	92.0%	**90.2%**
global	88.0%	87.3%	90.4%	**88.6%**
global+lokal	89.7%	89.8%	93.4%	**91.1%**
Wörter:	PRINTED/CURSIVE	PRINTED/CURSIVE	PRINTED	
lokal	98.7%/90.1%	94.8%/89.9%	92.9%	**92.2%**
global	96.2%/87.2%	92.2%/88.1%	92.6%	**90.7%**
global+lokal	97.1%/90.8%	92.8%/92.2%	94.7%	**93.4%**

Tabelle 5.7: Erkennungsraten mit unterschiedlichen Merkmalen

Worterkennung, wenn jeweils nur alle lokalen Merkmale, alle globalen Merkmale oder alle zusammen verwendet werden. Wie man sieht, beinhalten die lokalen Merkmale in allen Fällen offensichtlich etwas mehr Informationen als die globalen Merkmale, da die Erkennungsrate jeweils um etwa 2% höher liegt. Aber auch die globalen Merkmale alleine, die nur eine sehr grobe Repräsentation der Eingabe darstellen, kommen bereits sehr nah an die beste Erkennungsleistung heran, die sich erst durch Kombination aller Merkmale ergibt. Je schwieriger die Erkennungsaufgabe ist, desto mehr tragen die globalen Merkmale zu einer Leistungsverbesserung bei.

5.5 Beiträge dieser Arbeit

Die Ergebnisse dieses Kapitels belegen, daß durch einen sorgfältigen Entwurf der Vorverarbeitung die Robustheit eines Handschrifterkenners gegenüber Schreiber- und Hardware-bedingter Variabilität der Eingabe deutlich erhöht werden kann. In diesem System werden erstmals alle wesentlichen Effekte, die in dem Eingabesignal auftreten, bei der Normalisierung berücksichtigt. Grundlage für viele der Normalisierungsschritte ist die Bestimmung der Schreiblinien, die mit dem vorgestellten Ansatz gleichzeitig und in Abhängigkeit voneinander berechnet werden können. Eine vorherige Korrektur der Rotation der Eingabe, wie im Ansatz von [LB94], wird hier durch entsprechende Initialisierung der Modellparameter vermieden. Es wurde gezeigt, daß eine nichtlineare Interpolation großer Abtastlücken, die z.B. bei drucksensitiven Bildschirmen

häufig auftreten, zu einer Reduktion der Fehlerrate beiträgt. Solche Abtastlücken werden in anderen Systemen bislang nicht berücksichtigt, da sie häufig erst im praktischen Einsatz eine Rolle spielen und in den existierenden Datenbasen seltener auftreten.

Lokale Merkmale, wie die Schreibrichtung oder Krümmung für einen Punkt der Trajektorie, sind in vielen heutigen Systemen zu finden. Die Ergebnisse dieses Kapitels zeigen, daß bereits mit wenigen lokalen Merkmalen Erkennungsraten von über 90% auf Einzelzeichen und Wörtern erzielt werden können. Bereits in [Sch95] wurde gezeigt, daß durch weitere oder modifizierte lokale Merkmale die Erkennungsleistung nur noch marginal verbessert werden kann. Erst durch die in dieser Arbeit entwickelte Kombination von lokalen und globalen Merkmalen ergibt sich eine deutliche Fehlerreduktion auf allen Erkennungsaufgaben. Je schwieriger die Erkennungsaufgabe ist, desto größer ist auch die beobachtete Verbesserung der Erkennungsleistung durch die globalen Merkmale. Mit den globalen Merkmalen, die unabhängig von der zeitlichen Entstehung der Trajektorie sind, wird die Distanz zwischen der Off-line Erkennung mit ihrer bildlichen Repräsentation der Eingabe und der On-line Erkennung mit der rein zeitlichen Information ein Stück weit überbrückt. Die in den globalen Merkmalen enthaltene Information über räumliche Nachbarschaften einzelner Trajektorienbereiche kann durch die zeitlich lokalen Merkmale nicht erfaßt werden.

Die aus der Vorverarbeitung resultierende Repräsentation der handschriftlichen Eingabe orientiert sich eng an der ursprünglichen, punktbasierten Repräsentation. Die zeitliche Information über die Entstehung der Trajektorie bleibt weiterhin erhalten. Bis zu diesem Zeitpunkt wurden keine Segmentierungs- oder anderweitige Entscheidungen getroffen, die in nachfolgenden Erkennungsschritten Fehlerkennungen verursachen.

Kapitel 6

Das MS-TDNN zur Zeichenerkennung

6.1 Einleitung

Die im vorhergehenden Kapitel vorgestellte Repräsentation der handschriftlichen Eingabe als Folge von Merkmalsvektoren erfordert einen Erkennungsansatz, der ein über die Zeit veränderliches Signal mit variabler Länge verarbeiten kann. Mit dem Time Delay Neural Network (TDNN) und seiner Erweiterung, dem Multi-State Time Delay Neural Network (MS-TDNN), ist eine solche Erkennerarchitektur aus der Spracherkennung bekannt. Dort wurden für diese konnektionistischen Architekturen ihre diskriminativen Fähigkeiten bei der Erkennung einzelner Phoneme und auch kontinuierlicher Sprache bereits unter Beweis gestellt [WHH+89, HFW91, Hil97].

Auch in einigen der in Kapitel 3 vorgestellten Systeme basiert die Erkennung auf TDNNs [GAL+91, Sch95, Sen95]. Bei diesen Ansätzen wird jedoch jedes Zeichen als Ganzes betrachtet. Eine weitere Unterteilung der Zeichen in kleinere Einheiten, die die Erkennung erleichtern, wird dort nicht vorgenommen. Lediglich in HMM-basierten Systemen erfolgt die Modellierung einzelner Zeichen durch eine Folge von Zuständen. In dieser Arbeit wird nun gezeigt, wie die bereits aus der Spracherkennung bekannte Modellierung gesprochener Buchstaben durch eine Folge von Phonemen auch für die On-line Handschrifterkennung adaptiert und in die TDNN Architektur integriert werden kann, so daß sich ein hybrides System aus neuronalem Netz und HMM ergibt. Durch diese Modellierung eines Zeichens als Folge von Zuständen kann die Fehlerrate bei der Einzelworterkennung relativ um etwa 50% reduziert werden, wie die Ergebnisse dieser Arbeit zeigen. Diese Architektur ist damit allen bisher vorgestellten Ansätzen, die nur auf neuronalen Netzen oder nur auf HMMs beruhen, deutlich überlegen.

Dieses Kapitel stellt zunächst die TDNN Architektur für die Einzelzeichenerkennung vor, wie sie üblicherweise verwendet wird. Diese Architektur wird dann in einem nächsten Schritt zu einem MS-TDNN erweitert, so daß Zeichen nun auch als Folge von Zuständen modelliert und erkannt werden können.

6.2 Das Time Delay Neural Network (TDNN)

Time Delay Neural Networks (TDNN) sind eine Klasse vorwärtsgerichteter Netze, die ursprünglich für Spracherkennungsaufgaben entwickelt wurden. Das TDNN wurde erstmalig 1987 für die Klassifikation der drei Phoneme /b/, /d/ und /g/ in einem gegebenen Stück Sprache eingesetzt [WHH+87, WHH+89, LWH90]. Ziel der Entwicklung des TDNN war es, Netze zu konstruieren, die charakteristische Merkmale in zeitlich veränderlichen Mustern an unterschiedlichen Positionen mit unterschiedlicher Länge erkennen können. Übertragen auf die Handschrifterkennung bedeutet dies: Das TDNN ist in der Lage, ein Zeichen bzw. Teile eines Zeichens unabhängig von dessen Position innerhalb einer Eingabesequenz und unabhängig davon, durch wieviele Merkmalsvektoren es in dieser Eingabesequenz repräsentiert wird, zu erkennen. Die zwei wesentlichen Merkmale eines TDNN sind daher zum einen seine Fähigkeit, zeitliche Zusammenhänge aufeinanderfolgender Merkmalsvektoren zu erlernen, und zum anderen seine Verschiebungsinvarianz. Die Fähigkeit, zeitliche Zusammenhänge zu erfassen, wird dadurch erreicht, daß sowohl in der Eingabeschicht als auch der verborgenen Schicht jeweils ein zeitlicher Kontext (Time-Delay) in die Zukunft und Vergangenheit, also mehrere Merkmalsvektoren gleichzeitig, betrachtet werden. Die Verschiebungsinvarianz ergibt sich aus der Verwendung gekoppelter Gewichte in der Zeitachse. Ein Eingabefenster wird über die einzelnen Schichten des TDNN hinweggeschoben und für jeden Zeitpunkt, d.h. jede Position des Fensters, mit denselben Gewichten eine Ausgabe in der Ausgabeschicht erzeugt. Das schrittweise Verschieben dieses Eingabefensters ermöglicht insbesondere die Verarbeitung unterschiedlich langer Eingaben, wie sie sowohl in der Sprach- als auch der Handschrifterkennung vorliegen.

6.2.1 Architektur des TDNN

Die Architektur des TDNN soll nun ausgehend von dem Spezialfall, daß die Breite der Eingabefenster als 1 gewählt ist, d.h. kein zeitlicher Kontext betrachtet wird, erläutert werden. Wird der zeitliche Kontext weggelassen, führt dies zu einem einfachen mehrschichtigen Perzeptron (siehe Kapitel 2.5.2), welches jeweils mit identischen Gewichten für einen einzigen Merkmalsvektor x_t der Eingabe einen Ausgabevektor y_t berechnet (Abbildung 6.1a). Wird die Klassifikation mit diesem MLP für alle Vektoren der Merkmalssequenz $x_1^T = x_1 \dots x_T$ durchgeführt, ergibt sich die Darstellung in Abbildung 6.1b, diesmal mit der für TDNNs üblichen vertikalen Anordnung der Neuronen. Für jeden Merkmalsvektor der Eingabe ergibt sich also genau ein Ausgabevektor, der die Wahrscheinlichkeit für jede Ausgabeklasse angibt, daß dieser Merkmalsvektor zu dieser Klasse gehört. Das Gesamtergebnis einer Ausgabeklasse y_i bildet abschließend die Summe über alle Einzelklassifikationen $y_{t,i}$:

$$y_i = \sum_{t=0}^{T} y_{t,i}$$

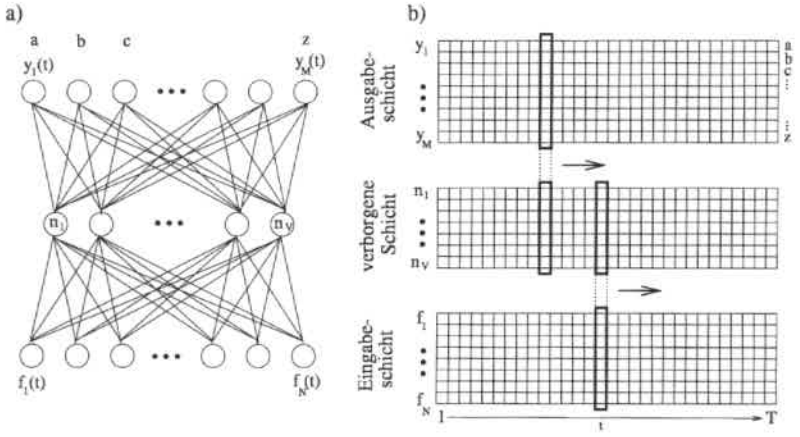


Abbildung 6.1: Spezialfall des TDNN ohne Zeitverzögerungen (Time-Delays)

Die Überlegung, daß bei dieser Art der Erkennung ohne Kontext das MLP anhand eines einzigen Merkmalsvektors entscheiden muß, zu welcher Ausgabeklasse dieser gehört, läßt erkennen, daß dieser einfache Spezialfall der TDNN Architektur nicht zum Ziel führt. Läßt man außer Betracht, daß die Merkmale an sich bereits einen gewissen zeitlichen Kontext beinhalten, hieße dieses Vorgehen auf die Originaltrajektorie der Eingabe bezogen, aus einem einzigen Datenpunkt auf das zugehörige Zeichen zu schließen. Allerdings wird bereits bei diesem Spezialfall die Verschiebungsinvarianz des TDNN deutlich, da durch die sukzessive Abarbeitung der Eingabe ein bestimmter Merkmalsvektor unabhängig von seiner Position klassifiziert wird.

Die eigentliche Architektur des TDNN ergibt sich dadurch, daß sowohl in der Eingabeschicht als auch der verborgenen Schicht anstelle eines einzigen Vektors jeweils ein begrenzter zeitlicher Kontext betrachtet wird. Abbildung 6.2 zeigt die Standarddarstellung des TDNN mit vier Schichten. Jedes Neuron der verborgenen Schicht hat hier nun nicht mehr nur eine vollständige Verbindung zu einem einzigen Vektor der Eingabeschicht, sondern zu einem Fenster bestimmter Breite von mehreren Vektoren. Gleiches gilt für die, in der Spracherkennung „Phonemschicht" und im folgenden im Vorgriff auf das nächste Kapitel „Zustandsschicht" genannte, zweite verborgene Schicht. Abhängig von der Breite der Fenster in der Eingabe- und der verborgenen Schicht bewerten die Neuronen der Zustandsschicht daher einen wesentlich größeren Bereich der Eingabe gleichzeitig und klassifizieren ihn in eine der möglichen Ausgabeklassen. Die Ausgabe des TDNN ergibt sich wiederum durch Summation der einzelnen Zeilen der Zustandsschicht in jeweils ein einziges Ausgabeneuron.[1]

Die Arbeitsweise eines TDNN kann in Anlehnung an [Hil97] wie im folgenden

[1] Auf das Ausgabeneuron kann wie in [WHH+89] zusätzlich noch die Sigmoidfunktion angewendet werden.

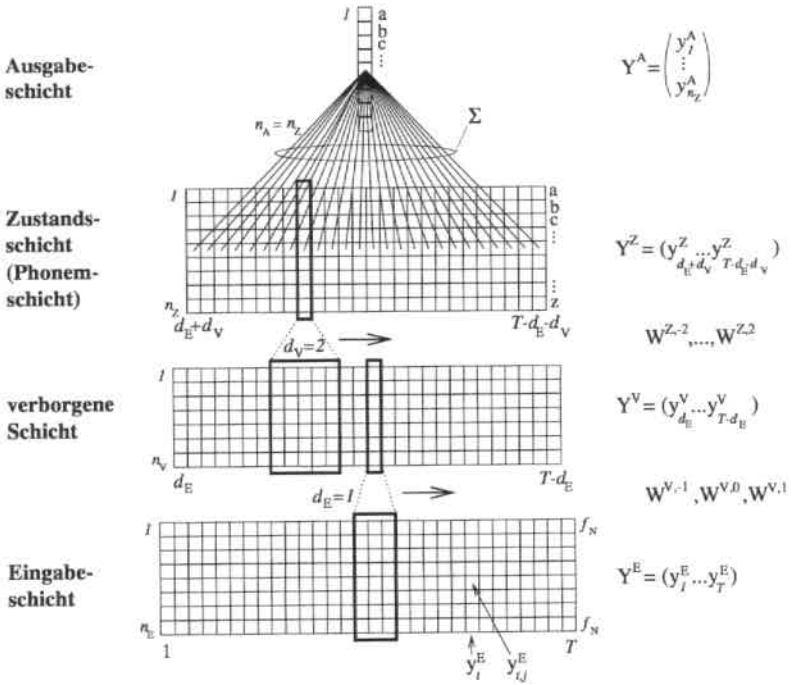


Abbildung 6.2: Die Architektur des TDNN

formalisiert werden. Die Eingabeschicht Y^E nimmt die zu klassifizierende Folge $x_1^T = x_1 \ldots x_T$ der Merkmalsvektoren $x_t = f_1(t) \ldots f_N(t)$ mit der Dimension N $(= n_E)$ direkt auf, d.h. in der Eingabeschicht findet keine Verarbeitung statt:

$$Y^E = (y_1^E \ldots y_T^E) = \left(\begin{pmatrix} y^E_{1,1} \\ \vdots \\ y^E_{1,N} \end{pmatrix} \cdots \begin{pmatrix} y^E_{T,1} \\ \vdots \\ y^E_{T,N} \end{pmatrix} \right)$$

mit $y^E_{i,j} = f_j(i)$ für $i = 1 \ldots T$ und $j = 1 \ldots N$. Ein Vektor y_t^V der verborgenen Schicht ist über Gewichte W

$$W = (w_{i,j}) = \begin{pmatrix} w_{1,1} & \cdots & w_{1,n_E} \\ \vdots & & \vdots \\ w_{n_V,1} & \cdots & w_{n_V,n_E} \end{pmatrix}$$

vollständig verbunden mit dem Vektor y_t^E, sowie über analoge Gewichtsmatrizen $W^{S,d}$ mit den $2 \cdot d_E$ Vektoren $y^E_{t-d_E}, \ldots, y^E_{t-1}, y^E_{t+1}, \ldots, y^E_{t+d_E}$ zur Linken und zur Rechten von y_t^E. n_V gibt hierbei die Anzahl der Neuronen der verborgenen Schicht an und d

den Kontext. Die Indizes S für die Zielschicht und d für die Zeitverzögerung von $W^{S,d}$ spezifizieren, zu welcher Verbindung eine Gewichtsmatrix gehört. Die Ausgabe eines Neurons der verborgenen Schicht ergibt sich aus

$$y_t^V = f(\sum_{d=-d_E}^{d_E} W^{V,d} y_{t+d}^E)$$

mit der Transferfunktion f. Analog dazu werden die Ausgabeaktivierungen der Zustandsschicht Z berechnet aus

$$y_t^Z = f(\sum_{d=-d_V}^{d_V} W^{Z,d} y_{t+d}^V).$$

In einem letzten Schritt werden die Aktivierungen der Zustandsschicht über die Zeit akkumuliert und in die Ausgabeschicht propagiert sowie zusätzlich die Sigmoidfunktion angewendet:

$$y_i^A = f(\sum_{t=d_E+d_V}^{T-d_E-d_V} y_{t,i}^Z).$$

Die Breite der verwendeten Fenster in der Eingabe- und der verborgenen Schicht sowie die Anzahl der Neuronen der verborgenen Schicht sind empirisch bestimmt und hängen von der jeweiligen Erkennungsaufgabe ab. Das ursprüngliche, in [WHH+89] für die Phonemerkennung vorgeschlagene TDNN hatte Fenster der Breite 3 in der Eingabeschicht bzw. 5 in der verborgenen Schicht und 15 Neuronen in der verborgenen Schicht. Diese Einstellungen wurde auch in neueren Arbeiten wie [Hil97] übernommen. Die in dieser Arbeit verwendeten TDNNs zur Handschrifterkennung haben einen breiteren Kontext mit einer Fensterbreite von 7 sowohl in der Eingabe- als auch der verborgenen Schicht. Als Anzahl der Neuronen in der verborgenen Schicht wurde 30 für die Ziffernerkennung und 80 für die Erkennung von Klein- und Großbuchstaben gewählt. D.h. die Anzahl der Neuronen wurde entsprechend der Schwierigkeit des Erkennungsproblems und der Anzahl der Trainingsdaten angepaßt. Die Zahl der Neuronen in der Eingabe-, Zustands- und Ausgabeschicht des TDNN sind durch die Dimension der Merkmalsvektoren bzw. der zu unterscheidenden Zeichenklassen gegeben. In Abweichung von der Standardarchitektur des TDNN wird in den hier verwendeten Netzen das Fenster in der Eingabeschicht mit jedem Zeitschritt um 2 Merkmalsvektoren verschoben, so daß die verborgene Schicht bereits weniger als die Hälfte der Länge der Eingabeschicht hat. Dadurch wird zum einen der Gesamtkontext für einen Ausgabevektor des TDNN auf 19 Merkmalsvektoren vergrößert und zum anderen die Berechnung der Propagierung durch das Netzwerk zeitlich erheblich verkürzt, was sich insbesondere bei längeren Eingabesequenzen wie Wörtern und Sätzen bemerkbar macht. Die Ergebnisse in Kapitel 7.2 für die Worterkennung zeigen, daß dies eine bessere Lösung ist, als bereits in der Vorverarbeitung die Abtastrate zu halbieren, was dieselbe Verkürzung der Berechnungszeit zur Folge hätte. Abbildung 6.3 zeigt eine schematische Darstellung der Architekturparameter für die

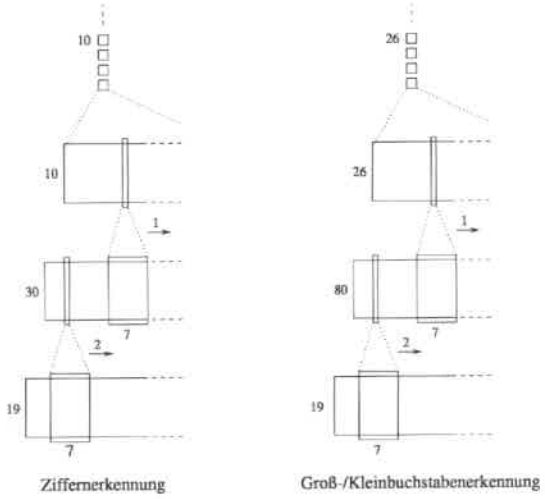


Abbildung 6.3: TDNN Architekturen für die Einzelzeichenerkennung

Erkennung von Ziffern und Groß- bzw. Kleinbuchstaben. Insgesamt gibt es bei diesen gewählten Netzgrößen für die Ziffernerkennung 6 090 und für die Erkennung von Klein- bzw. Großbuchstaben 25 200 lernbare Gewichte.

6.2.2 Backpropagation für das TDNN

Das Lernziel für das TDNN ist die korrekte Klassifikation der handschriftlichen Eingabe als eines der Zeichen $c^i \in C$. Die Trainingsmenge besteht aus P Mustern $X^p, p = 1 \ldots P$, denen jeweils eine Zeichenklasse $c^p \in C$ zugeordnet ist. Die Sollausgaben $z^p = (z_i^p)$ des TDNN sind 1-aus-N Kodierungen der korrekten Ausgabeklasse c^p, d.h. 1 für den der Klasse c^p zugeordneten Ausgabeknoten und 0 für alle übrigen Knoten. Gesucht ist nun ein Gewichtssatz W, der z.B. den mittleren quadratischen Fehler

$$E_{\text{MSE}} = \frac{1}{2} \sum_{p=1}^{P} \sum_{i=1}^{n_Z} (y_i^{A,p} - z_i^p)^2$$

auf den Ausgaben $y^{A,p}$ des TDNN für alle Trainingsmuster X^p minimiert. Die Bestimmung von W erfolgt analog dem für das MLP vorgestellten Backpropagation-Verfahren durch Gradientenabstieg. Bei der Berechnung der Ableitung des Fehlers $\partial E / \partial w_{ij}$ müssen jedoch hier zusätzlich die Kontexte in der Eingabe- und der verborgenen Schicht sowie die zeitinvariante Verschiebung der Gewichte über die Zeit berücksichtigt werden. Zunächst wird die Abhängigkeit der Zustandsbewertungen $y_{t,i}^Z$ betrachtet. Die Ableitung des Fehlers nach $y_{t,i}^Z$ berechnet sich nach der Kettenregel

wie folgt:

$$\frac{\partial E}{\partial y^Z_{t,i}} = \frac{\partial E}{\partial y^A} \cdot \frac{\partial y^A}{\partial y^Z_{t,i}} = \sum_{k=1}^{n_Z} \frac{\partial E}{\partial y^A_k} \cdot \frac{\partial y^A_k}{\partial y^Z_{t,i}} = \frac{\partial E}{\partial y^A_i} \cdot \frac{\partial y^A_i}{\partial y^Z_{t,i}} = \frac{1}{T} \frac{\partial E}{\partial y^A_i},$$

wobei T die Anzahl der Ausgabevektoren der Zustandsschicht ist. Die Summe über k entfällt, da die i-te Zustandsbewertung $y^Z_{t,i}$ nur die i-te Ausgabe y^A_i beeinflußt. Die gesuchte Ableitung nach den Gewichten $w^{Z,d}_{i,j}$ ergibt sich dann als

$$\frac{\partial E}{w^{Z,d}_{i,j}} = \frac{\partial E}{\partial \mathrm{Y}^Z} \cdot \frac{\partial \mathrm{Y}^Z}{w^{Z,d}_{i,j}} = \sum_t \sum_{k=1}^{n_Z} \frac{\partial E}{\partial y^Z_{t,k}} \cdot \frac{\partial y^Z_{t,k}}{\partial w^{Z,d}_{i,j}}$$

Nach einigen weiteren Umformungen (siehe dazu z.B. [Hil97], Seite 187 ff.) erhält man die gesuchten Ableitungen

$$\begin{aligned}
\frac{\partial E}{\partial y^Z_{t,i}} &= \frac{1}{T} \frac{\partial E}{\partial y^A_i} \\
\frac{\partial E}{\partial w^{Z,d}_{i,j}} &= \sum_t \frac{\partial E}{\partial y^Z_{t,i}} f'(x^Z_{t,i}) \cdot y^V_{t+d,j} \\
\frac{\partial E}{\partial y^V_{t,i}} &= \sum_{d'=-d_V}^{d_V} \sum_{k=1}^{d_Z} \frac{\partial E}{\partial y^Z_{t-d',k}} \cdot f'(x_{t-d',k}) \cdot w^{Z,d'}_{k,i} \\
\frac{\partial E}{\partial w^{V,d}_{i,j}} &= \sum_t \frac{\partial E}{\partial y^V_{t,i}} f'(x^V_{t,i}) \cdot y^E_{t+d,j}
\end{aligned}$$

wobei f' die Ableitung der Transferfunktion ist. Es wird zunächst die Ableitung $\partial E/\partial y^A$ berechnet, die lediglich von der gewählten Fehlerfunktion abhängt. $\partial E/\partial x$ erhält man dadurch, daß man nun mit $\partial E/\partial y$ rückwärts durch die Transferfunktion f geht. Entlang der Gewichte der Zeitverzögerungen wird dieses Fehlersignal anschließend in die verborgene Schicht zurückpropagiert. Entsprechend geht man bei der Rückpropagierung in die Eingabeschicht vor.

6.2.3 Einzelzeichenerkennung mit dem TDNN

Mit der in den vorigen Abschnitten beschriebenen TDNN Architektur können bereits einfache leistungsfähige Einzelzeichenerkenner für Ziffern, Klein- und Großbuchstaben realisiert werden. Für jede der drei Erkennungsaufgaben wird auf den entsprechenden Trainingsdaten aller drei Datenbasen ein TDNN trainiert und anschließend auf den Testmengen evaluiert. Tabelle 6.1 listet die dabei auf den drei Testmengen erzielte sowie die Gesamterkennungsrate der einzelnen Erkenner auf. Ein Vergleich dieser Ergebnisse mit den bereits in Kapitel 4 präsentierten besten Ergebnissen zeigt, daß durch Erweiterungen des TDNN noch Verbesserungen der Erkennungsleistung zu erzielen sind. Diese Erweiterungen führen zu der im folgenden Abschnitt beschriebenen Multi-State Time Delay Neural Network Architektur.

	UKA	CMU	MIT	Gesamt
0...9:	94.2%	94.9%	94.5%	**94.6%**
A...Z:	90.8%	87.8%	94.0%	**90.8%**
a...z:	85.2%	87.1%	89.5%	**87.7%**

Tabelle 6.1: Erkennungsraten bei Einzelzeichen mit dem TDNN

6.3 Das Multi-State Time Delay Neural Network (MS-TDNN)

Bei der Beschreibung der TDNN Architektur in den vorhergehenden Abschnitten wurde ein handschriftliches Zeichen der Handschrifterkennung mit einem gesprochenen Phonem der Spracherkennung gleichgesetzt. Phoneme bilden jedoch nur einen bestimmten akustischen Abschnitt eines gesprochenen Buchstabens, d.h. der Buchstabe wird durch eine definierte Sequenz von Phonemen gebildet. Eine mögliche akustische Modellierung des Buchstabens „B" durch drei Phoneme wäre z.B. „/b/ /b-eh/ /eh/". Um einen gesprochenen Buchstaben zu erkennen, müssen demzufolge die zugehörigen Phoneme oder auch Zustände (States) in der definierten Reihenfolge erkannt werden. Die eigentliche Klassifikationsaufgabe wird so in kleinere, für ein neuronales Netz leichter erlernbare Klassifikationsaufgaben aufgeteilt.

Aufgrund dieser Erfahrungen der Spracherkennung bietet sich auch für die Erkennung eines geschriebenen Zeichens eine analoge Aufteilung in kleinere Zeicheneinheiten an. Die Trajektorie eines Zeichens weist im zeitlichen Verlauf in der Regel viele Änderungen z.B. der Schreibrichtung, Krümmung oder Orientierung auf. Es ergeben sich also unterschiedliche Trajektorienbereiche, die in der obigen TDNN Architektur zur Einzelzeichenerkennung von einem einzigen Neuron der Zustandsschicht einer Zeichenklasse zugeordnet werden muß. Die Lernaufgabe wird nun im folgenden dadurch erleichtert, daß die Klassifikation eines Zeichens auf mehrere Neuronen verteilt, d.h. in kleinere Teilprobleme zerlegt wird. Da aufgrund der unterschiedlichen Schreibstile eine bedeutungstragende und intuitive Unterteilung der Zeichen in Teilbereiche wie in der Spracherkennung nur in den wenigsten Fällen möglich ist, wird hier die bereits in Kapitel 2.6.5 erläuterte abstrakte Modellierung eingeführt [MB94, MFW95b].

6.3.1 Modellierung von Einzelzeichen

$C = \{c_1, c_2, \ldots, c_N\}$ sei die Menge der zu unterscheidenden Zeichenklassen, d.h. der Wortschatz des Einzelzeichenerkenners, mit $N = 10$ für die Ziffernerkennung und $N = 26$ für die Erkennung von Klein- bzw. Großbuchstaben. Jedes Zeichen c_i des Wortschatzes wird nun modelliert durch ein HMM mit M Zuständen:

$$c_i \equiv s_{i,1} s_{i,2} \ldots s_{i,M}$$

Die Anzahl M der Zustände ist aufgrund der abstrakten Modellierung im Gegensatz zur Spracherkennung für alle Zeichen identisch. Die Gesamtmenge aller Zustände $S = \{s_{i,j}\}_{i=1\cdots N, j=1\cdots M}$ hat daher die Größe $N \cdot M$. Die Modellierung erfolgt durch einfache Links-Rechts-Modelle, wie sie für die Buchstaben „a", „A" und „0" mit $M = 3$ in Abbildung 6.4 dargestellt sind. Ein Durchlauf eines Modells beginnt stets im ersten

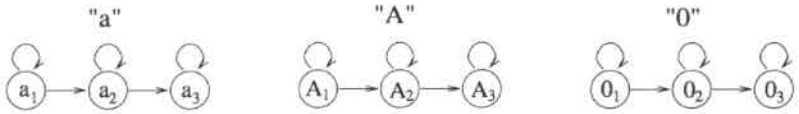


Abbildung 6.4: Buchstabenmodellierung

und endet im letzten Zustand des Modells. Ein Übergang von einem Zustand ist nur in den unmittelbar folgenden möglich. Es können also keine Zustände übersprungen werden.

Eine Modellierung der Zeichen durch genau 3 Zustände ergibt für die Einzelworterkennung das beste Erkennungsergebnis, wie die Experimente in Kapitel 7.2 zeigen. Diese Modellierung wird aus Gründen der Einheitlichkeit auch für die Einzelzeichenerkennung übernommen, obwohl sich hier vereinzelt eine Modellierung durch mehr als 3 Zustände als geringfügig besser erweist, wie durch die Experimente in Kapitel 6.5.3 ermittelt wird.

6.3.2 Die Architektur des MS-TDNN

Für die Integration der Zeichenmodelle mit mehreren Zuständen wird nun das sog. Multi-State Time Delay Neural Network (MS-TDNN) verwendet, ursprünglich eine Weiterentwicklung des TDNN für die Erkennung von Phonemfolgen in der Spracherkennung [HFW91, HW92]. Zu jedem der $N \cdot M$ Zustände von S gibt es in der Zustandsschicht des MS-TDNN eine Zeile von Neuronen, wobei $y^Z_{t,i}$ die Bewertung von Zustand s_i zum Zeitpunkt t berechnet. Die Zustandsschicht des ursprünglichen TDNN wurde also von N auf $N \cdot M$ Neuronen erweitert.

Die Berechnung der Aktivierung eines Ausgabeneurons y^A_i des MS-TDNN kann nun nicht mehr durch einfache Integration der Aktivierungen des zugehörigen Neurons der Zustandsschicht erfolgen, da sie durch die Aktivierungen der zu dem entsprechenden Modell gehörenden M Neuronen gebildet wird. Betrachtet man die Aktivierungen der zu einem Modell c_i gehörenden Neuronen über die Zeit als eine Matrix, wie in Abbildung 6.5 für das Modell des Buchstabens „a" dargestellt, so entspricht ein Durchlauf des Modells c_i einem Pfad durch diese Matrix, der mit dem Viterbi-Algorithmus bestimmt werden kann. Aufgrund der Verwendung von Links-Rechts-Modellen muß ein solcher Pfad in der Matrix stets links oben, d.h. im ersten Zustand des Modells, beginnen und entsprechend rechts unten, d.h. dem letzten Zustand, enden. Eine Schleife in dem Modell entspricht hierbei einem Verbleiben in derselben Zeile der Matrix, ein Übergang zum nächsten Zustand dem Sprung in die nächste

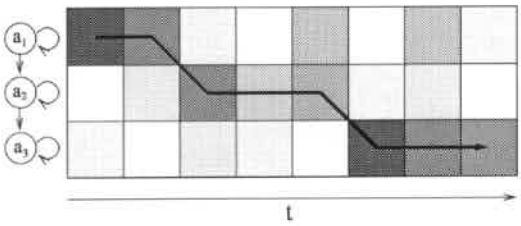


Abbildung 6.5: Bester Pfad für Buchstaben „a"

Zeile. Die Summe der Aktivierungen entlang dieses besten Pfades durch die Matrix bildet die gesuchte Aktivierung für das zugehörige Ausgabeneuron des MS-TDNN. Jedes zu erkennende Zeichen c_i besitzt sein eigenes Modell und damit eine eigene Bewertungsmatrix, deren Zeilen aus den zugehörigen Zuständen gebildet werden. Für $M = 3$ und $N = 10(= n_A)$ bei der Ziffernerkennung und $N = 26$ bei der Erkennung von Klein- und Großbuchstaben existieren insgesamt 10 Ziffernmodelle bzw. 26 Buchstabenmodelle und $n_Z = 30$ bzw. $n_Z = 78$ Zustandsneuronen im MS-TDNN.

Im TDNN wurde in der Ausgabeschicht für jedes zu klassifizierende Zeichen c_i eine Entscheidungsfunktion $g_i(x) = y_i^A$ berechnet. Die Architektur des MS-TDNN ergibt sich aus der TDNN Architektur durch Einfügen einer zusätzlichen Schicht Y^M mit einem Knoten y_i^W für jedes zu erkennende Zeichen $c_i \in C$:

$$Y^M = \begin{pmatrix} y_1^M \\ \vdots \\ y_{n_M}^M \end{pmatrix},$$

In Abbildung 6.6 ist die sich daraus ergebende Architektur schematisch für die Erkennung von Kleinbuchstaben dargestellt. Die Berechnung der Entscheidungsfunktion $g_i(x) = y_i^M$ erfolgt mit Hilfe der oben eingeführten Zeichenmodelle durch einen hybriden Ansatz aus dem MS-TDNN und den HMMs der zu erkennenden Zeichen. Die Zustandsbewertungen $y_{t,i}^Z$ werden als a posteriori Wahrscheinlichkeiten

$$y_{t,i}^Z = P(q_t = s_i|\bar{x}_t)$$

interpretiert. Aufgrund der Berücksichtigung der zeitlichen Kontexte in der Eingabe- und der verborgene Schicht des TDNN entspricht $\bar{x}$ hier einem Eingabefenster $(x_{t-k} \ldots x_{t+k})$ aus $2k + 1$ Eingabevektoren. Mit der Bayes-Regel werden die klassenbedingten Wahrscheinlichkeiten berechnet als

$$P(\bar{x}_t|s_i) = \frac{P(s_i|\bar{x}_t)P(\bar{x}_t)}{P(s_i)}$$

$P(\bar{x}_t|s_i)$ entspricht der Emissionswahrscheinlichkeit $b_i(x)$ eines HMM. Für eine gegebene Eingabe X kann der Pfad π^* mit der höchsten Beobachtungswahrscheinlichkeit

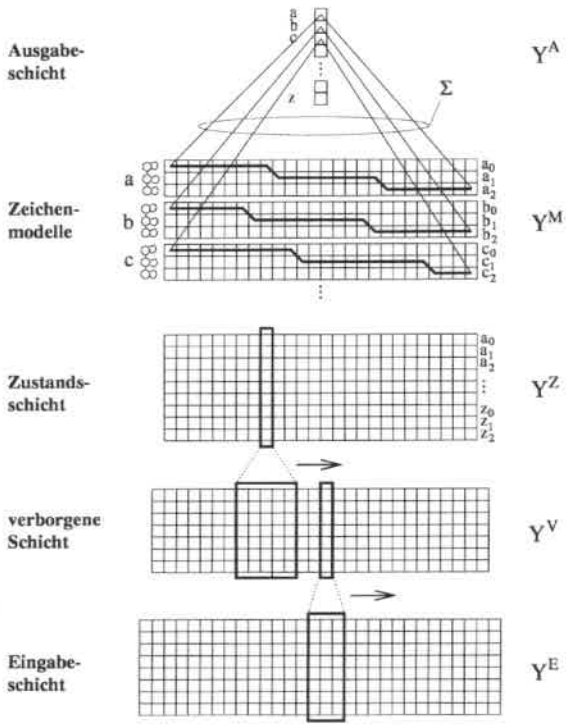


Abbildung 6.6: Die Architektur des MS-TDNN

$P^*(X|\lambda_j) := P(X, \pi^*|\lambda_j)$ durch den Viterbi-Algorithmus bestimmt werden:

$$P^*(X|\lambda_j) = \max_{\pi} \prod_{t=1}^{T} P(\bar{x}_t|q_t = s_{\pi_j(t)}),$$

wobei T die Anzahl der Vektoren in der Zustandsschicht angibt. Die Übergangswahrscheinlichkeiten der HMMs sind wie in den meisten HMM-Systemen gleichwahrscheinlich gesetzt, d.h. sie brauchen hier nicht berücksichtigt zu werden.

6.4 Training des MS-TDNN

Das MS-TDNN besitzt ebenso wie das TDNN oberhalb der Zustandsschicht keine lernbaren Parameter, da die Bewertungen entlang der mit dem Viterbi-Algorithmus bestimmten Pfade ohne weitere Gewichtung in den Neuronen der Ausgabeschicht aufsummiert werden. Während des Trainings des MS-TDNN werden demnach le-

diglich die Schichten bis einschließlich der Zustandsschicht einbezogen. Auf der Zustandsschicht wird der aktuelle Fehler für ein gegebenes Trainingsmuster x_1^T entsprechend der gewählten Fehlerfunktion E berechnet und von dort ausgehend der Fehler durch das Netz zurückpropagiert. Zu erlernen ist eine 1-aus-N Repräsentation der Zustände, d.h. der Sollwert $d_{t,i}$ für den i-ten Zustandsknoten $y_{t,i}^Z$ zum Zeitpunkt t ist gleich 1, wenn es sich zu diesem Zeitpunkt in dem betrachteten Kontext der Eingabe mehrheitlich um Merkmalsvektoren des Zustands s_i handelt, sonst 0 (siehe Abbildung 6.7). Dies setzt jedoch voraus, daß für jeden Merkmalsvektor x_t der Eingabesequenz $x_1^T = x_1 \dots x_T$ bekannt ist, welchem Zustand $s_i \in S$ er zugeordnet ist, um die Sollwerte zu bestimmen. Aufgrund der hier verwendeten abstrakten Modellierung der einzelnen Zeichen durch eine Folge von 3 Zuständen existieren keine Daten, die diese Bedingung erfüllen. Daher ist ein mehrstufiges Training erforderlich, das es nach einer kurzen Trainingsphase, in der die Zustandsgrenzen durch Drittelung festgelegt werden, erlaubt, die Zustandsgrenzen und später auch Zeichen- und Wortgrenzen mit dem Viterbi-Algorithmus automatisch zu bestimmen.

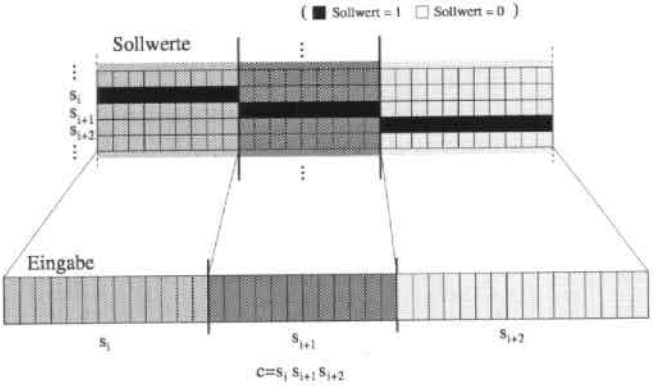


Abbildung 6.7: Berechnung der Sollwerte für ein Zeichen $c \in C$

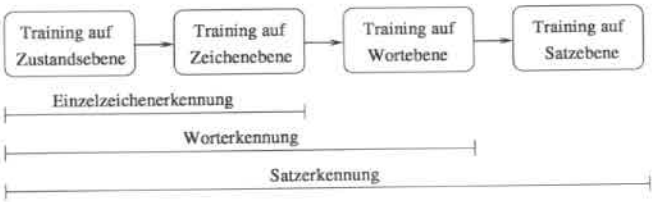


Abbildung 6.8: Die Trainingsschritte des MS-TDNN im Überblick

Das vollständige Training des MS-TDNN für die Handschrifterkennung ist ein vierstufiger Prozeß, der für die Einzelzeichenerkennung bereits nach der zweiten und

für die Worterkennung nach der dritten Stufe endet, wie in Abbildung 6.8 dargestellt ist. Das Lernen in diesem System erfolgt im „On-Line"-Verfahren, d.h. nach jeder Präsentation eines Lernmusters und Rückpropagierung des Fehlers für dieses Muster werden die Gewichte des Netzes aktualisiert. Die Trainingsmenge für eine bestimmte Erkennungsaufgabe wird aus allen entsprechend der in Kapitel 5 vorgestellten Verfahren vorverarbeiteten Trainingsmustern der drei Datenbasen für diese Aufgabe gebildet. In jeder Trainingsiteration werden alle Muster der Trainingsmenge in zufälliger Reihenfolge dem Netz präsentiert. Am Ende einer jeden Iteration wird die Erkennungsrate auf der zugehörigen Kreuzvalidierungsmenge gemessen und so die bis zu diesem Zeitpunkt erreichte Generalisierung ermittelt. Ist über mehrere Trainingsiterationen hinweg keine weitere Verbesserung der Generalisierung auf der Validierungsmenge festzustellen, werden das Training beendet und die Parameter W^* des Netzes, mit denen die beste Generalisierung zu beobachten war, für die abschließende Evaluierung des Systems auf der Testmenge gewählt. Der Wechsel von einer Trainingsstufe in die nächste wird vor Beginn des Trainings heuristisch festgelegt und richtet sich unter anderem nach der Größe der für eine bestimmte Stufe verfügbaren Trainingsmenge.

In diesem Kapitel werden zunächst nur die beiden ersten Trainingsstufen für die Einzelzeichenerkennung, d.h. das Training auf Zustands- und Zeichenebene erläutert (Abbildung 6.9). Die Beschreibung des zusätzlichen Trainings auf Wort- und Satzebene folgt in den anschließenden Kapiteln 7 und 8.

Abbildung 6.9: Trainingsschritte für Einzelzeichenerkennung

6.4.1 Training auf der Zustandsebene

Das Training auf der Zustandsebene für die Einzelzeichenerkennung findet im Gegensatz zum Training für die Worterkennung auf allen verfügbaren Mustern der Trainingsmenge statt, da es sich bei jedem Muster jeweils nur um ein einziges Zeichen handelt, d.h. keine Zeichengrenzen berücksichtigt werden müssen. Für jeden Merkmalsvektor x_t eines Musters x_1^T ist daher bekannt, zu welcher Zeichenklasse er gehört. Jedoch ist nicht bekannt, welchen Zustand dieser Zeichenklasse er repräsentiert. Um dennoch die Sollwerte $d_{t,i}$ für jeden Ausgabeknoten $y_{t,i}^Z$ zu bestimmen, werden die

verfügbaren Ausgabevektoren der Zustandsschicht zunächst gleichmäßig auf die 3 Zustände verteilt, wie in Abbildung 6.10 dargestellt ist. Dies entspricht der ursprünglichen Motivation der 3 abstrakten Zustände, den „Anfangs-", „Mittel-" und „Endteil" eines Zeichens zu modellieren. Der Pfad, der die gesuchten Sollwerte für ein Muster festlegt, wechselt also jeweils nach einem Drittel der Gesamtlänge der Zustandsschicht in den nächsten Zustand über. Üblicherweise wird hier diese Art des Trainings für etwa 3-4 Iterationen beibehalten, bis zum Training auf Zeichenebene gewechselt wird.

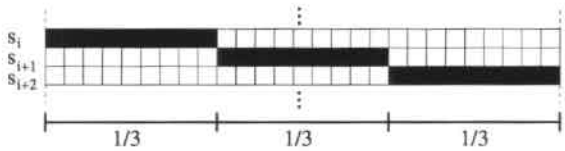


Abbildung 6.10: Berechnung der Sollwerte beim Training auf Zustandsebene

6.4.2 Training auf der Zeichenebene

Die Zuordnung der Zustände eines Trainingsmusters zu den Ausgabevektoren durch einfache Drittelung der Gesamtlänge der Zustandsschicht entspricht in der Regel nicht der tatsächlich optimalen Aufteilung, die sowohl für alle Zeichenklassen als auch für jedes Trainingsmuster individuell verschieden ist. Die endgültige Aufteilung der Zustände soll daher durch das Training auf Zeichenebene von dem MS-TDNN selbst bestimmt werden. Dazu wird für jedes präsentierte Trainingsmuster anstelle der Drittelung mit dem Viterbi-Algorithmus ein bester Pfad $\pi(t)$ durch das Modell des zugehörigen Zeichens bestimmt, wie Abbildung 6.11 zeigt. Die gesuchten Sollwerte $d_{t,i}$ für das Trainingsmuster werden entsprechend entlang des Pfades als 1 (d.h. $d_{t,\pi(t)} = 1$) gewählt, sonst 0. Das vorausgehende Training auf Zustandsebene unterstützt, daß die Zustände eines Zeichens in etwa gleichmäßig auf die Länge des Zeichens verteilt werden, die Pfade durch die Matrix des Zeichens also bereits begonnen haben, sich auszubilden. Ein direktes Training auf Zeichenebene würde in vielen Fällen dazu führen, daß ein einziger Zustand die Lernaufgabe für den Großteil eines Zeichens erfüllen würde und die verbleibenden zwei Zustände jeweils nur noch in einigen wenigen Zeitschritten durchlaufen würden.

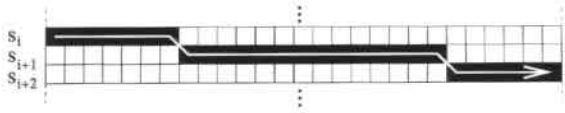


Abbildung 6.11: Berechnung der Sollwerte beim Training auf Zeichenebene

6.4.3 Einzelzeichenerkennung mit dem MS-TDNN

Für alle Experimente dieser Arbeit wurde einheitlich für alle Erkennungsaufgaben eine Modellierung der Zeichen mit drei Zuständen gewählt, obwohl sich in einzelnen Fällen der Einzelzeichenerkennung mit mehr Zuständen noch eine Leistungsverbesserung erzielen läßt, wie Abschnitt 6.5.3 zeigt. Ein Vergleich der Erkennungsraten zwischen der TDNN und MS-TDNN Architektur für die Einzelzeichenerkennung ist in Tabelle 6.2 angegeben. Über alle drei Erkennungsaufgaben und alle Datenbasen hinweg ist durch die Modellierung der Zeichen mit drei Zuständen eine deutliche Fehlerreduktion zu beobachten.

	UKA	CMU	MIT	Gesamt
0...9:				
TDNN	94.2%	94.9%	94.5%	**94.6%**
MS-TDNN	97.9%	97.0%	95.3%	**96.5%**
A...Z:				
TDNN	90.8%	87.8%	94.0%	**90.8%**
MS-TDNN	93.4%	90.5%	94.9%	**92.7%**
a...z:				
TDNN	85.2%	87.1%	89.5%	**87.7%**
MS-TDNN	89.7%	89.8%	93.4%	**91.1%**

Tabelle 6.2: Vergleich der Erkennungsraten von TDNN und MS-TDNN mit drei Zuständen pro Zeichen

6.5 Architekturparameter

Die Erkennungsleistung eines MS-TDNN hängt von der Wahl der Architekturparameter ab. Die optimale Bestimmung der Anzahl der Neuronen in der verborgenen Schicht, der Breite der Kontexte in der Eingabe- und der verborgenen Schicht und nicht zuletzt auch die Anzahl der Zustände für die Modellierung der Zeichen ist im allgemeinen ein aufwendiger Prozeß, da diese Architekturparameter von vielen Faktoren abhängig sind. Für jede Erkennungsaufgabe und für jeweils unterschiedlich große Trainingsmengen gibt es unterschiedliche Größen, die zu der optimalen Leistung führen. Viele der in dieser Arbeit verwendeten Netzgrößen basieren auf Erfahrungswerten, die während der Entwicklung dieses Systems gesammelt wurden, da alleine aufgrund der großen Datenmengen die Trainingszeiten für einen einzigen Erkenner bis zu 2 Wochen betragen können. Dies läßt nur eine begrenzte Anzahl von Experimenten zu. Eine gezielte manuelle Optimierung der MS-TDNN Architektur für jede Erkennungsaufgabe und hier verwendete Trainingsmenge ist daher ebensowenig möglich wie eine automatische Optimierung, wie sie für das MS-TDNN z.B. in [BM93, BMW93] für die Handschrifterkennung und in [BH95] für die Spracherkennung vorgeschlagen wurde. Am Beispiel der Einzelzeichenerkennung soll hier dennoch in Ansätzen gezeigt

werden, welchen Einfluß die Architekturparameter auf die Erkennungsleistung haben. Dabei wird hier eine Unabhängigkeit der Parameter untereinander vorausgesetzt, die in der Realität nicht existiert. Bei Änderung eines Parameters, z.B. der Breite des Eingabekontextes, müßten alle anderen Parameter entsprechend neu eingestellt werden. Dieses Problem wird in erster Linie durch die starke Strukturierung des MS-TDNN hervorgerufen.

6.5.1 Fensterbreite

Die Fensterbreite in der Eingabe- und der verborgenen Schicht bestimmt den Gesamtkontext, aus dem ein einziger Ausgabevektor der Zustandsschicht berechnet wird. Sind $b_E = 2d_E + 1$ und $b_V = 2d_V + 1$ die Fensterbreiten der Eingabe- bzw. der verborgenen Schicht und wird von der hier verwendeten Schrittweite von 2 in der Eingabeschicht ausgegangen, berechnet sich der Gesamtkontext für einen Ausgabevektor aus $b_E + 2(b_V - 1)$. Für $b_E = 7$ und $b_V = 7$, wie hier für alle Experimente verwendet, beträgt der Gesamtkontext also 19 Eingabevektoren. Tabelle 6.3 zeigt für unterschiedliche andere Kombinationen von Fenstergrößen die Erkennungsraten für die Erkennung von Ziffern, Klein- und Großbuchstaben. Die Differenzen der Er-

b_E	b_V	Kontext	0_9	A_Z	a_z
5	5	13	96.2%	91.6%	89.4%
6	6	16	96.7%	92.0%	90.9%
7	7	19	96.5%	92.7%	91.1%
8	8	22	96.2%	92.3%	91.3%
9	9	25	96.0%	92.2%	91.2%
6	8	20	96.2%	92.0%	90.8%
8	6	18	96.4%	92.9%	91.0%
5	9	21	95.3%	91.6%	90.2%
9	5	17	95.8%	92.0%	90.7%

Tabelle 6.3: Erkennungsraten für verschiedene Fenstergrößen

kennungsraten für die unterschiedlichen Kombinationen der Fensterbreiten sind in allen Fällen sehr gering. Erst bei größeren Änderungen zeigen sich deutlichere Verschlechterungen. Die Wahl der Fensterbreite ist in erster Linie auch von der Wahl der Abtastrate während der Vorverarbeitung abhängig. Wird diese geändert, müßten die Fensterbreiten entsprechend neu optimiert werden. Hier wurde allerdings der umgekehrte Weg beschritten und die Abtastrate für die gegebene Fensterbreite von 7 optimiert, wie in Kapitel 5.3.6 gezeigt.

6.5.2 Anzahl der Neuronen in der verborgenen Schicht

Nachdem durch die Fensterbreiten der zeitliche Kontext festgelegt wurde, kann durch die Wahl der Anzahl von Neuronen in der verborgenen Schicht die Anzahl der lernba-

ren Gewichte und damit die Lernkapazität des Netzes reguliert werden. Die Anzahl der Gewichte des MS-TDNN berechnet sich bei gegebenen Fenstergrößen b_E und b_V durch $b_E n_E n_V + b_V n_V n_Z$, wenn n_E, n_V und n_Z die Anzahl der Neuronen in der Eingabe-, verborgenen und Zustandsschicht angeben. In Tabelle 6.4 sind für unterschiedliche Größen der verborgenen Schicht die Anzahl der Gewichte und die erzielte Erkennungsrate angegeben. Im Fall des MS-TDNN zur Ziffernerkennung ist für mehr

	0..9		A..Z		a..z	
# Neuronen	# Gewichte	%	# Gewichte	%	# Gewichte	%
20	6860	95.1%	13580	-	13580	-
30	10290	96.5%	20370	-	20370	-
40	13720	96.6%	27160	-	27160	-
50	17150	96.3%	33950	89.6%	33950	88.4%
60	20580	96.4%	40740	91.3%	40740	89.1%
70	24010	-	47530	92.4%	47530	90.3%
80	27440	-	54320	92.7%	54320	91.1%
90	30870	-	61110	92.6%	61110	91.2%
100	43300	-	67900	92.7%	67900	90.8%

Tabelle 6.4: Erkennungsraten für verschiedene Größen der verborgenen Schicht

als 30 Neuronen in der verborgenen Schicht nur noch eine marginale Verbesserung zu verzeichnen, während bei weniger als 30 Neuronen die Erkennungsaufgabe deutlich schlechter gelöst werden kann. Ähnlich verhält es sich für die Erkennung von Klein- und Großbuchstaben. Dort ist ab etwa 80 verborgenen Neuronen keine signifikante Verbesserung der Erkennungsrate mehr zu beobachten.

6.5.3 Anzahl der Zustände

Einen weitaus größeren Einfluß auf die Erkennungsrate hat die gewählte Anzahl der Zustände zur Modellierung der einzelnen Zeichen, wie bereits beim Wechsel von der TDNN zur MS-TDNN Architektur deutlich wurde. In dieser Arbeit wurde einheitlich eine Modellierung durch 3 Zustände pro Zeichen verwendet. Wie Kapitel 7.2 zeigt, ist dies die Anzahl, die zumindest für die Einzelworterkennung das beste Erkennungsergebnis erzielt. Für die Einzelzeichenerkennung kann auf einzelnen Testmengen jedoch durch eine größere Anzahl von Zuständen noch eine leichte Verbesserung erzielt werden. Tabelle 6.5 faßt die Ergebnisse für eine unterschiedliche Anzahl von Zuständen für die Erkennung von Ziffern-, Klein- und Großbuchstaben zusammen. Die Modellierung der Zeichen durch weniger als 3 Zustände führt zu einer deutlichen Erhöhung der Fehlerrate auf allen getesteten Datenmengen. Für Kleinbuchstaben führt eine weitere Erhöhung der Anzahl zu keinen signifikanten Verbesserungen. Bei der Erkennung von Großbuchstaben führt offensichtlich eine Modellierung durch 4 Zustände zum optimalen Ergebnis.

	UKA	CMU	MIT	Gesamt
0...9:				
1 Zustand	94.2%	94.9%	94.5%	**94.6%**
2 Zustände	97.4%	96.1%	94.8%	**95.8%**
3 Zustände	97.9%	97.0%	95.3%	**96.5%**
4 Zustände	98.4%	98.1%	95.5%	**97.2%**
5 Zustände	99.5%	97.6%	95.7%	**97.3%**
A...Z:				
1 Zustand	90.8%	87.8%	94.0%	**90.8%**
2 Zustände	91.7%	89.9%	93.9%	**91.8%**
3 Zustände	93.4%	90.5%	94.9%	**92.7%**
4 Zustände	94.0%	91.9%	95.6%	**93.7%**
5 Zustände	92.9%	92.1%	95.4%	**93.6%**
a...z:				
1 Zustand	85.2%	87.1%	89.5%	**87.7%**
2 Zustände	89.1%	89.3%	90.8%	**89.8%**
3 Zustände	89.7%	89.8%	93.4%	**91.1%**
4 Zustände	90.5%	90.4%	92.5%	**91.2%**
5 Zustände	90.3%	90.0%	93.2%	**91.3%**

Tabelle 6.5: Erkennungsraten mit unterschiedlicher Anzahl von Zuständen

6.6 Beiträge dieser Arbeit

Time Delay Neural Networks sind in der On-line Handschrifterkennung eine in vielen Systemen anzutreffende Architektur. Allerdings werden in diesen Systemen die Zeichen jeweils nur durch einen einzigen Ausgabeknoten repräsentiert, eine Unterteilung der Zeichen in kleinere Einheiten erfolgt nicht. Mit der vorliegenden Arbeit wird jedoch gezeigt, daß durch geeignete Modellierung der Zeichen eine deutliche Verbesserung der Erkennungsleistung erzielt werden kann. Eine explizite und bedeutungstragende Segmentierung der Zeichen in einzelne Zustände ist hierzu nicht erforderlich. Mit der MS-TDNN Architektur werden die Fähigkeiten eines HMMs, zeitliche Abfolgen zu modellieren, mit den diskriminativen Fähigkeiten eines neuronalen Netzes kombiniert. Während für die Einzelzeichenerkennung mit speziell hierfür optimierten Systemen bereits bessere Ergebnisse präsentiert wurden [GAL+91], zeigt sich bei der Wort- und Satzerkennung, die in den folgenden Kapiteln behandelt werden, daß durch diesen hybriden Ansatz sowohl rein auf neuronalen Netzen als auch rein auf HMMs basierende Systeme deutlich übertroffen werden.

Kapitel 7

Worterkennung mit Vokabularen

7.1 Einleitung

Das vorhergehende Kapitel beleuchtete die MS-TDNN Architektur für die Erkennung einzelner Zeichen. Dieser Ansatz soll in diesem Kapitel nun auf die Erkennung ganzer Wörter, also Folgen einzelner Zeichen, erweitert werden. Dazu werden unterschiedliche Methoden mit und ohne Integration syntaktischen Wissens vorgestellt und durch Evaluierung auf den Datenbasen miteinander verglichen.

In allen auf neuronalen Netzen basierenden Systemen der Handschrifterkennung werden Vokabulare bislang als nachgeschalteter Verarbeitungsschritt an den Erkennungsprozeß angefügt. Die Erzeugung einer Sequenz von Buchstabenhypothesen aus den Ausgabevektoren des neuronalen Netzes erfolgt dort also ohne das syntaktische Wissen. In dieser Arbeit wird gezeigt, daß durch direkte Integration des Vokabulars in den Suchprozeß die Erkennungsrate deutlich gegenüber solchen nachgeschalteten Ansätzen verbessert werden kann. Das beste Ergebnis wird hier mit einer Baumsuche erzielt, die zudem durch Pruning-Methoden eine Erkennung der Eingabe in Echtzeit ermöglicht.

Die folgenden zwei Abschnitte stellen zunächst die Erweiterungen der MS-TDNN Architektur und des Trainings für die Worterkennung vor, bevor anschließend die unterschiedlichen Möglichkeiten der Worterkennung mit dem MS-TDNN diskutiert werden.

7.2 Die MS-TDNN Architektur für die Worterkennung

Da die Wörter der drei Datenbasen sowohl Klein- als auch Großbuchstaben enthalten, ist gegenüber der Einzelzeichenerkennung, bei der Klein- und Großbuchstaben durch getrennte Systeme erkannt wurden, eine Erweiterung der Zustandsschicht erforderlich. Durch die Verdopplung der zu unterscheidenden Zeichenklassen auf 52 enthält sie nun insgesamt 156 Zustände, die jeweils einem Neuron zugeordnet sind, wie Abbil-

dung 7.1 zeigt. Da bei der Worterkennung 3 zusätzliche Merkmale („Hut"-Merkmal,

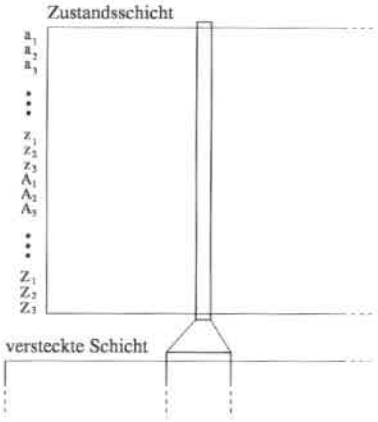


Abbildung 7.1: Die Zustandsschicht des MS-TDNN für die Worterkennung

Ober-/Unterlängen) mit in die Merkmalsvektoren aufgenommen werden, muß die Eingabeschicht ebenfalls um 3 weitere Neuronen ergänzt werden. Die Anzahl der Neuronen in der verborgenen Schicht wird von 80 auf 120 erhöht, um die erschwerte Lernaufgabe und die größere Trainingsmenge auszugleichen. Die Breite der Kontexte in der Eingabe- und der verborgenen Schicht wird von der Einzelzeichenerkennung übernommen. Auch diese Architekturparameter wurden analog den Experimenten in Kapitel 6.5 empirisch bestimmt. Wie bereits im vorhergehenden Kapitel erwähnt, hat sich die Modellierung der einzelnen Zeichen durch jeweils 3 Zustände für die Worterkennung als optimal erwiesen, auch wenn für einzelne Teilmengen der Datenbasen mit einer höheren Anzahl geringfügig bessere Erkennungsraten erzielt werden. Tabelle 7.1 zeigt einen Vergleich der Worterkennungsraten für unterschiedliche Modellierungen.

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
1 Zustand	92.4%	75.9%	85.6%	76.7%	89.0%	**83.6%**
2 Zustände	97.1%	88.6%	92.8%	88.4%	92.3%	**91.8%**
3 Zustände	97.1%	90.8%	92.8%	92.2%	94.7%	**93.4%**
4 Zustände	98.1%	89.9%	94.3%	92.0%	93.9%	**92.9%**
5 Zustände	96.8%	90.3%	93.7%	93.9%	92.9%	**92.8%**

Tabelle 7.1: Erkennungsraten mit unterschiedlicher Anzahl von Zuständen

Das MS-TDNN hat bei dieser Konfiguration der Netzgrößen über 31 500 lernbare Gewichte, die für jeden Ausgabevektor der Zustandsschicht durchlaufen werden müssen. Um die Berechnungszeit insbesondere bei langen Eingabesequenzen zu

verkürzen, wird wie bei der Einzelzeichenerkennung das Fenster der Eingabeschicht mit jedem Zeitschritt um jeweils 2 Merkmalsvektoren verschoben, so daß die verborgene Schicht nur noch etwa halb so lang wie die Eingabeschicht ist. Das Fenster in der verborgenen Schicht wird weiterhin nur um einen Schritt verschoben. Die so erreichte Halbierung der Länge der Zustandsschicht halbiert zusätzlich auch den nachfolgenden Suchaufwand nach der Worthypothese des Systems aus den Bewertungen der Zustandsschicht. Die gleiche Reduzierung des Berechnungsaufwandes kann auch durch

	UKA PRINTED/CURSIVE	CMU PRINTED/ CURSIVE	MIT PRINTED	Gesamt
Abtastrate	97.5%/89.9%	93.7%/90.2%	92.4%	**91.8%**
Schrittweite	97.1%/90.8%	92.8%/92.2%	94.7%	**93.4%**

Tabelle 7.2: Worterkennungsraten bei Schrittweite 2 im Vergleich zu halbierter Abtastrate

die Halbierung der Abtastrate während der Vorverarbeitung erzielt werden. Dies hat jedoch zur Folge, daß auch die Auflösung des Eingabesignals halbiert wird, was in einer schlechteren Erkennungsleistung resultiert. Tabelle 7.2 stellt die Ergebnisse mit halbierter Abtastrate den Ergebnisse mit einer Schrittweite von 2 in der Eingabeschicht gegenüber. Nicht nur die schlechtere Auflösung des Eingabesignals, sondern auch der geringere Gesamtkontext, der für die Berechnung eines Ausgabevektors der Zustandsschicht betrachtet wird, trägt zu der Verschlechterung des Ergebnisses bei. Bei einer Fensterbreite von 7 sowohl in der Eingabe- als auch der verborgenen Schicht beträgt der betrachtete Kontext bei der halbierten Abtastrate und Schrittweite 1 nur 13 Merkmalsvektoren, bei der Verdopplung der Schrittweite jedoch 19 Merkmalsvektoren. Abbildung 7.2 zeigt abschließend die Gesamtarchitektur des MS-TDNN für die Worterkennung mit tatsächlichen Aktivierungen am Beispiel der Erkennung des Wortes „Clinton".

7.3 Training des MS-TDNN für die Worterkennung

Gegenüber der Einzelzeichenerkennung wird das Training für die Worterkennung um eine weitere Trainingsstufe ergänzt, wie Abbildung 7.3 zeigt. Für das Training auf Zustands- und Zeichenebene werden hier nicht mehr die Einzelzeichen der Trainingsmenge, sondern alle Einzelworte der Datenbasen, die manuell mit den Buchstabengrenzen etikettiert wurden, verwendet. Dies ist nur eine kleine Teilmenge der Gesamttrainingsdaten. Anstelle eines einzelnen Zeichens besteht ein Trainingsmuster x_1^T nun aus einem vollständigen Wort, für das die Zustandsbewertungen durch das MS-TDNN berechnet werden. Die Sollwerte beim Training auf Zustands- und Zeichenebene werden für die manuell segmentierten Wörter analog der Bestimmung bei Einzelzeichen bestimmt, diesmal jedoch innerhalb der bekannten Buchstabengrenzen, wie in Abbil-

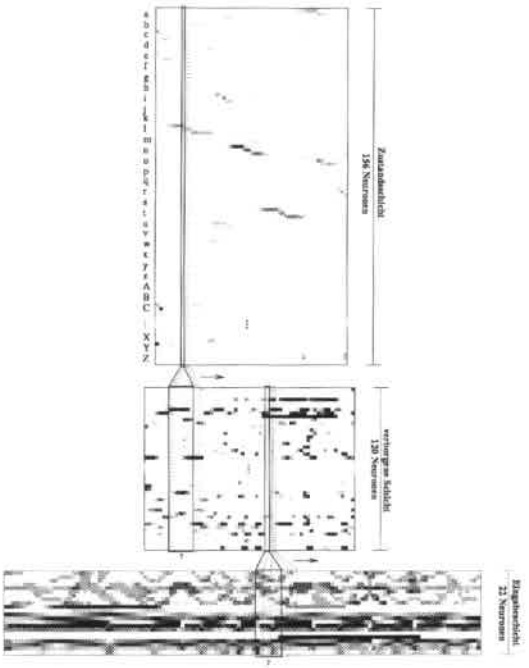


Abbildung 7.2: Das MS-TDNN beim Erkennen des Wortes „Clinton"

dung 7.4 am Beispiel des Wortes „Clinton" dargestellt ist. Für jeden Buchstaben der Eingabe werden die Sollwerte separat durch Drittelung oder den Viterbi-Algorithmus ermittelt.

Wurde das MS-TDNN ausreichend lange auf der Zeichenebene trainiert, kann das Training auf die Wortebene ausgedehnt werden. Erst dieser Schritt erschließt die große Menge der Trainingsdaten, die nicht auf Buchstabenebene segmentiert wurden. Während die segmentierten Wörter der Trainingsmenge weiterhin auf der Zeichenebene trainiert werden, müssen für unsegmentierte Wörter die Sollwerte automatisch durch den Viterbi-Algorithmus über das gesamte Wortmodell bestimmt werden, wie Abbildung 7.5 für das Wort „Clinton" zeigt. Erst durch das vorausgehende Training auf Zustands- und Zeichenebene wird der Einsatz des Viterbi-Algorithmus ermöglicht, da sich bis zu diesem Zeitpunkt die Pfade für unsegmentierte Wörter bereits deutlich ausgebildet haben. Das Training auf Wortebene ist nicht nur notwendig, um unsegmentierte Daten zu erschließen, sondern ermöglicht dem MS-TDNN, nun auch zusätzlich Buchstabenübergänge zu interpretieren, die bei den vorausgehenden Trainingsschritten nicht berücksichtigt wurden, da nur innerhalb der Buchstabengrenzen trainiert wurde. Das Training auf Wortebene für unsegmentierte und auf Zeichene-

Abbildung 7.3: Trainingsschritte für die Worterkennung

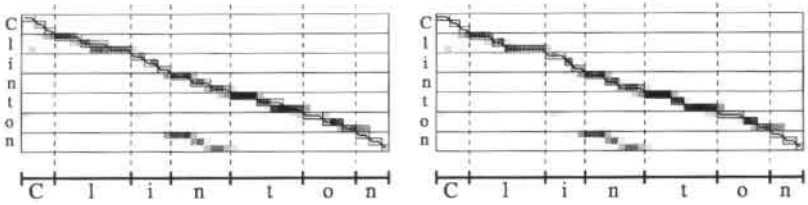


Abbildung 7.4: Berechnung der Sollwerte beim Training auf Zustands- (links) und Zeichenebene (rechts) bei der Worterkennung

bene für segmentierte Daten wird solange fortgesetzt, bis über mehrere Iterationen hinweg keine Verbesserung der Generalisierung auf der Kreuzvalidierungsmenge mehr zu beobachten ist.

7.4 Das MS-TDNN mit freier Suche für die Worterkennung

Bei der Einzelzeichenerkennung enthielt die Zustandsschicht des MS-TDNN die Zustandsbewertungen für ein einziges isoliertes Zeichen in der Eingabe. Die Pfade durch die Bewertungsmatrizen wurden für alle Zeichenklassen innerhalb der gegebenen Modelle der einzelnen Zeichen berechnet (Abbildung 7.6a). Mit dem Ende des Pfades für ein Modell war auch gleichzeitig der letzte Vektor der Zustandsschicht erreicht. Um auch Zeichenfolgen erkennen zu können, muß die Suche des besten Pfades über die Zeichengrenzen hinaus ausgeweitet werden. Ist der letzte Zustand eines Zeichenmodells erreicht, sind Übergänge in den ersten Zustand eines beliebigen anderen Zeichenmodells erlaubt. Dies entspricht der in Abbildung 7.6b dargestellten einfachen Rückkopplung der Modelle. Das Ergebnis dieser Suche ist wiederum ein Pfad durch alle Vektoren der Zustandsschicht, diesmal jedoch durch die Modelle unterschiedlicher Zeichen hindurch. Die Folge der Zeichen, deren Modelle dieser Pfad durchläuft, bildet

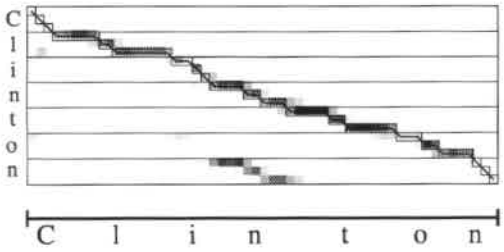


Abbildung 7.5: Berechnung der Sollwerte beim Training auf Wortebene für unsegmentierte Daten

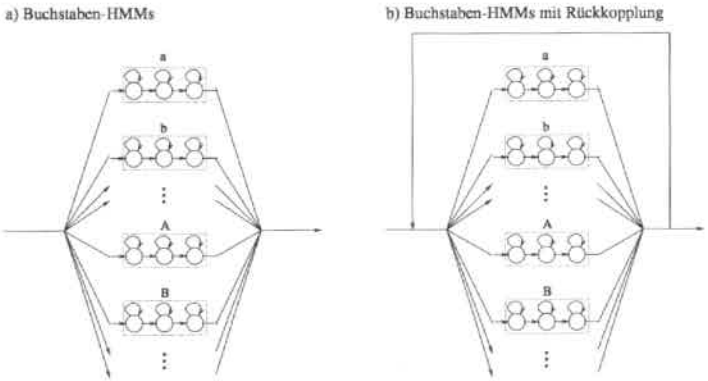


Abbildung 7.6: Erweiterung der Modelle für die Worterkennung

die gesuchte Ausgabehypothese des Erkenners (siehe Abbildung 7.7).

Da bislang keinerlei Einschränkungen bezüglich der erlaubten Zeichenübergänge gemacht wurden, d.h. in welches Modell von dem letzten Zustand eines anderen Modells gewechselt werden darf, können so beliebige Zeichenfolgen erkannt werden. Die Übergänge von einem Modell in alle anderen werden als gleichwahrscheinlich angenommen. Die Ausgabehypothese des Erkenners muß daher nicht zwangsläufig ein legales Wort der zugrundeliegenden Sprache sein. Tatsächlich ist die Worterkennungsrate mit dieser freien, syntaktisch nicht eingeschränkten Suche sehr gering, wie die Testergebnisse auf den drei Datenbasen in Tabelle 7.3 zeigen. Dort sind neben der Worterkennungsrate WA für die einzelnen Datenbasen auch die Buchstabenakkuratheit BA angegeben, d.h. die Erkennungsrate auf Zeichenebene unter Berücksichtigung von Einfüge-, Auslassungs- und Verwechslungsfehlern. Die Ergebnisse zeigen, daß die Wahrscheinlichkeit, alle Zeichen eines Wortes richtig zu erkennen, mit einer freien Suche sehr gering ist. Sie liegt hier mit 26.8% sogar noch über der bei einer durchschnittlichen Wortlänge von 8 Zeichen und 80% Buchstabenakkuratheit in etwa zu

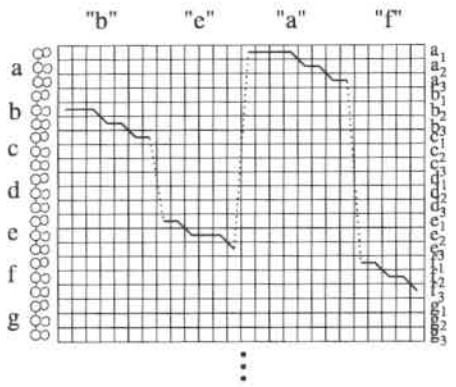


Abbildung 7.7: Freie Suche durch die Zustandsschicht

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
BA	85.0%	70.6%	80.1%	74.3%	83.3%	**79.0%**
WA	37.3%	14.2%	24.7%	17.8%	36.0%	**26.8%**

Tabelle 7.3: Die Wortakkuratheit (WA) und Buchstabenakkuratheit (BA) bei freier Suche

erwartenden Erkennungsrate von 16%. Die Buchstabenakkuratheit und die Worterkennungsrate bei Druckschrift sind für alle Datenbasen jeweils deutlich höher als bei Kursivschrift. Grund hierfür ist zum einen, daß die einzelnen Zeichen bei Druckschrift in der Regel deutlicher geschrieben und zum anderen Zeichenübergänge durch Abheben des Stiftes von der Schreiboberfläche eingeleitet werden.

Selbst bei einer Buchstabenakkuratheit von 90% läge die Worterkennungsrate bei einer durchschnittlichen Wortlänge von 8 Zeichen erst bei etwa 50%. Um eine akzeptable Worterkennungsrate von über 90% zu erzielen, würde eine Buchstabenakkuratheit von über 97% benötigt, die mit einer freien Suche von keinem heutigen Erkennungssystem mit 52 Zeichenklassen erreicht wird. Die einzige Möglichkeit, Worterkennungsraten in einem akzeptablen Bereich zu erreichen, ist daher, die möglichen Zeichenübergänge bei der Suche einzuschränken. Wie dies auf unterschiedliche Arten durch syntaktisches Wissen über die zugrundeliegende Sprache realisiert werden kann, ist Inhalt der folgenden Abschnitte.

7.5 Suche mit Vokabularen

Syntaktisches Wissen über die zugrundeliegende Sprache kann in Form von Vokabularen, d.h. Listen legaler Wörter einer Sprache, entweder nach der Suche als zusätzli-

cher Verarbeitungsschritt oder direkt in den Suchprozeß integriert werden. Im ersten

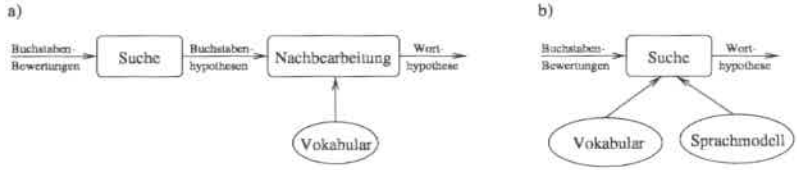


Abbildung 7.8: Möglichkeiten der Integration syntaktischen Wissens

Fall bedeutet dies eine nachträgliche Korrektur der mit der freien Suche erzeugten Folge von Buchstabenhypothesen durch Vergleich mit dem Vokabular. Eine direkte Integration des syntaktischen Wissens in die Suche beschränkt bereits während des Suchvorgangs die Anzahl der möglichen Zeichenübergänge. Die Einschränkung kann z.B. über Sprachmodelle auf Zeichenebene erfolgen, d.h. anhand von Bi- oder Trigrammen werden bestimmte Zeichenübergänge aufgrund ihrer im Vokabular auftretenden Häufigkeit höher bewertet als andere weniger oder gar nicht auftretende Übergänge. Bei dieser Vorgehensweise können weiterhin beliebige Zeichenfolgen erkannt werden, also auch solche, die nicht in dem Vokabular vorkommen, auf dem die Sprachmodelle berechnet wurden. Eine weitere Möglichkeit ist, bei der Suche nur noch Zeichenübergänge zuzulassen, die ein legales Wort des Vokabular erzeugen. Dies ist die restriktivste Möglichkeit der Einschränkung, da keine beliebigen Zeichenfolgen mehr erkannt werden können, sie erzielt dafür aber die besten Erkennungsraten.

Im folgenden Abschnitt wird zunächst eine einfache, der Suche nachgeschaltete Möglichkeit der Integration eines Vokabulars mittels Editierdistanzen beschrieben. Der anschließende Rest des Kapitels widmet sich Methoden der direkten Integration in die Suche. Sofern nicht anders angegeben, wurde bei den präsentierten Ergebnissen das Vokabular WSJ_20K mit 20 000 Wörtern verwendet.

7.6 Hypothesenkorrektur durch Editierdistanzen

Wurde durch die freie Suche eine Worthypothese $w = c_1 c_2 \ldots c_L$ berechnet und ist w ein legales Wort des gegebenen Vokabulars $W = \{w_1, w_2, \ldots, w_N\}$, d.h. ist $w \in W$, so ist w direkt das Ergebnis des Erkennungsprozesses. Ist w jedoch nicht in W enthalten, muß dasjenige Wort $w^* \in W$ ausgewählt werden, das w am ähnlichsten ist, d.h.

$$w^* = \underset{w_i \in W}{\operatorname{argmin}}\, d(w, w_i)$$

Das Abstandsmaß $d(w_i, w_j)$ ist dabei definiert als die in Kapitel 4.4 beschriebene Editierdistanz zwischen den Wörtern w_i und w_j. w^* ist dabei nur in den wenigsten Fällen eindeutig. Insbesondere bei großen Vokabularen existieren meist mehrere Wörter, die dieselbe minimale Distanz zu der Worthypothese haben. In diesen Fällen muß eine weitere Auswahl aus diesen Wörtern getroffen werden.

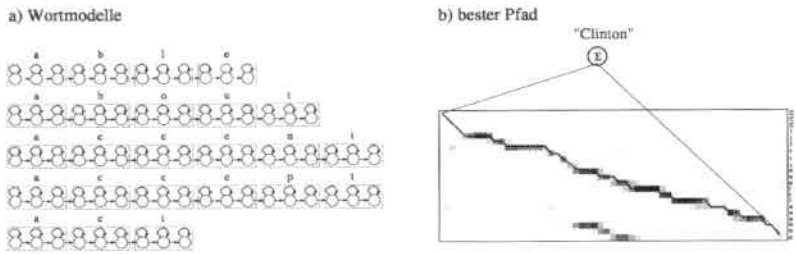


Abbildung 7.9: Wortmodelle und bester Pfad für das Wort „Clinton"

Die Auswahl der endgültigen Worthypothese aus den M Wörtern $\{w_1^*, w_2^*, \ldots, w_M^*\}$ mit minimaler Editierdistanz zu w erfolgt durch Bewertung jedes dieser Wörter anhand der Aktivierungen der bereits bei der freien Suche durchlaufenen Zustandsschicht des MS-TDNN. Für jedes Wort $w_i^* = c_1^i c_2^i \ldots c_{L_i}^i$ werden die Modelle der einzelnen Zeichen c_j^i des Wortes zu einem Wortmodell verbunden (z.B. Abbildung 7.9a) und die Aktivierungen der Zustandsschicht entlang dieses Wortmodells durchlaufen. In der bereits in Kapitel 6.3.2 verwendeten Matrixdarstellung des Modells bedeutet dies wiederum anschaulich, mit dem Viterbi-Algorithmus einen optimalen Pfad durch die Matrix von links oben nach rechts unten zu berechnen, wie in Abbildung 7.9b für das Wort „Clinton" dargestellt ist. Die Bewertung eines Wortes w_i^* ist die Summe der Aktivierungen entlang des besten Pfades für das Modell von w_i^*. Das Wort mit der besten so berechneten Bewertung wird als endgültige Ausgabe gewählt.

Die Verbesserungen sowohl der Buchstabenakkuratheit als auch der Worterkennungsrate durch diese einfache Methode im Vergleich zur Erkennung mit freier Suche zeigt Tabelle 7.4. Im Hinblick auf die Erkennungsrate erweist sich dieses Verfahren

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
BA ohne	85.0%	70.6%	80.1%	74.3%	83.3%	**79.0%**
WA ohne	37.3%	14.2%	24.7%	17.8%	36.0%	**26.8%**
BA mit	97.4%	90.2%	96.8%	92.1%	95.8%	**94.2%**
WA mit	93.9%	80.6%	90.2%	83.0%	90.8%	**87.2%**

Tabelle 7.4: Buchstabenakkuratheit und Wortakkuratheit mit Vergleich einer Wortliste und ohne

also bereits als sehr effektiv. Die Fehlerrate auf Wortebene wird um etwa 80% gesenkt. Das Vokabular kann auf einfache Art und Weise ausgetauscht und sogar zur Laufzeit erweitert oder geändert werden. Allerdings steigt der Berechnungsaufwand für die Editierdistanzen linear mit der Größe des Vokabulars, dem praktischen Einsatz sind damit bereits ab Vokabulargrößen von 10 000 Wörtern Grenzen gesetzt. Es eignet sich

daher eher für Anwendungen, die kleinere Vokabulare erfordern. Dieser Ansatz ist nur für die Satzerkennung erweiterbar, wenn eine explizite Segmentierung der Eingabe in einzelne Worte erfolgt, da eine vollständige Modellierung aller Sätze einer Sprache nicht möglich ist. Aus diesem Grund bleiben insbesondere auch die in Kapitel 3 vorgestellten Systeme von Schenkel[Sch95] und Seni[Sen95] auf die Einzelworterkennung beschränkt.

Eine Analyse der obigen Ergebnisse zeigt, daß die Gesamterkennungsrate von 87.2% auch gleichzeitig in etwa die Anzahl der Fälle angibt, in denen die korrekte Antwort tatsächlich in der Liste der Wörter mit minimaler Editierdistanz vorhanden ist. D.h. wenn das korrekte Wort unter den Wörtern mit minimaler Distanz zur Worthypothese ist, so wird dieses auch anhand der nachfolgenden Bewertung identifiziert. Diese Beobachtung führt zu einem einfachen weiteren Experiment, in dem nicht mehr nur die Wörter mit minimaler Distanz d bewertet werden, sondern noch zusätzlich alle Wörter mit der Distanz $d+1$. Das erhöht zwar den Gesamtberechnungsaufwand durch die zusätzlichen Bewertungen, ergibt aber eine weitere Verbesserung der Worterkennungsrate auf 88.4%.

Diese Art der nachgeschalteten Suche ist eine Variante der in [Sch95, Sen95] vorgestellten Ansätze.

7.7 Suche mit Sprachmodellen

Die schwächste Form einer Einschränkung der Suche mittels eines Vokabulars sind Sprachmodelle (hier Bi- und Trigramme) auf Buchstabenebene, die direkt in den Suchprozeß integriert werden, da sie weiterhin die Erkennung beliebiger Zeichenfolgen zulassen. Das Vokabular kommt lediglich bei der Berechnung der Sprachmodelle, aber nicht mehr bei der Suche selbst zum Einsatz. Das Sprachmodell stellt die a priori Wahrscheinlichkeiten für das Auftreten bestimmter Buchstabenkombinationen zur Verfügung. Die Wahrscheinlichkeiten $P(c_i|c_{i-1})$ für das Bigramm und $P(c_i|c_{i-2}c_{i-1})$ für das Trigramm werden auf einem möglichst großen Vokabular geschätzt. In der freien Suche wird nun die Entscheidung, in welches neue Zeichenmodell aus dem letzten Zustand des vorhergehenden Zeichenmodells gewechselt werden soll, nicht mehr nur anhand der Aktivierungen der Zustandsschicht getroffen, sondern zusätzlich bei dem Sprachmodell für jedes mögliche Zeichen die Bigramm- oder Trigrammwahrscheinlichkeit angefragt, daß das Zeichen auf das vorhergehende bzw. die zwei vorhergehenden Zeichen folgt. So werden in der Sprache häufig vorkommende Buchstabenfolgen wie z.B. „on" oder „ion" bevorzugt gegenüber selten oder gar nicht vorkommenden Folgen wie „qw" oder „yy".

Tabelle 7.5 zeigt einen Vergleich der Ergebnisse einer freien Suche mit der durch ein Bigramm bzw. Trigramm eingeschränkten Suche sowohl für die Buchstabenakkuratheit als auch die eigentlich interessierende Worterkennungsrate. Sprachmodelle reduzieren zwar den Fehler auf Buchstabenebene um etwa 25%, die Worterkennungsrate bleibt aber weiterhin deutlich hinter der Leistung der Suche mit Editierdistanzen und

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
freie Suche:						
BA	85.0%	70.6%	80.1%	74.3%	83.3%	**79.0%**
WA	37.3%	14.2%	24.7%	17.8%	36.0%	**26.8%**
Bigramm:						
BA	89.4%	77.1%	85.7%	80.5%	87.6%	**84.2%**
WA	44.6%	18.2%	30.2%	24.3%	45.0%	**33.8%**
Trigramm:						
BA	89.8%	77.5%	86.1%	81.1%	88.4%	**84.8%**
WA	45.5%	18.6%	31.9%	25%	46.3%	**34.9%**

Tabelle 7.5: Wortakkuratheit und Buchstabenakkuratheit mit und ohne Sprachmodellen auf Buchstabenebene

auch der folgenden Suchverfahren zurück. Die Sprachmodelle können im übrigen auch zusätzlich in die Suche mit Editierdistanzen integriert werden. So ergeben sich dort bereits bei der Berechnung der Zeichenhypothesen bessere Ergebnisse, wodurch auch die Berechnung der Wörter mit minimaler Editierdistanz genauer durchgeführt werden kann. Die gesuchte Worthypothese befindet sich also häufiger unter den Wörtern mit minimaler Editierdistanz, und die Gesamterkennungsrate kann weiter auf 89.3% verbessert werden.

7.8 Flache Suche

Das beste Erkennungsergebnis liefert eine direkte vokabulargesteuerte Suche, die von vornherein nur legale Wörter, d.h. Wörter aus einem vorgegebenen Vokabular $W = \{w_1, w_2, \ldots, w_N\}$ als Ausgabe erzeugt [MFW95b, MFW95a]. Der Fall, daß die Ausgabehypothese w nicht in W enthalten ist, tritt also nicht auf. Dies ist die strengste Form der Einschränkung des Suchprozesses. Es soll nun zunächst eine einfache vokabulargesteuerte Suche vorgestellt werden, die für alle Wörter des Vokabulars eine Bewertung berechnet und das Wort mit der höchsten Bewertung als Ausgabe wählt. Wie der Suchraum und damit der Suchaufwand erheblich gegenüber einer solchen vollständigen Suche eingeschränkt werden kann, ist Inhalt des anschließenden Abschnittes.

Bei der freien Suche mit anschließender Korrektur der Worthypothese wurde mit Hilfe der Editierdistanz eine Auswahl aus dem Vokabular getroffen, die anhand der Aktivierungen der Zustandsschicht bewertet wurde. Ist das gesuchte korrekte Wort nicht in der Liste der Wörter mit minimaler Editierdistanz, kann es auf diese Weise nicht gefunden werden, obwohl es sich im Vokabular befindet. Unter Umständen hat das korrekte Wort eine weitaus höhere Editierdistanz, könnte aber dennoch durch eine Bewertung anhand der Aktivierungen der Zustandsschicht und durch Vergleich mit

den Bewertungen anderer Wörter des Vokabulars identifiziert werden. Wird nun auf die Vorauswahl von Wörtern mit der freien Suche und Editierdistanzen verzichtet, und werden alle Wörter des Vokabulars bewertet, führt dies zu der flachen, vollständigen Suche.

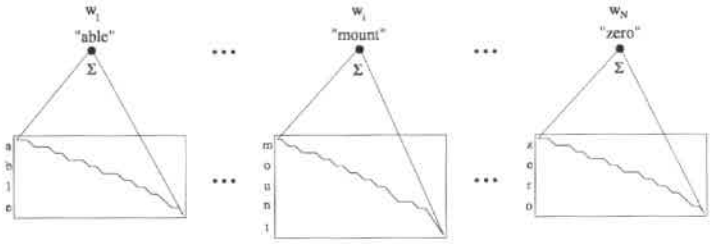


Abbildung 7.10: Flache Suche durch alle Wortmodelle des Vokabulars

Für alle Wörter w_i des Vokabulars W werden die Wortmodelle durch Konkatenation der jeweiligen Modelle der einzelnen Zeichen gebildet und mit dem Viterbi-Algorithmus ein bester Pfad durch die entsprechende Matrix der Aktivierungen berechnet, wie schematisch in Abbildung 7.10 zu sehen ist. Aus den N so bewerteten Wörtern des Vokabulars wird abschließend dasjenige mit der höchsten Bewertung als Ausgabehypothese ausgewählt. Tabelle 7.6 zeigt die Buchstabenakkuratheit und Worterkennungsrate, die mit dieser erschöpfenden Suche auf den drei Datenbasen erzielt werden. Da bei dieser Art der Suche alle Wörter betrachtet werden, ist sie die optimale Suche in Bezug auf die Erkennungsrate.

	UKA		CMU		MIT	
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED	Gesamt
BA	99.0%	95.4%	98.2%	96.8%	97.3%	**97.0%**
WA	97.1%	90.8%	92.8%	92.2%	94.7%	**93.4%**

Tabelle 7.6: Wortakkuratheit und Buchstabenakkuratheit mit flacher, vollständiger Suche

Obwohl sie die optimale Erkennungsrate für die Worterkennung garantiert, ist die flache Suche ebenso wie die freie Suche mit Editierdistanzen nur eingeschränkt, d.h. bei Verwendung kleiner Vokabulare, für den praktischen Einsatz geeignet, da die Berechnungszeit linear mit der Größe des Vokabulars steigt. Während bei der freien Suche mit Editierdistanzen die Berechnung der Distanzen eines Wortes zu allen Wörtern des Vokabulars den Hauptberechnungsaufwand ausmacht, ist es hier das Berechnen der Pfade für alle Wörter. Eine genauere Betrachtung aller durchlaufenen Wortmodelle und des bisherigen Berechnungsschemas zeigt, daß für gemeinsame Wortanfänge die Modelle dieser Wortanfänge unnötigerweise für jedes Wort mit diesem Wortanfang separat durchlaufen werden. Wie diese Mehrfachberechnungen mit einer Repräsen-

tation des Vokabulars durch eine Baumstruktur vermieden werden kann, zeigt der folgende Abschnitt.

7.9 Baumsuche

Die flache Suche mit ihrer Repräsentation der Wörter des Vokabulars als voneinander unabhängigen linearen Listen erfordert bei einer Vokabulargröße von N Wörtern und einer durchschnittlichen Wortlänge von a Buchstaben die Berechnung von insgesamt $N \cdot a$ Buchstabenmodellen, um alle Bewertungen zu berechnen. Mit $N = 20\,000$ und $a = 8.2$ für das Vokabular WSJ_20K sind dies bereits 164 000 Buchstabenmodelle. Selbst auf schnellen Rechnern ergibt dies Erkennungszeiten von 10 Sekunden und mehr. Die Anzahl der erforderlichen Buchstabenmodelle kann erheblich reduziert werden, wenn jeweils identische Wortpräfixe nur noch durch eine einzige Kopie der Buchstabenmodelle repräsentiert werden, das Vokabular also wie in Abbildung 7.11 als Baum angeordnet wird [MFW96]. Die ursprünglichen 164 000 Buchstabenmodelle des Vokabulars WSJ_20K reduzieren sich dadurch auf nur noch etwas mehr als 55 000, d.h. etwa ein Drittel der ursprünglichen Zahl.

In der Baumstruktur können drei Arten von Knoten unterschieden werden:

- Die *Wurzelknoten* bilden die Wurzeln der einzelnen Teilbäume. Für jedes Zeichen, mit dem ein Wort des Vokabulars beginnen kann, existiert ein solcher Wurzelknoten mit entsprechendem Teilbaum. In diesen Knoten wird der Suchprozeß gestartet.

- *Wortendeknoten* markieren das jeweils letzte Buchstabenmodell eines Wortes. Insbesondere sind alle Blätter des Baumes Wortendeknoten. Sind vollständige Wörter des Vokabulars auch Präfix eines anderen Wortes, sind interne Knoten des Baumes auch zugleich Wortendeknoten.

- *Interne Knoten* sind alle Knoten des Baumes, in denen die Suche weder begonnen noch beendet werden kann, außer der interne Knoten ist auch zugleich Wortendeknoten.

Der Baum der Buchstabenmodelle wird zeit-synchron in einer Breitensuche durchlaufen, ausgehend von allen Wurzelknoten gleichzeitig. Zunächst werden alle Pfade durch den Baum weiterverfolgt, später wird gezeigt, wie durch Einführung einer Strahlensuche der Suchaufwand erheblich verringert werden kann. Die Breitensuche ist beendet, wenn wie üblich der letzte Aktivierungsvektor der Zustandsschicht erreicht wurde. Es müssen abschließend nur noch alle Wortendeknoten nach der maximalen Bewertung durchsucht werden. Das zu diesem Wortendeknoten gehörige Wort ist dann die Ausgabehypothese der Erkennung. Mit dieser Baumsuche werden dieselben Erkennungsraten wie mit der flachen Suche erzielt, wobei jedoch weitaus weniger Buchstabenmodelle durchlaufen werden müssen.

Durch die Reduzierung der Buchstabenmodelle wird nur eine unwesentliche Verbesserung der Erkennungszeit erzielt, da im Vergleich zur flachen Suche ein deutlich

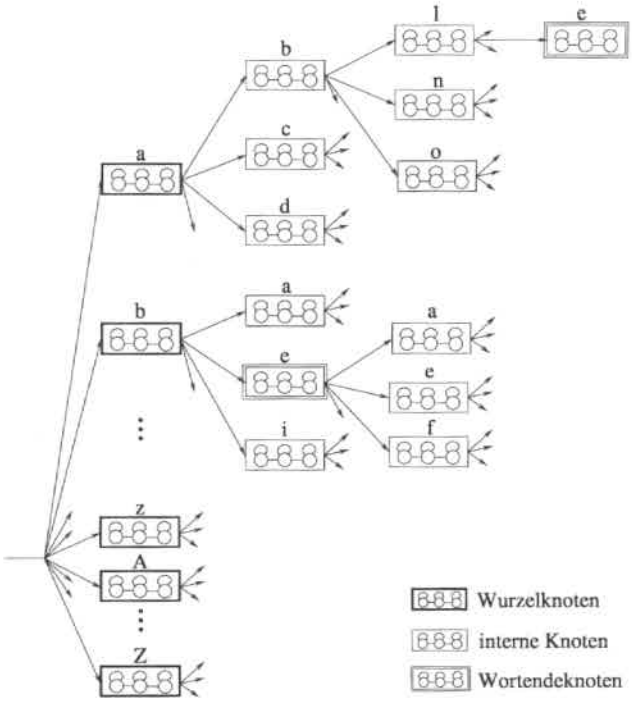


Abbildung 7.11: Das Vokabular als Baumstruktur

höherer Verwaltungsaufwand erforderlich ist (siehe dazu auch Abschnitt 7.9.1). Eine Möglichkeit ist daher, nicht mehr alle Buchstabenmodelle zu berücksichtigen, sondern eine Strahlensuche (beam search) einzusetzen, die im Laufe der Suche zu jedem Zeitpunkt t nur noch die besten Pfade des Baumes weiterverfolgt. Zu diesem Zweck werden im folgenden Knoten des Baumes als „aktiv" oder „inaktiv" definiert. Aktiv bedeutet, daß der Knoten in den Suchprozeß einbezogen wird, und inaktiv, daß er nicht berücksichtigt wird. Ziel ist es, die Suche jeweils nur noch an denjenigen Knoten fortzusetzen, deren Bewertung bis zu dem Zeitpunkt nicht zu weit hinter der Bewertung des bis dahin besten Knotens zurückgefallen ist. Ein Knoten k bleibt daher nur aktiv, wenn seine aktuelle Bewertung $s(k)$ innerhalb des Bereiches $s(k^*) - b$ liegt, d.h. die Differenz der aktuellen Bewertung $s(k)$ und der Bewertung des besten Knotens $s(k^*)$ die Strahlbreite b (beam) nicht überschreitet. Es ergibt sich der in Abbildung 7.12 dargestellte Suchablauf mit folgenden Verarbeitungsschritten:

1. **Initialisierung.** Im ersten Schritt der Suche werden die Wurzelknoten auf aktiv und alle übrigen Knoten des Baumes auf inaktiv gesetzt. Die ersten Zustände der Knoten werden mit den Bewertungen aus dem erstem Ausgabevektor der

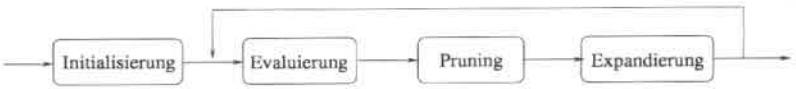


Abbildung 7.12: Der Suchalgorithmus

Zustandsschicht initialisiert.

2. **Evaluierung.** Für alle aktiven Knoten $k_1, \ldots, k_N$ wird ein Schritt des Viterbi-Algorithmus ausgeführt, um deren Bewertung für den nächsten Aktivierungsvektor der Zustandsschicht zu berechnen. Anschließend wird die bis dahin beste Bewertung $\hat{s}$ aller Knoten ermittelt:

$$\hat{s} = \max_{i=1\ldots N} s(k_i)$$

3. **Pruning.** Alle aktiven Knoten k_i mit

$$s(k_i) < \hat{s} - b$$

werden auf inaktiv gesetzt, d.h. bei der nächsten Iteration durch diese Schritte nicht mehr berücksichtigt.

4. **Expandierung.** Für alle aktiven Knoten k_i wird überprüft, ob ein Übergang von dem letzten Zustand des Knotens in den ersten Zustand eines Nachfolgeknotens $\text{child}(k_i)$ eine höhere Bewertung ergibt. Ist dies der Fall und die neue Bewertung $s(\text{child}(k_i))$ zusätzlich über dem Schwellwert $\hat{s} - b$, d.h. $s(\text{child}(k_i)) \geq \hat{s} - b$, wird dieser Nachfolgeknoten für die nächste Iteration als aktiv markiert. Ist das Ende der Zustandsschicht noch nicht erreicht, wird bei Schritt 2 fortgefahren.

Auch bei diesem Suchverfahren werden zwar weiterhin mehrere Pfade parallel verfolgt, jedoch in Abhängigkeit der Strahlbreite nur eine geringe Anzahl im Vergleich zur vollständigen Suche. Für die maximale Strahlbreite ist das Ergebnis dieser Suche identisch mit dem der flachen Suche, da alle Pfade des Baumes durchlaufen werden und somit auf jeden Fall auch der Pfad des gesuchten Wortes bewertet wird.

7.9.1 Einfluß der Strahlbreite auf Erkennungszeit und -rate

Die Wahl der Breite b des Suchstrahls ist kritisch in Bezug sowohl auf die Erkennungszeit als auch die Erkennungsrate. Ein schmalerer Strahl bedeutet eine größere Einschränkung des Suchraumes und damit eine Verbesserung der Antwortzeit des Systems. Jedoch befindet sich die gesuchte Hypothese bei kleinerem b häufiger in einem inaktiven Bereich des Suchbaumes, der aufgrund eines zwischenzeitlich höher bewerteten Pfades in einem anderen Bereich nicht weiter verfolgt wird, obwohl sich am Ende der Suche absolut die höchste Bewertung für die gesuchte Hypothese ergeben hätte. Die Folge eines zu kleinen Suchstrahls ist demnach ein Absinken der Erkennungsrate.

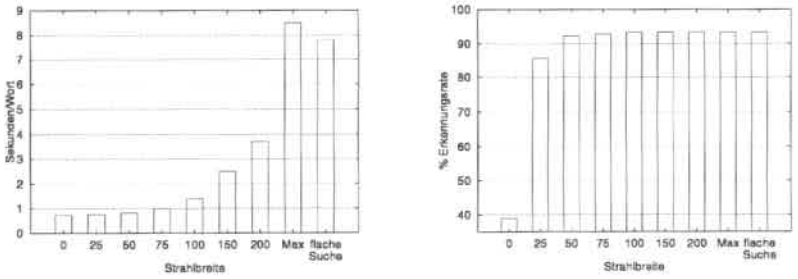


Abbildung 7.13: Die Erkennungszeit und -rate in Abhängigkeit der Strahlbreite für Vokabular WSJ_20K

Ein breiter Suchstrahl garantiert zwar die bestmögliche Erkennungsrate, führt aber zu langen Antwortzeiten, die den praktischen Nutzen des Systems einschränken. Die „optimale" Einstellung von b hängt von der jeweiligen Anwendung des Systems ab. Für die Bestimmung der maximalen Erkennungsleistung des Systems durch Evaluierung anhand einer Datenbasis, bei der lange Antwortzeiten nicht von Bedeutung sind, wird man b so groß wie möglich wählen, um Fehlerkennungen durch Beschneidung des Suchraumes zu vermeiden. Für den praktischen Einsatz hingegen wird b als ein möglichst guter Kompromiß zwischen der gewünschten Antwortzeit und Erkennungsrate gewählt.

Abbildung 7.13 stellt für verschiedene Strahlbreiten die durchschnittlich benötigte Erkennungszeit pro Wort der Gesamterkennungsrate mit dem Vokabular WSJ_20K gegenüber. Die in Sekunden angegebene Erkennungszeit pro Wort beinhaltet für alle Strahlbreiten jeweils 0.16 Sekunden für die Vorverarbeitung und 0.52 Sekunden für die Berechnung der Ausgabe des MS-TDNN. Die Messungen wurden auf einem herkömmlichen PC mit 200Mhz Pentium MMX Prozessor und 64MB SDRAM durchgeführt. Die angegebenen Zeiten sind der Durchschnitt über alle getesteten Wörter der drei Datenbasen. Zusätzlich sind in der Abbildung zum Vergleich mit der Baumsuche auch die entsprechenden Werte der flachen Suche eingetragen.

Wie bereits in Abschnitt 7.9 erwähnt, ist aufgrund des erhöhten Verwaltungsaufwands bei der Baumsuche ohne Beschränkung des Suchraumes die Erkennungszeit geringfügig größer als die der flachen Suche. Doch bereits für $b = 200$ halbiert sich die Erkennungszeit gegenüber der flachen Suche, ohne daß eine Verringerung der Erkennungsrate auftritt. Erst für $b < 100$ ist bei weiter deutlich sinkenden Antwortzeiten eine leichte Verschlechterung der Erkennungsrate zu beobachten. Für $b = 0$ wird ebenso wie bei der freien Suche jeweils nur ein einziger Pfad durch den Suchbaum verfolgt. Da dieser Pfad jedoch nur entlang eines legalen Wortes des Vokabulars verläuft, liegt die Erkennungsrate in diesem Fall mit 39% über der der freien Suche mit 27%. Die Experimente dieser Arbeit wurden mit einer Strahlbreite von $b = 150$ durchgeführt. Für interaktive Demonstrationen des Erkennungssystems bilden $b = 75$ oder $b = 50$

einen guten Kompromiß zwischen Geschwindigkeit und Erkennungsrate.

7.9.2 Einfluß der Vokabulargröße auf die Erkennungsrate

Die bisherigen Experimente in diesem Kapitel wurden mit dem 20 000 Wörter umfassenden Vokabular WSJ_20K durchgeführt. In realen Anwendungen treten jedoch unterschiedliche Anforderungen an die Vokabulargröße auf. Für spezielle, eingeschränkte Anwendungen kann das Vokabular wesentlich kleiner gewählt werden. Eine allgemeine Benutzerschnittstelle für die Eingabe beliebigen Textes hingegen erfordert unter Umständen deutlich größere Vokabulare. Mit der Größe des Vokabulars sinkt oder steigt auch die Verwechselbarkeit der Wörter, so daß sich für jede Größe eine spezifische Erkennungsrate ergibt. Wie sich die Erkennungsrate mit der Vokabulargröße ändert, zeigt Abbildung 7.14 am Beispiel der 4 Vokabulare WSJ_5K, WSJ_10K, WSJ_20K und WSJ_50K sowohl anhand der Gesamtworterkennungsrate als auch der Erkennungsraten für die einzelnen Testmengen der Datenbasen.

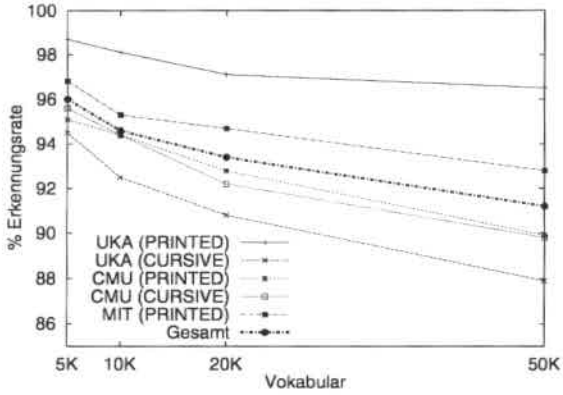


Abbildung 7.14: Die Erkennungsrate in Abhängigkeit von der Vokabulargröße

Die Gesamterkennungsrate steigt bis auf 96.0% im Falle des kleinsten gewählten Vokabulars mit 5 000 Wörtern und fällt auf 91.2% beim größten Vokabular mit 50 000 Wörtern. Die Verzehnfachung der Vokabulargröße zieht also in diesem Fall etwa eine Verdopplung der Fehlerrate nach sich. Selbst bei einer weiteren Verdoppelung der Vokabulargröße auf 100 000 Wörter ist aufgrund des flacher werdenden Kurvenverlaufs eine Erkennungsrate im Bereich von etwa 90% zu erwarten.

7.10 Beiträge dieser Arbeit

In diesem Kapitel wurden erstmals unterschiedliche Methoden der Integration eines Vokabulars in die Erkennung handschriftlicher Einzelwörter untersucht und anhand eines einzigen neuronalen Erkenners und einer einheitlichen Datenbasis evaluiert. Dabei wurde gezeigt, daß mit einer Baumsuche, die in der Handschrifterkennung bislang nur in HMM-basierten Systemen eingesetzt wurde, auch in Verbindung mit einem neuronalen Ansatz gegenüber anderen Suchansätzen die besten Ergebnisse sowohl in Bezug auf die Erkennungsrate als auch benötigte Erkennungszeit erzielt werden.

Bereits durch die Modellierung der Zeichen durch eine Folge von Zuständen und eine einfache Variante der nachgeschalteten Suche mit Editierdistanzen wird mit diesem System die Leistung vergleichbarer neuronaler Systeme um mehr als 7% übertroffen [Sen95, Sch95] und mit 87.2% auch etwa das beste vergleichbare Ergebnisse eines HMM-basierten Erkenners erreicht [DHU97]. Durch die direkte Integration des Vokabulars in den Suchprozeß wird diese Erkennungsleistung noch um etwa 6% auf 93.4% für ein 20 000 Wörter umfassendes Vokabular gesteigert.

Die Experimente in diesem Kapitel belegen, daß durch den effektiven Einsatz von Pruning-Methoden bei der Baumsuche der Suchraum stark eingeschränkt werden kann und so eine Erkennung der Einzelworte in Echtzeit möglich ist. Durch den modularen Aufbau und die Modellierung von Wörtern als Folge einzelner Zeichen ist in diesem System ein einfacher Austausch des Vokabulars ohne ein Neutraining des neuronalen Netzes oder sonstige Anpassungen möglich. Selbst zur Laufzeit des Systems können durch dynamische Erweiterung der Baumstruktur neue Wörter zum Vokabular hinzugefügt werden.

Kapitel 8

Die Suche für Satzerkennung

8.1 Einleitung

Die vorhergehenden Kapitel befaßten sich mit der schrittweisen Entwicklung eines Einzelzeichen- und Worterkenners, ausgehend von einer einfachen TDNN Architektur bis hin zum MS-TDNN mit einer effizienten Baumrepräsentation des Vokabulars. Dieses Kapitel stellt abschließend die erforderlichen Erweiterungen für die Satzerkennung vor. Dabei kommen bereits für die Erweiterung von einem Einzelzeichen- zum Worterkenner vorgestellte Techniken zum Einsatz. Anstelle eines einzigen Wortes besteht die Eingabe nun aus einem Satz bzw. allgemein aus einer Folge von Wörtern. Es kommt also ein weiteres Segmentierungsproblem hinzu, da die Wortgrenzen ebensowenig bekannt sind wie die Buchstabengrenzen innerhalb eines Wortes.

Bisher existierende neuronale Ansätze für die Handschrifterkennung sind durch die nachgeschaltete Suche auf Einzelworterkennung beschränkt [Sch95, Sen95], sofern nicht die Eingabe explizit in einzelne Worte segmentiert wird. Bei einer expliziten Segmentierung auf Wortebene treten jedoch ähnliche Probleme wie bei der Segmentierung von Worten auf Buchstabenebene auf, so daß durch falsche Segmentierung Fehlerkennungen erzeugt werden. In der Spracherkennung, wie auch in den HMM-basierten Systemen von Starner et al. [SMSC94] und Ratzlaff et al. [RNM96] für die Handschrifterkennung, wird dieses Problem durch direkte Einbindung der Segmentierung in die Suche gelöst. Wie auch in diesem neuronalen System das Segmentierungsproblem mit dem Prinzip der integrierten Erkennung und Segmentierung gelöst werden kann, ist Inhalt dieses Kapitels.

Der nächste Abschnitt befaßt sich zunächst mit den notwendigen Erweiterungen der MS-TDNN Architektur und dem Training auf Satzebene. Daran schließt sich die Darstellung der Baumsuche für die Satzerkennung an. Abschließend wird gezeigt, wie in diese Baumsuche Sprachmodelle integriert werden können und welche Verbesserungen sich daraus ergeben.

8.2 Die MS-TDNN Architektur für die Satzerkennung

Da in dieser Arbeit auch bei der Satzerkennung nur Klein- und Großbuchstaben berücksichtigt werden, muß die MS-TDNN Architektur der Worterkennung lediglich um ein einziges zusätzliches Zeichen, das „Leerzeichen", ergänzt werden. Das Leerzeichen repräsentiert den Übergang von dem Ende eines Wortes zum Beginn des neuen Wortes. Im Gegensatz zu den Klein- und Großbuchstaben wird das Leerzeichen hier nur durch einen einzigen Zustand (SPACE) modelliert, da die Trajektorie für das Leerzeichen zum einen oftmals sehr kurz ist und zum anderen durch die lineare Interpolation der Bereiche, an denen der Stift abgehoben wurde, wenig Änderung im zeitlichen Verlauf aufweist. Um von einem Wort in das nächste Wort übergehen zu können, muß also jeweils das Modell für das Leerzeichen durchlaufen werden. Abbildung 8.1

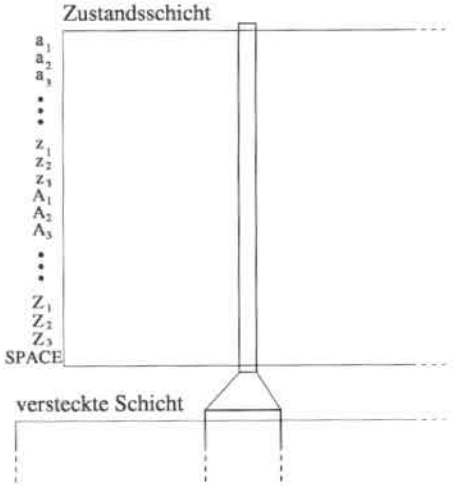


Abbildung 8.1: Die Zustandsschicht des MS-TDNN für die Satzerkennung

zeigt die sich aus dieser Erweiterung ergebende Architektur der Zustandsschicht des im folgenden verwendeten MS-TDNN. Die Eingabeschicht wird ebenso wie die Zustandsschicht um ein zusätzliches Neuron erweitert, da die Merkmalsvektoren für die Satzerkennung um ein Merkmal ergänzt werden, wie bereits in Kapitel 5.4.3 beschrieben wurde. Dieses zusätzliche Merkmal kodiert – als Gegenstück des Zustands für das Leerzeichen in der Zustandsschicht – in der Eingaberepräsentation eines Satzes für jeden Punkt die heuristisch ermittelte Wahrscheinlichkeit, daß in diesem Punkt ein neues Wort beginnt. Diese zusätzliche Information soll das im folgenden Abschnitt beschriebene Trainieren des Zustands SPACE erleichtern. Alle weiteren Netzwerkparameter, d.h. die Anzahl der Neuronen in der verborgenen Schicht und die Breite

der Kontexte in der Eingabe- und der verborgenen Schicht, bleiben gegenüber der Worterkennung unverändert.

8.3 Training des MS-TDNN für die Satzerkennung

Mit der Einführung eines zusätzlichen Trainingsschrittes auf Satzebene in den Trainingsablauf, wie Abbildung 8.2 zeigt, werden prinzipiell dieselben zwei Ziele verfolgt wie bei der Einführung des Trainings auf Wortebene. Zum einen erschließen sich dadurch mit den in den Datenbasen enthaltenen Sätzen, für die weder die Buchstaben- noch Wortgrenzen bekannt sind, zusätzliche Trainingsdaten. Zum anderen kann nur durch das Training auf ganzen Sätzen oder Wortfolgen das zusätzlich in die MS-TDNN Architektur aufgenommene Modell des Leerzeichens trainiert werden. Das Training auf Zustands-, Zeichen- und Wortebene wird analog dem Trainingsablauf für die Worterkennung mit identischen Daten durchgeführt. Nach einigen Iterationen des Trainings auf Wortebene werden dann auch die Sätze der drei Datenbasen zum Training hinzugenommen. Die Sollwerte für diese Sätze werden jeweils wiederum mit dem Viterbi-Algorithmus bestimmt, diesmal jedoch durch das Modell des gesamten Satzes, wobei zwischen den einzelnen Wortmodellen das Modell für das Leerzeichen eingefügt wurde. Das Training auf Wortebene für unsegmentierte und auf Zeichenebene für segmentierte Einzelworte wird auch in dieser Trainingsphase beibehalten.

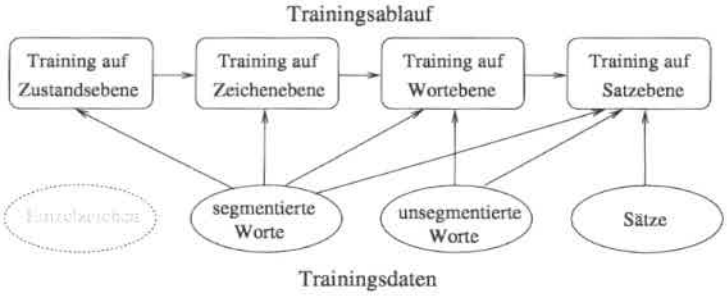


Abbildung 8.2: Trainingsschritte für die Satzerkennung

8.4 Erweiterung der Baumsuche für die Satzerkennung

Der Schritt von der Worterkennung zur Erkennung von Wortfolgen erfordert zwei Erweiterungen der in Kapitel 7.9 vorgestellten Baumrepräsentation des Vokabulars, die in Abbildung 8.3 dargestellt sind. Zusätzlich zu den bereits vorhandenen Übergängen

von einem Knoten des Baumes in die unmittelbar folgenden Knoten, also dem Wechsel von einem Zeichen eines Wortes in das nachfolgende Zeichen, sind nun auch Übergänge von allen Wortendeknoten in das Modell des Leerzeichens möglich. Von dem Modell des Leerzeichens existieren wiederum Übergänge zu allen Wurzelknoten des Baumes, so daß ein neues Wort begonnen werden kann. Dieses Modell bildet sozusagen nach einem ersten Durchlauf des Baumes ausgehend von den ursprünglichen Wurzelknoten den neuen und einzigen Wurzelknoten des Baumes, von dem aus jedes Wort des Vokabulars erreicht wird.

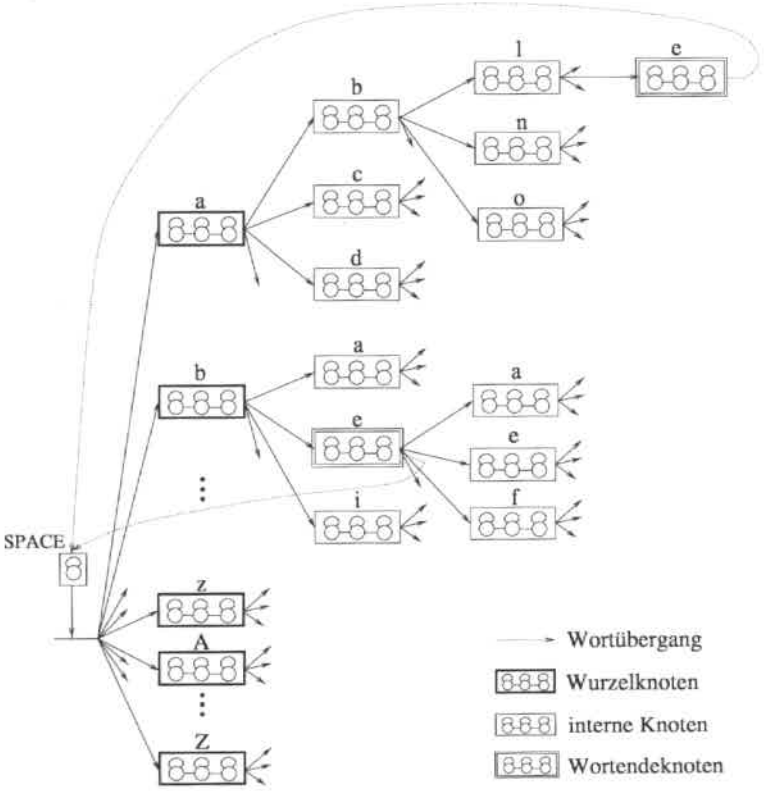


Abbildung 8.3: Das Vokabular als Baumstruktur mit Wortübergängen und dem Modell für Leerzeichen

Die Wortübergänge machen nicht nur obige Änderung in der Baumarchitektur nötig, sondern erfordern damit verbunden gegenüber der Worterkennung auch den zusätzlichen Schritt **Wortübergang** in dem Suchalgorithmus, der nun vollständig in Abbildung 8.4 schematisch dargestellt ist. Die Schritte **Initialisierung**, **Evalu-**

ierung, **Pruning** und **Expandierung** werden analog dem Algorithmus der Einzelwortsuche in Kapitel 7.9 durchgeführt. Nachdem für alle aktiven Knoten k_i die Übergänge von dem letzten Zustand des Knotens in den ersten Zustand der Nachfolgeknoten child(k_i) bewertet wurden, wird nun zusätzlich für alle aktiven Wortendeknoten der Übergang von dem letzten Zustand in den Knoten des Leerzeichens k_{SPACE} bewertet. Ergibt sich durch den Übergang von einem der aktiven Wortendeknoten in k_{SPACE} eine höhere Bewertung $s(k_{\text{SPACE}})$ und ist $s(k_{\text{SPACE}}) \geq \hat{s} - b$, wird k_{SPACE} für die nächste Iteration des Suchalgorithmus auf aktiv gesetzt. Um am Ende der Suche die gefundene Wortfolge rekonstruieren zu können, werden für jede Iteration alle Wortübergänge festgehalten.

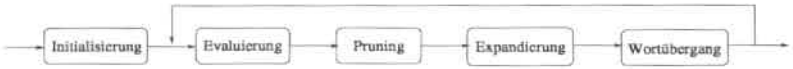


Abbildung 8.4: Der Suchalgorithmus für die Satzerkennung

	WSJ_5K	WSJ_10K	WSJ_20K	WSJ_50K
BA	93.8%	92.7%	91.9%	89.8%
WA	85.7%	83.9%	81.7%	75.8%

Tabelle 8.1: Die Wort- (WA) und Buchstabenakkuratheit (BA) der Satzerkennung ohne Sprachmodelle für verschiedene Vokabulargrößen auf der Datenbasis CMU

Die Wort- und Buchstabenakkuratheit auf den Testsätzen der Datenbasis CMU für unterschiedliche Vokabulargrößen sind in Tabelle 8.1 angegeben. Der Großteil der Fehler wird durch Verwechslungen einzelner Wörter verursacht. Einfügungen und Auslassungen von Wörtern treten im Vergleich mit Verwechslungen nur selten auf. Im Fall des Vokabulars WSJ_20K stehen 376 Verwechslungen nur 28 Einfügungen und 20 Auslassungen von Wörtern gegenüber. Die Wortgrenzen werden also bei diesem Ansatz offensichtlich bereits sehr zuverlässig identifiziert, wozu insbesondere die Einführung des Modells für das Leerzeichen beiträgt. Ein Vergleich mit einer Baumrepräsentation ohne Modellierung des Leerzeichens zeigt, daß die Erkennungsrate für das Vokabular WSJ_20K auf ca. 75% sinkt, wenn Wortübergänge direkt in die Wurzelknoten des Baumes durchgeführt werden, wobei in diesem Fall deutlich häufiger Einfügungen und Auslassungen auftreten. Eine detailliertere Analyse der bei der Satzerkennung auftretenden Fehler ist in Kapitel 9.3.3 zu finden.

8.5 Sprachmodelle bei der Satzerkennung

Bereits bei der Worterkennung mit freier Suche konnte die Erkennungsleistung durch den Einsatz von Sprachmodellen auf der Buchstabenebene verbessert werden, da Sprachmodelle die Einbeziehung syntaktischen Wissens über die Sprache ermöglichen.

Bei der Worterkennung war dies die statistische Häufigkeit bestimmter Buchstabenfolgen, die auf einem großen Vokabular berechnet wurden. So wurde die Entscheidung für den Übergang von einem Zeichenmodell in ein anderes nicht nur anhand der Bewertungen der Zustandsschicht des MS-TDNN getroffen, sondern zusätzlich auf Basis der Wahrscheinlichkeit für eine Buchstabenfolge. Bei der Satzerkennung tritt nun dasselbe grundsätzliche Problem zutage, jedoch nicht auf der Ebene von einzelnen Zeichen, sondern auf der Ebene von Wörtern. In obigem Suchalgorithmus wird die Entscheidung für einen Wortübergang wiederum nur aufgrund der Zustandsbewertungen des MS-TDNN getroffen. Syntaktisches Wissen über die Wahrscheinlichkeit bestimmter Wortfolgen in der Sprache wurde bislang nicht berücksichtigt, um Wortübergänge zu bewerten. Dies soll nun durch die Integration eines Bigramms nachgeholt werden.

Ein Problem bei der Integration von Bigrammen in die oben vorgestellte Baumstruktur ergibt sich dadurch, daß die Identität eines möglichen Wortnachfolgers erst in einem Wortendeknoten und nicht bereits in den Wurzelknoten bekannt ist, da die Wurzelknoten den Beginn einer großen Menge unterschiedlicher Wörter repräsentieren. Daher werden hier, wie in Suchansätzen der kontinuierlichen Spracherkennung üblich, sogenannte verzögerte Bigramme (Delayed Bigrams) verwendet [ONA97]. Die Wahrscheinlichkeit $P(w_2|w_1)$ des Nachfolgewortes w_2 eines Wortes w_1 wird bei verzögerten Bigrammen nicht beim Beginn eines neuen Wortes w_2 berechnet, sondern erst, wenn dessen zugehöriger Wortendeknoten, d.h. der letzte Buchstabe des Wortes, während der Suche erreicht wird. Dort wird der Eintrittszeitpunkt t ermittelt, an welchem w_2 in dem Wurzelknoten begonnen wurde und anhand des Sprachmodells aus allen Worten, die zu dem Zeitpunkt t beendet wurden, der wahrscheinlichste Vorgänger w_1 des Wortes ermittelt. Die so ermittelte Wahrscheinlichkeit $P(w_2|w_1)$ wird abschließend bei der Berechnung der Bewertung des Wortendeknotens von w_2 berücksichtigt.

Größter Nachteil verzögerter Bigramme ist, daß die eigentliche Segmentierungsentscheidung, d.h. die Bestimmung des Eintrittszeitpunktes eines neuen Wortes, ohne Information des Sprachmodells stattfinden muß. Wird das neue Wort an einem zu frühen oder späten Zeitpunkt begonnen, ist unter Umständen der tatsächlich gesuchte Vorgänger des Wortes noch nicht oder nicht mehr in der Liste aller möglichen Vorgänger für dieses Wort. Vorteil verzögerter Bigramme ist jedoch, daß viele Worthypothesen bereits vor Erreichen des Wortendeknotens aufgrund der begrenzten Breite des Suchstrahls nicht mehr aktiv sind und daher die zeitintensive Suche nach dem möglichen Vorgänger für diese Worte entfällt.

	WSJ_5K	WSJ_10K	WSJ_20K	WSJ_50K
BA	95.3%	94.8%	93.7%	91.3%
WA	89.7%	88.4%	86.6%	82.3%

Tabelle 8.2: Die Wort- (WA) und Buchstabenakkuratheit (BA) der Satzerkennung mit Bigramm für verschiedene Vokabulargrößen auf der Datenbasis CMU

Tabelle 8.2 zeigt die durch den Einsatz des Bigramms erreichte Wortakkuratheit auf den Testsätzen der Datenbasis CMU. Im Vergleich zur Wortakkuratheit des Systems ohne Bigramm ist hier eine Reduzierung der Fehlerrate um etwa 25% zu beobachten. Eine detailliertere Analyse der Fehler, die durch den Einsatz eines Sprachmodells vermieden werden, ist in Kapitel 9.3.3 zu finden.

8.6 Beiträge dieser Arbeit

In diesem Kapitel wurde gezeigt, daß im Gegensatz zu vergleichbaren Ansätzen mit nachgeschalteten Vokabularen in diesem neuronalen System die Segmentierung analog der Segmentierung auf Buchstabenebene direkt in die Suche integriert werden kann. Dies wurde bislang in der On-line Handschrifterkennung nur in HMM-basierten Systemen realisiert. Damit ist dieses System insbesondere das erste auf neuronalen Netzen basierende System für die On-line Handschrifterkennung, das durch den modularen Aufbau neben Einzelzeichen und Wörtern auch Wortfolgen verarbeiten kann.

Durch die explizite Modellierung des Leerzeichens, das die Aufgabe der Segmentierung unterstützt, wird bereits ohne den Einsatz von Sprachmodellen eine schreiberunabhängige Erkennungsleistung von über 80% erzielt. Mit einem Bigramm kann diese Erkennungsrate weiter auf über 86% für ein 20 000 Wörter umfassendes Vokabular gesteigert werden. Das bislang beste vergleichbare schreiberunabhängige System von Ratzlaff et al. [RNM96] wird damit um mehr als 23% übertroffen. Das System von Starner et al. [SMSC94] ist aufgrund seiner Schreiberabhängigkeit und den nicht realistischen Daten, auf denen das System getestet wurde, nicht mit diesem System vergleichbar.

Kapitel 9

Weitere Ergebnisse und Fehleranalysen

Die vorhergehenden Kapitel befaßten sich mit der Vorverarbeitung, dem neuronalen Netzwerk zur Berechnung der Buchstabenhypothesen und der vokabulargesteuerten Suche für die Wort- und Satzerkennung. An vielen Stellen wurden bereits Ergebnisse für die einzelnen Erkennungsaufgaben präsentiert, um bestimmte Entwurfsentscheidungen zu motivieren. Fragen, die sich aus den unterschiedlichen Charakteristiken der getesteten Daten ergeben, blieben bislang unbeantwortet. Ebenso wurde bislang nicht auf die Frage eingegangen, welcher Art die Fehler sind, die das System produziert. Die Beantwortung dieser Fragen steht im Mittelpunkt dieses Kapitels. Zunächst werden die bislang präsentierten Ergebnisse noch einmal zusammengefaßt und mit den Ergebnissen anderer Systeme verglichen, sofern dies möglich ist. Daran schließen sich weitere Experimente zum Verhalten des Systems auf verschiedenen Daten und unter bestimmten Bedingungen an. Eine Analyse der Fehlerkennungen für die Einzelzeichen-, Wort- und Satzerkennung beendet dieses Kapitel.

9.1 Zusammenfassung der bisherigen Ergebnisse

In Tabelle 9.1 sind die wichtigsten in dieser Arbeit erreichten Erkennungsergebnisse für die drei Erkennungsaufgaben noch einmal in der Übersicht dargestellt. Wie bereits früher erwähnt, sind dies nicht unbedingt für alle Erkennungsaufgaben und Datenbasen die bestmöglichen Ergebnisse, die unter speziellen Bedingungen erzielt werden können, sondern die Ergebnisse, die sich aus einer einheitlichen Modellierung und Erkennerarchitektur für alle durchgeführten Experimente ergeben. Dies vermittelt ein einheitlicheres Gesamtbild der Erkennungsleistung.

Wie bereits erwähnt wurde, ist ein Vergleich mit anderen Systemen schwierig, da für die On-line Handschrifterkennung keine standardisierten Testmengen existieren, die einen Test unter gleichen Bedingungen ermöglichen. Die bislang besten schreiberunabhängigen Ergebnisse auf diesem Forschungsgebiet wurden von Seni, Schenkel, Dolfing et al. und Ratzlaff et al. publiziert. Diese Ergebnisse wurden zumindest unter

Einzelzeichen (BE)	UKA	CMU	MIT	Gesamt
Ziffern	97.9%	97.0%	95.3%	**96.5%**
Großbuchstaben	93.4%	90.5%	94.9%	**92.7%**
Kleinbuchstaben	89.7%	89.8%	93.4%	**91.1%**
Wörter (WA)	UKA	CMU	MIT	Gesamt
	PRI / CUR	PRI / CUR	PRI	
ohne Vokabular	37.3%/14.2%	24.7%/17.8%	36.0%	**26.8%**
mit Bigramm	44.6%/18.2%	30.2%/24.3%	45.0%	**33.8%**
mit Trigramm	45.5%/18.6%	31.9%/25.0%	46.3%	**34.9%**
mit 5K Vokabular	98.7%/94.5%	95.1%/95.6%	96.8%	**96.0%**
mit 10K Vokabular	98.1%/92.5%	94.3%/94.4%	95.3%	**94.6%**
mit 20K Vokabular	97.1%/90.8%	92.8%/92.2%	94.7%	**93.4%**
mit 50K Vokabular	96.5%/87.9%	89.9%/89.8%	92.8%	**91.2%**
Sätze (WA) (ohne Sprachmodell)	UKA	CMU	MIT	Gesamt
mit 5K Vokabular	-	85.7%	-	**85.7%**
mit 10K Vokabular	-	83.9%	-	**83.9%**
mit 20K Vokabular	-	81.7%	-	**81.7%**
mit 50K Vokabular	-	75.8%	-	**75.8%**
Sätze (WA) (Bigramm)	UKA	CMU	MIT	Gesamt
mit 5K Vokabular	-	89.7%	-	**89.7%**
mit 10K Vokabular	-	88.4%	-	**88.4%**
mit 20K Vokabular	-	86.6%	-	**86.6%**
mit 50K Vokabular	-	82.3%	-	**82.3%**

(PRI = PRINTED, CUR = CURSIVE)

Tabelle 9.1: Zusammenfassung der wichtigsten Ergebnisse

ähnlichen Bedingungen wie in dieser Arbeit erzielt. Tabelle 9.2 zeigt die Ergebnisse dieses Systems und der vergleichbaren Systeme im Überblick. Mit 93.4% für Einzelworte auf einem Vokabular von 20 000 Wörtern werden alle anderen Systeme deutlich übertroffen. Dasselbe gilt für die Satzerkennung, wo bislang nur für ein einziges System schreiberunabhängige Ergebnisse publiziert wurden.

Die Ergebnisse demonstrieren die Stärke des hier vorgestellten Ansatzes, selbst kursive Handschrift auch bei großen Vokabularen mit hoher Zuverlässigkeit in Echtzeit erkennen zu können. Erreicht wird dies unter anderem durch eine umfassende Vorverarbeitung der Daten, die sowohl in Bezug auf die Normalisierung als auch die verwendeten Merkmale weit über die Vorverarbeitung anderer, insbesondere HMM-basierter Systeme hinausgeht. Desweiteren trägt die Modellierung einzelner Zeichen durch eine Folge von Zuständen zu diesem Erfolg bei. Gegenüber anderen neuronalen Ansätzen werden deutliche Verbesserungen dadurch erreicht, daß das Vokabular direkt in die Suche der Ausgabehypothese integriert ist und nicht erst in einem nachge-

	Ansatz	Vokabulargröße	Erkennungsrate
Wörter (WA)			
Seni [Sen95]	TDNN	21 000	62.4%
Schenkel et al. [SGH94, Sch95]	TDNN	25 000	80.0%
Dolfing et al. [DHU97]	HMM	20 000	88.8%
Manke	MS-TDNN	20 000	93.4%
Sätze (WA)			
Ratzlaff et al. [RNM96]	HMM	20 000	63.0%
Manke	MS-TDNN	20 000	86.6%

Tabelle 9.2: Ergebnisse im Vergleich mit anderen Systemen

schalteten Verarbeitungsschritt Berücksichtigung findet. Durch diese Integration des Vokabulars wird zudem die Erweiterung des Systems zur Satzerkennung möglich, was in anderen neuronalen Ansätzen durch die Nachschaltung des Vokabulars verhindert wird.

9.2 Weitere Ergebnisse und Analysen

Nachdem alle bisherigen Experimente dieser Arbeit jeweils auf allen Daten der drei Datenbasen durchgeführt wurden, soll nun das Verhalten des Erkenners auf bestimmten Teilmengen der Daten für die Worterkennung untersucht werden. Unter anderem soll damit festgestellt werden, ob durch die getrennte Berücksichtigung einzelner Schreibergruppen oder Datenbasen die Erkennungsleistung des Systems weiter verbessert werden kann.

9.2.1 Einfluß der Wortlänge auf die Erkennungsrate

Ebenso wie das beim Testen verwendete Vokabular besteht die Testmenge aus Wörtern unterschiedlicher Länge. Die Länge der Wörter variiert dabei zwischen 2 und 17 Zeichen. Die durchschnittliche Länge der Testworte ist 8.2 Zeichen pro Wort, was identisch ist mit der durchschnittlichen Wortlänge des 20 000 Wörter umfassenden Testvokabulars WSJ_20K. Hier soll nun die Frage untersucht werden, ob eine Abhängigkeit der Erkennungsrate des Systems von der Wortlänge der Testdaten besteht. Dazu wird für jede Wortlänge auf allen drei Testmengen eine getrennte Erkennungsrate mit dem Vokabular WSJ_20K berechnet. Abbildung 9.1 zeigt das Verhalten des Systems für die unterschiedlichen Wortlängen anhand der Gesamterkennungsrate. Deutlich ist hier zu sehen, daß tatsächlich eine starke Korrelation zwischen der Erkennungsrate und der Wortlänge existiert. Je kürzer die Testworte sind, desto geringer ist auch die Erkennungsrate.

Zwei Gründe sind hierfür maßgeblich verantwortlich. Zum einen verringert sich mit der Wortlänge auch der bei der Erkennung zur Verfügung stehende Kontext, d.h.

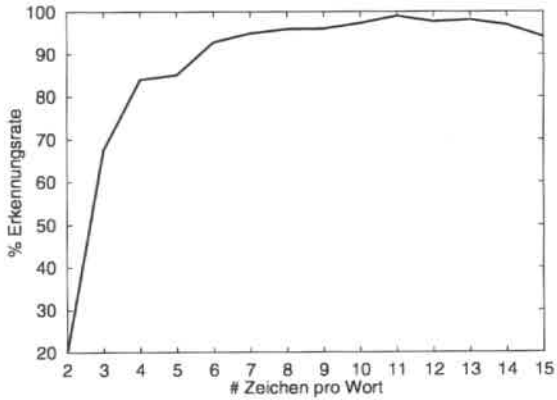


Abbildung 9.1: Die Erkennungsrate in Abhängigkeit der Wortlänge

unleserliche Zeichen innerhalb des Wortes fallen hier deutlich höher ins Gewicht. Sind z.B. nur zwei Zeichen eines Wortes unleserlich, so macht dies im Falle eines vier Zeichen langen Wortes bereits die Hälfte der gesamten Eingabe aus, kann also anhand der verbleibenden lesbaren Zeichen nur schwer korrigiert werden. Mit jedem weiteren Zeichen steht jedoch mehr Kontext zur Verfügung, und die Wahrscheinlichkeit einer richtigen Klassifikation anhand der lesbaren Zeichen steigt. Zum anderen wirkt sich die Länge des Wortes bereits auf die Normalisierung während der Vorverarbeitung aus. Dies betrifft insbesondere die Bestimmung der Schreiblinien und dadurch indirekt z.B. auch die Rotationsnormalisierung, die auf der berechneten Richtung der Basislinie beruht, oder die Größennormalisierung und Neuabtastung, die von der Kernhöhe des Wortes abhängen. Mit der Länge der Wörter verringert sich die Anzahl der lokalen Minima und Maxima der Trajektorie, auf denen die Berechnung der Schreiblinien beruht, und damit auch die Genauigkeit der Bestimmung. Analoges gilt für die Bestimmung und Korrektur des Neigungswinkels.

9.2.2 Einfluß des Schreibstils auf die Erkennungsrate

Bei den bisherigen Experimenten der Worterkennung wurde beim Training des Systems nicht zwischen Kursiv- und Druckschrift unterschieden. Das MS-TDNN muß also beide Schreibstile gleichermaßen lernen, was die Zahl der möglichen Schreibvarianten erhöht und damit die Lernaufgabe erschwert. Die Ergebnisse haben gezeigt, daß das MS-TDNN diese Aufgabe tatsächlich sehr gut löst und für beide Schreibstile hohe Erkennungsraten ermöglicht. Wie erwartet sind die Erkennungsraten für die Kursivschreiber der Datenbasen jeweils etwas geringer als die vergleichbaren Erkennungsraten der Schreiber mit Druckschrift.

Ein einfaches Experiment soll nun zeigen, wie verschieden die beiden Schreib-

stile tatsächlich sind und ob eine getrennte Erkennung der beiden Schreibstile ggf. die Erkennungsleistung erhöhen kann. Zu diesem Zweck wurde die Trainingsmenge für Einzelworte in eine Trainingsmenge mit kursiver Handschrift und eine zweite Trainingsmenge mit Druckschrift unterteilt. Mit beiden Trainingsmengen wurde jeweils ein eigener Erkenner für den speziellen Schreibstil trainiert. Beide Erkenner haben ansonsten völlig identische Architekturparameter, unterscheiden sich also bei der Evaluation lediglich durch die anhand der Trainingsmengen gelernten Gewichte des neuronalen Netzes. Tabelle 9.3 zeigt die mit diesen Erkennern erzielten Erkennungsraten auf allen Testmengen im Vergleich mit den Ergebnissen des Erkenners, der auf der Gesamttrainingsmenge trainiert wurde.

Trainingsdaten	UKA		CMU		MIT
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED
nur Kursivschrift	95.2%	**92.1%**	85.9%	**93.1%**	68.0%
nur Druckschrift	**97.8%**	72.0%	**94.3%**	46.0%	**94.0%**
Kursiv-/Druckschrift	97.1%	90.8%	92.8%	92.2%	94.7%

Tabelle 9.3: Erkennungsraten bei getrennten Schreibstilen

Mit Ausnahme der MIT Datenbasis liegen die Erkennungsergebnisse auf den Testdaten des Schreibstils, auf dem der jeweilige Erkenner trainiert wurde, geringfügig über denen des Erkenners für beide Schreibstile. Zu beachten ist jedoch, daß die Erkenner für Kursiv- und Druckschrift durch die Aufteilung der Trainingsdaten jeweils auf deutlich weniger Daten trainiert wurden. Stünden beliebig viele Trainingsdaten für jeden Schreibstil zur Verfügung, würde die jeweilige Verbesserung der Erkennungsrate vermutlich etwas deutlicher ausfallen. Es zeigt sich, daß der gemeinsame Erkenner für Druck- und Kursivschrift in der Lage ist, die Lernaufgabe im Vergleich zu den speziellen Erkennern sehr gut zu erfüllen.

Ein weiteres Ergebnis dieses Experiments ist, daß Druck- und Kursivschrift offensichtlich nicht grundsätzlich verschieden sind, sondern in vielen Fällen die gleichen Schreibvarianten für die einzelnen Zeichen beinhalten. Dies wird dadurch deutlich, daß die speziellen Erkenner für die beiden Schreibstile auch auf vielen Testdaten des jeweils anderen Schreibstils bereits gute Ergebnisse erzeugen. Dabei scheint der Erkenner für Kursivschrift besser auf die nicht gelernte Druckschrift generalisieren zu können als umgekehrt. Dies ist darauf zurückzuführen, daß zum einen für Kursivschrift insgesamt mehr Trainingsdaten zur Verfügung standen und zum anderen Kursivschrift zwar oftmals Zeichenübergänge aufweist, bei denen der Stift abgehoben wurde, aber in der Druckschrift keine kontinuierlichen Zeichenübergänge vorkommen. Der Erkenner für Druckschrift kann also die Zeichenübergänge der kursiven Schrift nicht korrekt den Ausgabeklassen zuordnen.

Fazit dieses Experiments ist, daß der hier vorgestellte Ansatz in der Lage ist, die Charakteristiken der unterschiedlichen Schreibstile durch die neuronale Erkennerarchitektur bereits so gut zu lernen, daß die Entwicklung separater Erkenner für den

jeweiligen Schreibstil kaum zu Verbesserungen führt.

9.2.3 Training und Test innerhalb der Datenbasen

Bei dem vorhergehenden Experiment wurde untersucht, wie sich ein Erkenner, der auf einen speziellen Schreibstil trainiert wurde, im Vergleich zu dem Gesamterkenner und auf den Testdaten des jeweiligen anderen Schreibstils verhält. Ein analoges Experiment soll nun für die drei unterschiedlichen Datenbasen durchgeführt werden, d.h. es soll der Frage nachgegangen werden, ob ein Erkenner, der nur auf den Daten einer einzigen Datenbasis trainiert wurde, auf den Testdaten dieser Datenbasis ein besseres Ergebnis erzielt als der Erkenner, der auf allen drei Datenbasen trainiert wurde. Tabelle 9.4 faßt die Ergebnisse dieser Untersuchung für Ziffern, Klein- und Großbuchstaben zusammen. Zum Vergleich sind dort auch die Ergebnisse des Erkenners angegeben, der auf allen drei Datenbasen trainiert wurde.

	Testmenge		
Trainigsmenge	UKA	CMU	MIT
0...9:			
UKA	**98.4%**	86.5%	90.8%
CMU	83.2%	**94.6%**	91.8%
MIT	92.1%	88.2%	**95.8%**
UKA, CMU, MIT	**97.9%**	**97.0%**	**95.3%**
A...Z:			
UKA	**90.2%**	69.5%	85.6%
CMU	88.0%	**91.6%**	92.3%
MIT	83.8%	73.4%	**95.2%**
UKA, CMU, MIT	**93.4%**	**90.5%**	**94.9%**
a...z:			
UKA	**90.9%**	74.4%	85.5%
CMU	84.2%	**88.7%**	88.3%
MIT	73.5%	68.8%	**90.1%**
UKA, CMU, MIT	**89.7%**	**89.8%**	**93.4%**

Tabelle 9.4: Erkennungsraten mit Training auf Daten einer einzigen Datenbasis

Die Ergebnisse lassen in diesem Fall keinen eindeutigen Schluß darüber zu, wie sich ein speziell für eine Datenbasis entwickelter Erkenner im Vergleich zu dem Gesamterkenner verhält, da sich für jede Datenbasis und jede Erkennungsaufgabe ein unterschiedliches Bild ergibt. In einigen Fällen ist die Erkennungsrate der speziellen Erkenner besser, in anderen Fällen wiederum schlechter als die des Gesamterkenners. Auch hier ist wieder zu beachten, daß aufgrund der Aufteilung der Daten in drei Trainingsmengen für jeden der Erkenner deutlich weniger Trainingsdaten zur Verfügung stehen.

Interessant ist eine Analyse der Ergebnisse der speziellen Erkenner auf den jeweils

fremden Datenbasen. Diese gibt Aufschluß darüber, inwieweit unterschiedliche Aufnahmebedingungen und länderspezifische Besonderheiten durch die Vorverarbeitung und das MS-TDNN ausgeglichen werden können. Zur Erinnerung: Die Datenbasen UKA und CMU wurden mit demselben Digitalisiertablett gesammelt; die Datenbasis UKA beinhaltet lediglich deutsche Schreiber, die Datenbasen CMU und MIT jeweils amerikanische Schreiber. Es bestehen also Gemeinsamkeiten zwischen der UKA und der CMU Datenbasis bzw. der CMU und der MIT Datenbasis, jedoch keine zwischen der UKA und MIT Datenbasis. Diese Gemeinsamkeiten spiegeln sich jedoch nur in geringem Maße in den Ergebnissen wider. Der auf den CMU Daten trainierte Erkenner generalisiert deutlich besser auf die MIT als auf die UKA Datenbasis. Hier überwiegt offenbar die Gemeinsamkeit des Herkunftslandes. Anders verhält es sich beim Erkenner, der auf den MIT Daten trainiert wurde. Entgegen der Erwartung einer entsprechenden Symmetrie, generalisiert dieser besser auf die UKA Datenbasis, obwohl hier weder Gemeinsamkeit in der Aufnahmehardware noch dem Herkunftsland besteht. Hier zeigt sich aber eine Symmetrie mit dem Erkenner, der auf den UKA Daten trainiert wurde. Dieser generalisiert besser auf die MIT Daten als auf die CMU Daten.

	UKA		CMU		MIT
	PRINTED	CURSIVE	PRINTED	CURSIVE	PRINTED
UKA	**95.2%**	**92.4%**	83.0%	85.1%	38.4%
CMU	95.9%	86.8%	**90.8%**	**92.4%**	82.2%
MIT	86.0%	43.1%	85.9%	26.4%	**94.4%**
UKA, CMU, MIT	97.1%	90.8%	92.8%	92.2%	94.7%

Tabelle 9.5: Erkennungsraten für Wörter mit Training auf Daten einer einzigen Datenbasis

Für die Worterkennung sind die analogen Ergebnisse in Tabelle 9.5 angegeben. Hier zeigt sich ein ähnliches Bild wie bei der Einzelzeichenerkennung. Nur in 2 Fällen ergibt sich durch die getrennte Berücksichtigung der Datenbasen ein geringfügig besseres Ergebnis.

Fazit dieses Experiments ist, daß der hier vorgestellte Ansatz in der Lage ist, die unterschiedlichen Charakteristiken der drei Datenbasen so gut zu erlernen, daß ein speziell für eine Datenbasis entwickelter Erkenner kaum zu Erkennungsverbesserungen führt. Allerdings ist auch ersichtlich, daß die Generalisierung von einer Datenbasis auf die anderen Datenbasen noch verbessert werden kann.

9.2.4 Rechts-/Linkshänder

Bei interaktiven Demonstrationen dieses Erkennungssystems an der Carnegie Mellon University wurde die subjektive Beobachtung gemacht, daß bei Linkshändern deutlich häufiger Fehlerkennungen auftreten als bei Rechtshändern. Eine objektive Bestätigung dieser Beobachtung anhand der drei hier verwendeten Datenbasen ist schwierig,

da nur einige wenige Testdaten von Linkshändern existieren. Bei der Sammlung der Datenbasis UKA wurde nicht festgehalten, ob es sich bei dem jeweiligen Schreibern um Rechts- oder Linkshänder handelt. Unter den Schreibern der Datenbasis CMU befinden sich insgesamt 18 Linkshänder, 5 davon in der Testmenge der Kursivschreiber. Von den insgesamt 12 Linkshändern der Datenbasis MIT sind durch die zufällige Aufteilung der Schreiber lediglich 2 in der Testmenge. Es stehen also Daten von zusammen 8 Linkshändern für ein einfaches Experiment mit Einzelworten zur Verfügung, um zumindest annähernd eine objektive Aussage treffen zu können.

Schreiber-ID	# Wörter	WA
CMU:		
lfcal113	30	100.0%
lfdlf20	26	84.6%
lfipr327	14	92.9%
lfras251	15	86.7%
lfrjw347	27	92.6%
Gesamt	112	92.0%
MIT:		
s102	53	88.7%
s127	53	96.2%
Gesamt	106	92.5%

Tabelle 9.6: Erkennungsraten bei Linkshändern

Tabelle 9.6 zeigt die Worterkennungsraten für jeden der linkshändigen Schreiber (in der Tabelle durch ihre Dateinamen unterschieden) bei Verwendung des Vokabulars WSJ_20K. Es zeigen sich im Mittel nur geringfügig niedrigere Erkennungsraten als die Gesamterkennungsrate über alle Daten der jeweiligen Datenbasis. Hier kann die subjektive Beobachtung bei interaktiven Demonstrationen also nicht bestätigt werden.

9.2.5 Varianz der Erkennungsrate für verschiedene Schreiber

Die in dieser Arbeit angegebenen Erkennungsraten geben nur den über eine Vielzahl von Schreibern gemittelten Wert wieder, d.h. nicht für jeden Schreiber wird diese Erkennungsrate tatsächlich erreicht. Für manche Schreiber wird der Erkenner im praktischen Einsatz also bessere Ergebnisse und damit auch einen höheren Zufriedenheitsgrad erzielen als für andere Schreiber. Einen Aufschluß darüber, welche Erkennungsrate ein Benutzer des Systems mindestens erwarten kann, gibt die in Abbildung 9.2 dargestellte Verteilung der Erkennungsraten für die Worterkennung mit dem Vokabular WSJ_20K, gemessen unabhängig vom Schreibstil über alle Schreiber der drei Datenbasen.

Die Verteilung zeigt, daß sich nahezu 30% der schreiberspezifischen Erkennungsraten tatsächlich im Bereich zwischen 90% und 95% befinden, d.h. ungefähr im Bereich

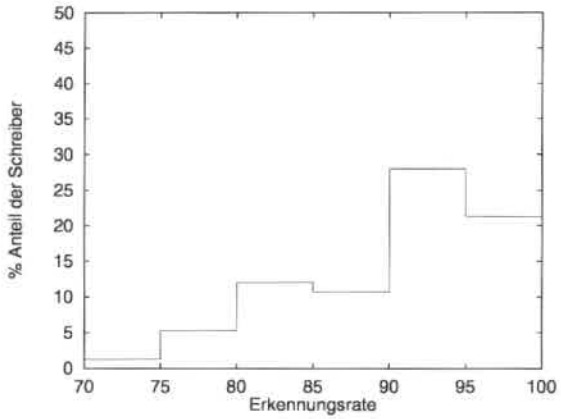


Abbildung 9.2: Verteilung der Erkennungsraten aller Schreiber

der mittleren Erkennungsrate. Weitere 20% liegen im Bereich oberhalb von 95%. Ein Großteil der verbleibenden 50% aller Schreiber erzielen eine Erkennungsrate von immerhin noch über 80%. Lediglich weniger als 7% aller Schreiber befinden sich im Bereich unter 80% Erkennungsrate. Die geringste gemessene schreiberspezifische Erkennungsrate liegt bei 72%. Sie bildet also zumindest für die hier verwendeten Datenbasen den mindestens für einen bestimmten Schreiber zu erwartenden Wert. Der Maximalwert von 100% wird insgesamt von 15 Schreibern, verteilt über alle Datenbasen und Schreibstile, erreicht. Dies ist aber nur von theoretischem Interesse, da im praktischen Einsatz dieser Wert nicht realistisch ist, zumal selbst die menschliche Erkennungsrate auf vergleichbaren Daten nur zwischen 95% und 99% liegt, wie in Experimenten ermittelt wurde [Sch95].

Abbildung 9.3 zeigt einige Beispiele fehlerkannter Wörter von 5 Schreibern mit Erkennungsraten zwischen 72 und 79%. Wie in diesen Beispielen sind auch bei den anderen Fehlerkennungen dieser Schreiber ähnliche Effekte auszumachen, die durch unsaubere Schreibweise oder Abtastfehler verursacht wurden und somit die Erkennung erschweren. Auch von Menschen sind einige dieser Fehlerkennungen nicht zweifelsfrei zu lesen. Insbesondere machen diese Beispiele noch einmal deutlich, daß eine Erkennung kursiver Handschrift ohne syntaktisches Wissen über die Sprache nicht zum Ziel führen kann.

9.2.6 N-Besten Listen

Besteht die Ausgabe eines Wort- oder Satzerkenners nicht nur aus der wahrscheinlichsten Hypothese, sondern aus einer nach ihrer Wahrscheinlichkeit sortierten Liste von N Hypothesen, so wird diese *N-Besten Liste* genannt. Eine Analyse dieser Listen gibt Aufschluß darüber, ob sich die gesuchte Ausgabe bei Fehlerkennungen des Systems

Abbildung 9.3: Beispiele fehlerkannter Wörter von Schreibern mit Erkennungsraten unter 80%

wenigstens unter den N besten Hypothesen befindet. Wird ein Erkennungsergebnis als richtig gewertet, wenn sich das gesuchte Wort unter den N besten Hypothesen befindet, so ergeben sich für $N = 1, \ldots 10$ die in Abbildung 9.4 dargestellten Erkennungsergebnisse für Einzelworte. Für $N = 1$ erhält man die bereits bekannten besten Ergebnisse des Systems.

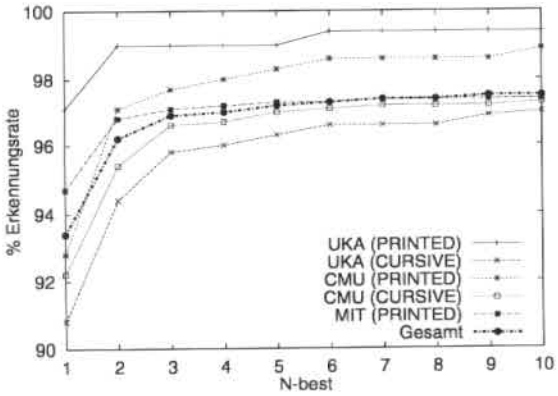


Abbildung 9.4: N-Besten Ergebnisse für Vokabular WSJ_20K

Die Ergebnisse zeigen, daß in nahezu 50% aller Fehlerkennungen die gesuchte Hypothese an zweiter Position der N-Besten Liste zu finden ist. Das gibt einen Anhaltspunkt dafür, daß die maximal mit diesem Ansatz realisierbare Erkennungsleistung noch nicht erreicht zu sein scheint. Es bleibt also Spielraum für weitere Verbesserungen, die ggf. die gesuchten Hypothesen von dem zweiten Platz der Hypothesenliste

auf den ersten Platz vorrücken lassen. Für $N > 3$ ist allerdings kein signifikanter Anstieg der Erkennungsrate mehr zu verzeichnen. Die Erfahrungen während der Entwicklung dieses Erkenners haben jedoch gezeigt, daß mit jeder erzielten Verbesserung der Erkennungsleistung der Kurvenverlauf der N-Besten Listen ähnlich ist, d.h. der prozentuale Anteil der falsch erkannten Wörter in der Liste jeweils gleich bleibt.

Die N-Besten Listen haben im übrigen neben ihrer Bedeutung für die Analyse der Leistungsfähigkeit des Systems auch eine praktische Bedeutung für den Einsatz eines Handschrifterkenners in einer realen Benutzerschnittstelle. Bei einer Fehlerkennung kann dann der Benutzer, statt das falsch erkannte Wort nochmals zu schreiben, ggf. die gesuchte Ausgabe aus der Hypothesenliste auswählen. Sofern sich bei Fehlerkennungen die richtige Hypothese häufig genug unter den besten Hypothesen befindet, kann diese Auswahl effizienter als eine Neueingabe sein.

9.3 Fehlerarten

Die Experimente dieser Arbeit haben aufgezeigt, in welchen Bereichen sich die Erkennungsrate des Systems für die Einzelzeichen-, Wort- und Satzerkennung unter den unterschiedlichen Bedingungen bewegt. Es wurde dabei immer der positive Aspekt in den Vordergrund gestellt, daß mit mehr oder weniger Abweichungen bereits hervorragende Erkennungsraten bei allen Erkennungsaufgaben erzielt werden. Doch was ist mit den Fehlerkennungen des Systems? Welcher Art sind diese Fehlerkennungen? Sind es intuitiv verständliche Fehler? Können die Fehlerkennungen einer bestimmten Erkennungskomponente zugeordnet werden? Geben sie einen Anhaltspunkt für Verbesserungsmöglichkeiten? Diesen Fragestellungen gehen die nun folgenden Abschnitte, getrennt für die jeweiligen Erkennungsaufgaben, nach.

9.3.1 Einzelzeichenerkennung

Für die Einzelzeichenerkennung ist die Frage nach der Art der Fehlerkennungen einfach zu beantworten, da hier nur Verwechslungen zwischen zwei Zeichen auftreten können. Einfüge- oder Auslassungsfehler treten hier nicht auf. Es muß also lediglich untersucht werden, ob bestimmte Verwechslungen zweier Zeichen besonders häufig vorkommen oder ob jedes Zeichen gleichermaßen zu der Fehlerrate beiträgt. Einen Aufschluß darüber ermöglicht die sog. *Konfusions-* oder *Verwechslungsmatrix*. Sie gibt für jede Zeichenklasse an, wie häufig ein zu erkennendes Zeichen dieser Klasse mit einem bestimmten Zeichen einer anderen Klasse verwechselt wurde. Anstelle der absoluten Zahl der Verwechslungen zwischen zwei Zeichen ist in den nun folgenden Abschnitten für Ziffern, Klein- und Großbuchstaben jeweils der prozentuale Anteil an der Gesamtzahl der Fehlerkennungen angegeben. Da die Einzelzeichenerkennung nur von sekundärem Interesse für diese Arbeit ist, werden nur einige wenige Besonderheiten in den jeweiligen Verwechslungsmatrizen hervorgehoben.

Ziffern

Tabelle 9.7 zeigt die Verwechslungsmatrix für die Ziffernerkennung, berechnet über alle Fehlerkennungen der drei Datenbasen. Daraus ist zu ersehen, daß die Ziffern „5" und „9" zusammen mehr als 50% aller Fehlerkennungen verursachen. Weitere 10% der Fehlerkennungen trägt die Ziffer „4" bei. Die übrigen 40% verteilen sich in etwa gleichmäßig auf die weiteren Ziffern, lediglich die Ziffer „3" wurde in allen Fällen richtig erkannt. Mit Abstand die häufigste Verwechslung ist, daß statt der Ziffer „5" die Ziffer „3" erkannt wurde. An zweiter Stelle folgt die Erkennung einer „8" anstelle der „9". Auffallend an dieser Verwechslungsmatrix ist, daß sie für die meisten Ziffernpaare asymmetrisch ist, d.h. manche Ziffern werden zwar besonders häufig mit einer bestimmten anderen Ziffer verwechselt, jedoch nicht umgekehrt.

	0	1	2	3	4	5	6	7	8	9	$\sum$
0	-	0.0	5.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.4
1	0.0	-	2.7	0.0	0.0	0.0	0.0	5.4	0.0	0.0	8.1
2	0.0	0.0	-	0.0	2.7	0.0	0.0	2.7	0.0	0.0	5.4
3	0.0	0.0	0.0	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	2.7	0.0	0.0	-	0.0	0.0	2.7	0.0	5.4	10.8
5	0.0	0.0	2.7	18.9	0.0	-	0.0	0.0	5.4	2.7	29.7
6	2.7	0.0	0.0	0.0	0.0	0.0	-	0.0	2.7	0.0	5.4
7	2.7	0.0	0.0	0.0	5.4	0.0	0.0	-	0.0	0.0	8.1
8	2.7	0.0	2.7	0.0	0.0	0.0	0.0	0.0	-	0.0	5.4
9	0.0	0.0	0.0	2.7	5.4	2.7	0.0	0.0	10.8	-	21.6

Tabelle 9.7: Verwechslungsmatrix der Ziffern (in %)

Interessant ist ein Blick auf einige der falsch erkannten Ziffern, denn nicht immer ist für einen menschlichen Betrachter des statischen Bildes der Ziffern auf Anhieb ersichtlich, was zu der jeweiligen Fehlerkennung geführt hat. Dies gilt insbesondere, da die in Abbildung 9.5a zusammen mit der Ausgabe des Erkenners dargestellten Ziffern zweifelsfrei lesbar sind. Erst die zeitliche Abfolge der Punkte macht deutlich, daß zwar nicht das vollständige statische Bild, jedoch große Teile der Trajektorie tatsächlich mit den falschen Hypothesen verwechselbar sind (Abbildung 9.5b). Die Ziffer „1" unterscheidet sich von der Ausgabehypothese lediglich durch die vertikale Position des unteren Querstrichs. Die „3" wird durch den unüblichen, vermutlich ungewollten Querstrich von rechts nach links direkt am Anfang der Trajektorie als „5" klassifiziert. Ähnliches gilt für den kurzen vertikalen Strich am Anfang der Trajektorie der Ziffer „5", der zu einer Fehlerkennung als „3" führt. Durch Weglassen des letzten diagonalen Striches von rechts unten nach links oben wird aus der „8" tatsächlich die Hypothese „2". Durch die ungewöhnliche Trajektorie wird statt der „9" eine „4" erkannt. Ähnliche Erklärungen lassen sich auch für viele der anderen Fehlerkennungen finden. Zurückzuführen sind diese Unsicherheiten bei der Erkennung im wesentlichen auf die zu kleine Trainingsmenge. Offensichtlich sind nicht alle Schreibvarianten eines

Zeichen ausreichend durch die Trainingsmenge abgedeckt, so daß selten auftretende Trajektorien zu Fehlerkennungen führen.

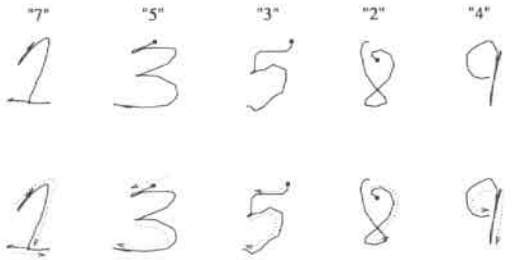


Abbildung 9.5: Fehlerkennungen bei der Ziffernerkennung

Großbuchstaben

Auch in der Verwechslungsmatrix der Großbuchstaben in Tabelle 9.8 treten ein paar Verwechslungen besonders häufig auf. Zum größten Teil sind dies die aufgrund der optischen Verwechselbarkeit erwarteten Häufungen, wie die Verwechslungen zwischen „U" und „V", „J" und „T" oder „A" und „H". Auf den ersten Blick weniger intuitiv verständlich sind die auffallend häufigen Verwechslungen zwischen „W" und „M", da diese sich üblicherweise sowohl durch das Abbild als auch die zeitliche Entstehung unterscheiden. In Abbildung 9.6 sind 5 dieser falsch erkannten Buchstaben „W" abgebildet. Die ungewöhnlichen Schreibweisen der ersten drei dieser Beispiele legen wiederum nahe, daß die Trainingsmenge nicht ausreichend groß für eine zuverlässigere Erkennung ist, da es sich hier um eher ungewöhnliche Schreibweisen dieses Buchstabens handelt, die durch die Trainingsmenge nicht ausreichend abgedeckt sind. Bei den verbleibenden zwei Beispielen kann nur vermutet werden, daß der Grund für die Fehlerkennung in dem zusätzlichen Strich am Ende des vierten „W" bzw. der wegen zu geringer Abtastrate unsauberen Schreibweise des letzten „W" zu suchen ist.

Kleinbuchstaben

Die Verwechslungsmatrix der Kleinbuchstaben in Tabelle 9.9 weist eine ähnlich breite Verteilung der Fehlerkennungen wie die der Großbuchstaben auf. Die häufigste

Abbildung 9.6: Als „M" klassifizierte Beispiele des Buchstabens „W"

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	$\sum$
A	-	0	0	0	0	0	0.5	2.6	0	0	0.5	0	0	0	0	0	0	0	0	0	0.5	0	0	0.5	0	0	4.6
B	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	-	0	0.5	0	0.5	0	0	0	0	1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.5
D	0	1.0	0	-	0	0.5	0	0	0.5	0.5	0	0	0	0	0.5	0.5	0	0	0	0.5	0	0	0	0	0	0	4.0
E	0	0	0	0	-	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0
F	0	0	0	0	0	-	0.5	0	0	0	1.0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0.5	2.5
G	0.5	0	0.5	0	0	0	-	0.5	0	0	0	0	0	0	0.5	0	0	0	1.0	0	0	0	0	0	1.0	0	4.0
H	1.5	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	2.5
I	0	0	0	0.5	1.5	0	0.5	0	-	2.0	0	0.5	0	0	0	0	0.5	0	0.5	0	0.5	0	0	0	0	0	6.5
J	0	0	0	0	0	0	0	0	1.0	-	0	0	0	0	0	0	0	0	0	3.6	0	0	0	0.5	0	0	5.1
K	0	0	0	0	0	0	0.5	0.5	0	0	-	0	0	0	0	0	0.5	0	0	0	0.5	0	0	0	0	0	2.0
L	0	0	0	0	0	0.5	0.5	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0.5	2.0
M	0.5	0	0	0	0	0	0	0	0	0	0	0	-	0.5	0	0	0	0	0	0.5	0	0	0	0	0	0	1.5
N	0	0	0	0	0	0	0	0.5	0	0	0	0	5.1	-	0	0	0	0	0	0	0.5	0	1.0	0	0	0	7.1
O	0	0	0	0.5	0	0	0	0	0	1.0	0	0	0	0	-	0	1.0	0	0	0	0.5	0.5	0	0	0	0	3.5
P	0	0	0	0.5	0	0.5	0	0	0	0	0.5	0	0	0	0	-	0	2.0	0	0.5	0	0	0	0	0	0	4.0
Q	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0.5	1.0
R	0	2.0	0	0.5	0	0	0	0	0	0	1.0	0	0	0	0	1.0	0	-	0	0	0	0	0	0	0	0	4.5
S	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	1.0	0	2.0
T	0	0	0	0	0	2.6	0	0	0	0	0	0.5	0	0	0	0	0	0	0	-	0	0	0	0	0	0	3.1
U	0.5	0	0.5	0	0	0	0	0.5	0	0	0	0	1.5	0.5	3.6	0	0	0	0	0	-	3.1	1.5	0	0	0	11.7
V	0	0	0	0	0	0	0	0	0	0	1.5	0	0	0.5	0	0	0	0	0	0	2.6	-	1.0	1.0	0.5	0	7.1
W	0	0	0	0	0	0	0.5	2.0	0	0	0	0	3.6	0.5	0	0	0	0	0	0	0	0.5	-	0	0	0	7.1
X	0.5	0	0	0	0	0	0	0.5	0	0	2.0	0	0	0	0	0	0.5	0	0	0	0	0	0	-	0.5	0	4.0
Y	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0.5	0	0	0	2.0	-	0	3.0
Z	0	0	0	0	0	0.5	0	0	0	1.0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	2.5

Tabelle 9.8: Verwechslungsmatrix der Großbuchstaben

Fehlerkennung tritt wie erwartet zwischen den Buchstaben „e“ und „l“ auf, da diese auch von Menschen ohne Kontext zum Größenvergleich schwer zu unterscheiden sind. Diese Fehlerquelle ist auch durch eine weitaus größere Trainingsmenge kaum zu vermeiden. Ebenso verständlich sind die häufigen Verwechslungen zwischen „y“ und „g“, „a“ und „d“, „h“ und „b“, „v“ und „n“ oder auch „n“ und „r“, die sich jeweils nur in einem sehr kleinen Bereich der Trajektorie unterscheiden. Abbildung 9.7 zeigt einige Beispiele dieser Fehlerkennungen.

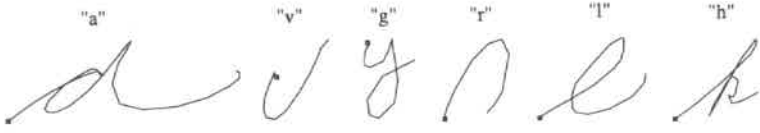


Abbildung 9.7: Die falsch erkannten Buchstaben „d“, „u“, „y“, „n“, „e“ und „k“

Fazit

Alle in den vorgehenden Abschnitten präsentierten Beispiele aus den Testdaten der drei Datenbasen machen deutlich, daß die Einzelzeichenerkennung trotz des kleinen Vokabulars von nur 10 bzw. 26 Zeichen aufgrund der Verwechselbarkeit, insbesondere bei natürlicher Handschrift, ein schwieriges Problem darstellt. Noch weitaus schwieriger gestaltet sich das Problem, wenn Ziffern, Klein- und Großbuchstaben gemein-

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	$\sum$
a	-	0	0	2.5	0	0	0	0	0	0	0	0	0	0.4	0	0	0.4	0	0	0	0.8	0	0	0.4	0	0.4	4.9
b	0.4	-	0	0	0	0.4	0	0.4	0	0	0	0.4	0	0	0	0	0	0	0	0.8	0.4	0	0	0	0	0	2.8
c	0	0	-	0	0.8	0	0	0	0	0	0	0	0	0	1.2	0	0	0	0	0	0	0	0	0	0	0.8	2.8
d	1.2	0	0	-	0	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.6
e	0	0	0.4	0	-	0	0	0	0	0	0	1.2	0	0	0	0.4	0	0	0	0.4	0	0	0	0	0	0	2.4
f	0	0	0	0	0	-	0	0	0	0.4	0	1.2	0	0	0	0.4	0	0	0	1.2	0	0	0	0	0	0	3.2
g	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	1.2	0	0	0	0	0	0	0	0.8	0	2.0
h	0	2.1	0	0	0	0	0	-	0	0	0	0.4	0	0.8	0	0	0	0	0	0	0	0	0.4	0	0	0	3.7
i	0	0	0	0	0	0	0	0	-	0	0	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4
j	0	0	0	0	0	0	0.4	0	0	-	0	0.4	0	0	0	0	0	0.4	0	0	0	0	0	0	1.2	0	2.4
k	0	1.2	0	0	0	0	0	1.2	0	0	-	0	0	0.4	0	0.4	0	0	0	0	0	0	0	0	0	0	3.2
l	0.4	0.8	0	0.4	3.7	0	0	0	0.4	0	0	-	0	0	0	0	0	0.4	0	0	0.4	0.4	0	0	0	0	6.9
m	0	0	0	0	0	0	0	0	0	0	0	0	-	2.1	0	0	0	0.8	0	0	0	0	1.7	0	0	0	4.6
n	1.2	0	0	0	0	0	0	0	0	0	0	0	1.7	-	0	0	0	1.7	0	0	0.4	0	0.8	0	0	0	6.0
o	0.8	0	0	0	0	0.4	0	0	0	0	0	0	0	0.4	-	0	0	0.8	0	0	0.8	0	0.4	0	0	0	3.6
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0.4	0	0.4	0	0	0	0	0	0	0.8
q	0	0	0	0.4	0	0.4	0.8	0	0	0	0	0	0	0	0.4	0	-	0	0	0	0	0	0	0.8	0	0	2.8
r	0	0	0	0	0	0.4	0.4	0.4	0	0	0	0	0	2.1	0.4	0.8	0	-	0	0.4	1.2	0.8	0	0	0	0	6.9
s	0.8	0.8	0	0	0	0	1.2	0	0	0	0	0	0	0	0	0.4	0	0	-	0	0	0	0	0	0	0	3.2
t	0	0.4	0	0	0	0.8	0	0	0	0	0	0	0	0	0	0	0	0.4	0	-	0.4	0	0	0	0	0	2.0
u	0.8	0	0	0.4	0	0	0	0	0	0	0.4	0	0	0.4	0.4	0	0	0	0	0	-	2.1	1.7	0	0	0	6.6
v	0	0	0	0	0	0	0	0	0	0	0.4	0	0	0.4	2.1	0	0	1.7	0	0	2.9	-	1.7	0.4	0	0.4	10.0
w	1.2	0	0	0	0	0	0	0.8	0	0	0	0	0	1.7	0	0	0	0	0	0	2.1	0.4	-	0.4	0	0	6.6
x	0	0	0	0	0	0	0	0	0	0	0.4	0	0	0.4	0	0	0	0	0	0	0	0	0	-	0.8	0	1.6
y	0	0	0	0	0	0	3.3	0	0.4	0	0	0.4	0	0	0	0	0	0	0	0.4	0	0	1.2	-	0.4		6.1
z	0	0	0	0	0	0	0.8	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	-		1.6

Tabelle 9.9: Verwechslungsmatrix der Kleinbuchstaben

sam von einem einzigen Erkenner verarbeitet werden sollen. Viele der beobachteten Fehler bei der Erkennung von Einzelzeichen sind durch eine Vergrößerung der Trainingsmenge zu beheben. Insbesondere die breite Verteilung der Fehlerkennungen in den Verwechslungsmatrizen auf alle Zeichen legt nahe, daß sie hauptsächlich durch ungewöhnliche oder undeutliche Schreibweisen der Zeichen verursacht werden, wie auch die Beispiele zeigen. Einige Fehlerquellen werden jedoch aufgrund der Verwechselbarkeit auch durch zusätzliche Trainingsdaten nicht beseitigt werden können. Diese Fehlerquellen ergeben sich zum einen durch die fehlenden Kontextinformationen, wie bei der Verwechslung von „e" und „l", oder zum anderen dadurch, daß für manche Zeichen, wie z.B. „r" und „n", Schreibweisen existieren, die bei anderen Schreibern für das jeweils andere Zeichen verwendet werden. Insgesamt geben die Untersuchungen keinen weiteren Hinweis auf systematische Fehler, die durch die Vorverarbeitung oder Erkennerarchitektur bedingt sind.

9.3.2 Worterkennung

Während bei der Einzelzeichenerkennung die Verwechslungsmatrix bereits vollständig die auftretenden Fehler charakterisiert, ist bei der Worterkennung eine Charakterisierung in dieser kompakten Form nicht möglich, da die Fehlerquellen vielfältiger sind. Eine Fehlerkennung kann durch eine einfache Verwechslung eines Zeichens, das Auslassen oder Einfügen eines Zeichens oder auch eine beliebige Kombination dieser Fehlerarten verursacht werden. Ein Maß für den Unterschied zwischen dem Refe-

renzwort und der Worthypothese des Erkenners ist die bereits für die Berechnung der Buchstabenakkuratheit verwendete Editierdistanz. Sie wird umso größer, je mehr sich die Worthypothese von dem gesuchten Wort unterscheidet, d.h. die Anzahl der Verwechslungen, Auslassungen und Einfügungen steigt. Daher bildet sie die Grundlage der Untersuchungen in diesem Abschnitt. Die Untersuchungen finden auf den Fehlerkennungen des Standardworterkenners mit dem Vokabular WSJ_20K statt, der eine Gesamtworterkennungsrate von 93.4% und Buchstabenakkuratheit von 97% auf den 4746 getesteten Wörtern der drei Datenbasen erzielt. Die verbleibende Wortfehlerrate von 6.6% entspricht 314 falsch erkannten Wörtern, verursacht durch insgesamt 699 Verwechslungen, 207 Einfügungen und 241 Auslassungen einzelner Zeichen. Diese 314 Wörter fließen in die folgenden Untersuchungen ein.

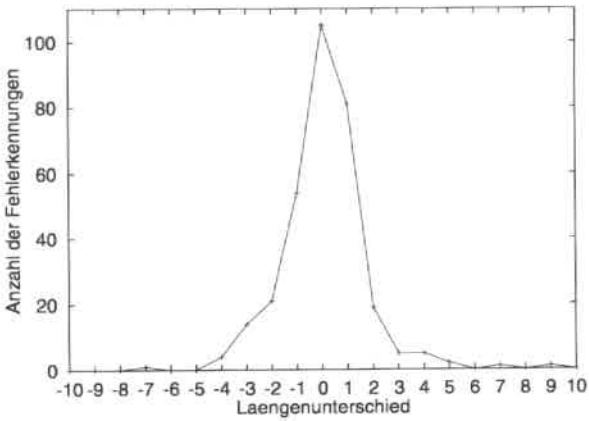


Abbildung 9.8: Längenunterschiede zwischen falschen Worthypothesen und den Referenzworten

Eine erste einfache Analyse der Fehlerkennungen untersucht, wie sich die Länge der Worthypothese zu der Länge des Referenzwortes verhält. In Abbildung 9.8 ist die Statistik über die Längenunterschiede der 314 falschen Worthypothesen zu den jeweiligen Referenzworten dargestellt. Ein negativer Längenunterschied bedeutet hierbei, daß die Worthypothese weniger Zeichen enthält als das Referenzwort, für positive Längenunterschiede entsprechend umgekehrt. Deutlich ist in der Abbildung zu sehen, daß bei einem Drittel der Fehlerkennungen (105 Wörter) zumindest die Länge der Worthypothesen mit der des Referenzwortes übereinstimmt, d.h. entweder nur Zeichenverwechslungen vorkommen oder aber Auslassungen und Einfügungen sich gegenseitig aufheben. Bei 43% aller Fehlerkennungen besteht lediglich ein Längenunterschied von ± 1 Zeichen. Weitere 20% der Worthypothesen sind mindestens 2 oder 3 Zeichen länger bzw. kürzer als die Referenzworte. Größere Längenunterschiede sind nur noch in 5% aller Fälle zu beobachten. Insgesamt ist also die Länge der generier-

ten Worthypothesen bei Fehlerkennungen eine gute Annäherung an die tatsächlich gesuchte Länge. Dies erklärt auch insbesondere das obige Verhältnis von 699 Verwechslungen zu insgesamt nur 448 Einfügungen und Auslassungen.

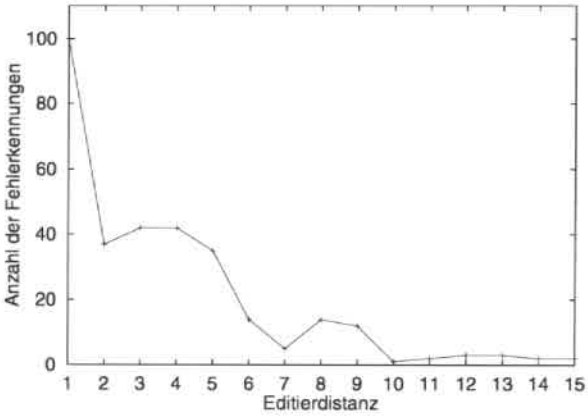


Abbildung 9.9: Anzahl der Fehlerkennungen mit gegebener Editierdistanz

Doch wie verteilen sich nun die insgesamt 1 147 Verwechslungen, Einfügungen und Auslassungen auf die 314 falsch erkannten Wörter? Verteilen sie sich gleichmäßig über alle Wörter, d.h. treten jeweils in etwa die theoretisch erwarteten 3 bis 4 Fehler pro Wort auf, oder teilen sich die Fehlerkennungen in „leichte" Fehler mit wenigen Unterschieden zwischen Hypothese und Referenz und „schwere" Fehler mit entsprechend größeren Unterschieden auf? Einen ersten Anhaltspunkt dazu liefert Abbildung 9.9. Dort ist für jede Editierdistanz die Häufigkeit ihres Auftretens unter den Fehlerkennungen angegeben. Wie hieraus zu ersehen, ist bei etwa 30% aller Fehlerkennungen, d.h. 100 Wörtern, die Worthypothese nur durch eine einzige Editieroperation von der gesuchten Ausgabe entfernt. 43 dieser 100 Wörter entstehen durch Verwechslung, 51 durch Auslassen und 6 durch Einfügen eines einzigen Zeichens. Bevor die Hypothesen mit größerer Editierdistanz näher untersucht werden, zunächst eine Analyse dieser „leichten" Fehler: 50% der Verwechslungen entstehen durch Fehler der Groß- und Kleinschreibung am Anfang des Wortes, insbesondere Verwechslungen von „s" und „S" oder „g" und „G". Die verbleibenden 50% sind gleichmäßig über die restlichen Kleinbuchstaben verteilt, wobei insbesondere auch die bereits bei der Einzelzeichenerkennung beobachteten Verwechslungen auftreten. Die Auslassungen und Einfügungen zeigen außer einem gehäuften Auslassen bzw. Hinzufügen eines Endungs-„s", die oftmals durch unsaubere Schreibweise eines abschließenden „s" oder einen zusätzlichen „Schnörkel" am Wortende verursacht werden, keine systematischen Auffälligkeiten. Es folgen nun ein paar Beispiele von Fehlerkennungen für Wörter, die sich nur in einem Zeichen von dem Referenzwort unterscheiden. Von links nach rechts ist jeweils die

handschriftliche Eingabe, das Referenzwort und der Anfang der geordneten N-Besten Liste des Erkenners angegeben:

Verwechslung


Geography ⟶ „geography Geography geographic ...“


bulb ⟶ „bull bulls buller lubell bulb ...“

Auslassung


Revving ⟶ „Reving Revving Reading ...“


pirouettes ⟶ „pirouette pirouettes priorities ...“

Einfügung


confirm ⟶ „confirms confirm confirmed confirming ...“


Cafeteria ⟶ „Cafetieria Cafeteria Castellanos ...“

Was bereits an diesen wenigen Beispielen deutlich wird, bestätigt sich auch bei der Untersuchung aller 100 Fehlerkennungen mit einer Editierdistanz von 1: Bei diesen Wörtern befindet sich die korrekte Antwort sehr weit vorne in der N-Besten Liste. In 75% dieser Fehlerkennungen ist bereits die zweite Hypothese die gesuchte, in 95% befindet sich die gesuchte Antwort unter den 5 besten Hypothesen.

Eine Untersuchung der Fehlerkennungen mit einer Editierdistanz größer als 1 zeigt keine systematischen Fehler. Wie Abbildung 9.9 bereits zeigte, treten jeweils etwa 13% aller Fehlerkennungen mit Editierdistanzen von 2, 3, 4 und 5 auf. Mit steigender Editierdistanz zwischen der Worthypothese und dem Referenzwort verschiebt sich das gesuchte Wort auch immer weiter nach hinten in der N-Besten Liste des Systems, wie Abbildung 9.10 zeigt. Dort ist für jede Editierdistanz die Anzahl der Fehlerkennungen angegeben, die diese Editierdistanz zu dem Referenzwort haben und bei denen sich gleichzeitig die gesuchte Antwort in der 10-Besten Liste des Erkenners befindet. Wie hieraus zu ersehen, nimmt bereits bei einer Editierdistanz von 2 die Wahrscheinlichkeit, daß das gesuchte Wort in der 10-Besten Liste gefunden werden kann, deutlich ab. Einige Beispiele von Fehlerkennungen mit unterschiedlichen Editierdistanzen verdeutlichen, daß mit der Größe der Distanz das intuitive Verständnis für die Fehler abnimmt:

Editierdistanz 2


Stahl ⟶ „Still Stahl still Shell Small stall...“


syring ⟶ „surging buying syring Spring swinging...“

Editierdistanz 3


Spector ⟶ „Spectrum spectrum Spector specter...“


Omitted ⟶ „permitted omitted submitted Omitted...“

Editierdistanz 4


clucas ⟶ „clinical clerical clucas clinics...“

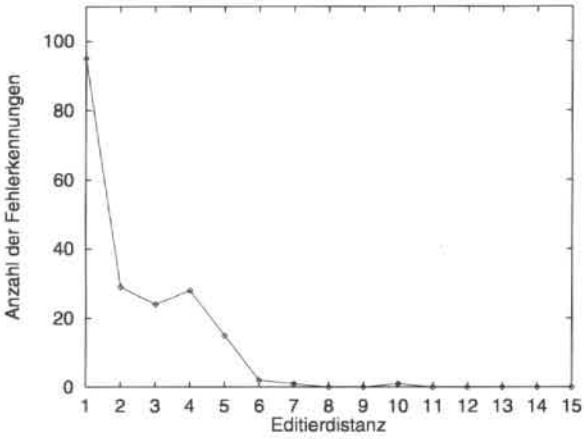


Abbildung 9.10: Anzahl der Fehlerkennungen, bei denen sich mit gegebener Editierdistanz das gesuchte Wort in der 10-Besten Liste des Erkenners befindet

 illegal ⟶ „elegant delegates delegate illegal illegally...“

Editierdistanz 5

 Etruscan ⟶ „Egyptian quaint question equation...“

 amnesty ⟶ „austerity auditing quietly fraudulently...“

Editierdistanz 6

 menzer ⟶ „multiyear midyear merger hamburger...“

 seeger ⟶ „village villages collagen Village Collagen...“

Insbesondere die schweren Fehler mit großen Editierdistanzen lassen den Schluß zu, daß die in dieser Arbeit verwendete Trainingsmenge nicht ausreichend alle Schreibstile abdeckt. Eine Vergrößerung der Trainingsmenge könnte also auch hier die Erkennungsrate verbessern. Die große Anzahl leichter Fehler, bei denen sich die Worthypothese und das Referenzwort nur in einem einzigen Buchstaben unterscheiden, ist nicht nur für die Analyse der Ergebnisse interessant, sondern hat auch Einfluß auf den praktischen Einsatz des Systems. Ähnlich wie bereits am Beispiel der N-Besten Liste gezeigt wurde, bedeutet dies, daß im Falle einer Fehlerkennung nur ein einziges Zeichen der Ausgabehypothese des Systems durch den Benutzer korrigiert werden muß, während bei schweren Fehlern oftmals ein vollständiges Neuschreiben des Wortes erforderlich ist.

9.3.3 Satzerkennung

Die Editierdistanz ist auch für die Analyse der Fehlerkennungen bei der Satzerkennung ein wichtiges Hilfsmittel. Im Gegensatz zur Worterkennung, wo die Editierdistanz auf Buchstabenebene betrachtet wurde, ist hier in erster Linie die Distanz der Satzhypothese von dem Referenzsatz auf Wortebene von Interesse. In dieser Arbeit wurde gezeigt, daß durch die Verwendung von Sprachmodellen die Erkennungsrate bei Sätzen verbessert werden kann. Bevor untersucht wird, welche Fehler durch die Sprachmodelle hauptsächlich vermieden werden, folgt zunächst eine Analyse der Fehlerkennungen ohne Sprachmodell auf den 161 Sätzen der CMU Datenbasis mit dem Vokabular WSJ_20K.

Satzerkennung ohne Sprachmodelle

Für das Vokabular WSJ_20K erzielt das hier vorgestellte System bei den getesteten Sätzen eine Wortakkuratheit von 81.7%. Die Erkennungsrate liegt damit deutlich unter der vergleichbaren Erkennungsrate von 93.4% der Einzelworterkennung für dasselbe Vokabular. Die Fehler werden durch 376 Verwechslungen, 28 Einfügungen und 20 Auslassungen verursacht. Es zeigt sich also in etwa das gleiche Bild wie bei Einzelworten, d.h. auch hier sind Verwechslungen die hauptsächliche Fehlerquelle. Zumindest etwa die Hälfte der Differenz der Erkennungsrate zwischen der Einzelwort- und Satzerkennung kann durch die zusätzlichen Auslassungen und Einfügungen erklärt werden, die bei der Worterkennung nicht auftreten. Die Wortakkuratheit steigt auf etwa 86%, wenn alle Auslassungen und Einfügungen sowie Verwechslungen, die durch eine dieser Auslassungen und Einfügungen direkt verursacht werden, bei der Berechnung der Fehler unberücksichtigt bleiben. Tatsächlich treten Einfügungen und Auslassungen in nahezu allen Fällen in der Form auf, daß ein Wort des Referenzsatzes durch zwei kürzere Wörter bzw. zwei kürzere Wörter durch ein langes Wort in der Satzhypothese ersetzt werden, wie folgendes Beispiel aus den Testdaten zeigt:

Referenzsatz: „... accident now seemed as obvious as it had ... "
Satzhypothese: „... accident now See Net as obvious assist had ... "

Für die übrigen Verwechslungen, die nicht durch eine Einfügung oder Auslassung verursacht werden, zeigt Abbildung 9.11 das Längenverhältnis zwischen der Worthypothese und dem zugehörigen Referenzwort. Die Verteilung der Längenunterschiede ist nahezu identisch mit der der Einzelworte. Die Mehrzahl der falschen Worthypothesen hat entweder die gleiche Länge wie das Referenzwort oder aber ist nur unwesentlich länger oder kürzer. Etwa 40% der falsch erkannten Wörter unterscheiden sich lediglich durch eine einzige Editieroperation von der Referenz, wie die in Abbildung 9.12 dargestellten Häufigkeiten bestimmter Editierdistanzen belegt. Also auch hier wird die Mehrzahl der Fehlerkennungen durch „leichte" Fehler verursacht.

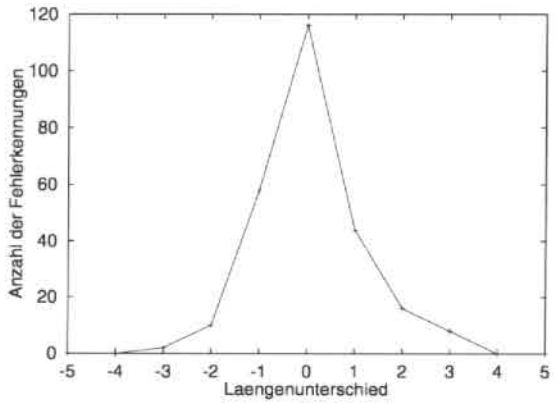


Abbildung 9.11: Längenunterschiede zwischen falschen Worthypothesen und den Referenzworten

Satzerkennung mit Sprachmodellen

Durch die Einführung eines Bigramms in die Satzerkennung wurde die Fehlerrate um etwa 25% reduziert. Das Bigramm bewirkt in diesem Fall eine Reduzierung der Anzahl der Worteinfügungen von 28 auf 9 und der Wortauslassungen von 20 auf 13. Die verbleibende Verbesserung wurde durch Vermeidung von Verwechslungsfehlern erreicht. Zu etwa 90% handelt es sich bei diesen Verwechslungsfehlern, die durch das Bigramm verbessert werden, um "leichte" Verwechslungen, d.h. Verwechslungen, bei denen eine Editierdistanz von 1 oder 2 zu dem entsprechenden Referenzwort auftritt. Verwechslungen mit größeren Editierdistanzen konnten durch das Sprachmodell fast nicht korrigiert werden. Hier tritt offensichtlich der bereits bei Einzelworten beobachtete Effekt in Erscheinung, daß sich mit steigender Editierdistanz das gesuchte Wort immer seltener in der N-Besten Liste des Systems befindet. Dies bedeutet insbesondere auch, daß die Bewertung des gesuchten Wortes so weit hinter der besten Bewertung zurückliegt, daß auch der Einsatz des Sprachmodells keine Korrektur mehr bewirken kann.

Fazit

Die Untersuchungen dieses Abschnittes zeigen, daß mit der Modellierung des Leerzeichens und dem Einsatz von Sprachmodellen nur noch wenige Einfügungen und Auslassungen bei der Satzerkennung auftreten. Verwechslungen stellen also das wesentliche verbleibende Problem dar. Es ist zu erwarten, daß eine Verbesserung der Erkennungsleistung auf Einzelworten auch eine Verbesserung der Erkennungsleistung auf Sätzen mit sich bringt. Eine genauere Analyse, ob die Verwechslungen durch ungenaue Segmentierung trotz des Zustands für das Leerzeichen verursacht werden, ist

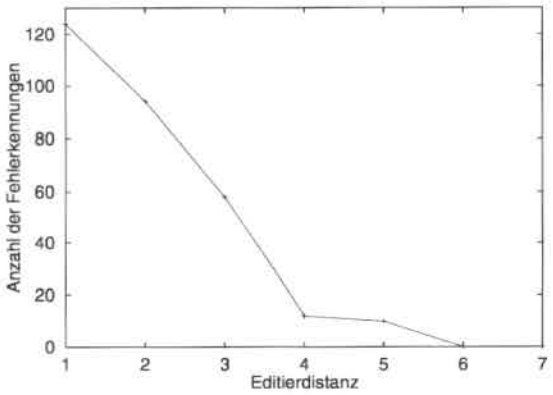


Abbildung 9.12: Anzahl der Fehlerkennungen mit gegebener Editierdistanz

derzeit nicht möglich, da zu diesem Zeitpunkt keine Testsätze zur Verfügung stehen, für die die Wortgrenzen bekannt sind. Lägen solche Sätze vor, könnten die einzelnen Wörter der Sätze separat getestet werden und die dabei entstehende Worterkennungsrate mit der Wortakkuratheit verglichen werden. Läge die Worterkennungsrate hierbei in etwa im Bereich der in dieser Arbeit präsentierten Einzelwortergebnisse von über 93%, so könnte daraus geschlossen werden, daß sich die Verwechslungen entweder aus Segmentierungsfehlern oder aber auch durch Fehler der Vorverarbeitung auf Satzebene ergeben.

9.4 Zusammenfassung

In diesem Kapitel wurden die Erkennungsleistung des Systems auf unterschiedlichen Teilmengen der Testdaten und die Fehlerkennungen des Systems analysiert. Die Ergebnisse zeigen, daß durch spezielles Training des Systems auf einen bestimmten Schreibstil oder eine Datenbasis nur geringfügige Verbesserungen der Erkennungsrate zu verzeichnen sind. Die Charakteristiken unterschiedlicher Schreibstile und Datenbasen können also bereits mit dem Gesamtsystem erfaßt und gelernt werden.

Die Analyse der N-Besten Liste und der Fehlerkennungen lassen erkennen, daß insbesondere durch eine Vergrößerung der Trainingsmenge, die eine breitere Abdeckung aller Schreibstile mit sich bringt, weitere Verbesserungen der Erkennungsrate zu erzielen sind. Direkte Fehlerquellen in einzelnen Teiles des Systems können durch die Analysen der Fehlerkennungen nicht ausgemacht werden.

Kapitel 10

Zusammenfassung und Ausblick

10.1 Zusammenfassung

In dieser Arbeit wurde ein schreiberunabhängiger On-line Erkenner für kursive Handschrift vorgestellt, der ohne jegliche Einschränkungen bezüglich der Eingabe hervorragende Erkennungsraten bei allen hier getesteten Erkennungsaufgaben erzielt. Bei der Evaluierung anhand von drei unterschiedlichen Datenbasen mit Handschriftproben von mehr als 500 Schreibern zeigt sich das System robust gegenüber schreiber- und länderspezifischen Schreibvarianten und unterschiedlichen technischen Gegebenheiten. Kursivschrift mit fließenden Übergängen zwischen den einzelnen Zeichen wird nahezu genauso gut erkannt wie Druckschrift mit deutlichen Zwischenräumen. Selbst handschriftliche Eingaben mit deformierten oder sogar fehlenden Zeichen können dank der Vermeidung früher Segmentierungsentscheidungen und der vokabulargesteuerten Suche oftmals erkannt werden. Durch den streng hierarchischen Aufbau des Systems eignet es sich mit leichten Modifikationen der Vorverarbeitung für die Erkennung von Einzelzeichen, Wörtern und Sätzen gleichermaßen. Die zeitliche Natur des ursprünglichen Eingabesignals wird durch alle Verarbeitungsstufen hinweg bewahrt und so großer Nutzen aus den dynamischen Schreibinformationen gezogen.

Die **Vorverarbeitung** beschränkt sich auf die Berechnung von Merkmalen, die sich nahe an dem punktbasierten Eingabesignal orientieren. Merkmale einer höheren Ebene werden erst in nachfolgenden Erkennungsschritten gewonnen. So werden Verarbeitungsschritte vermieden, die zu einem Informationsverlust oder gar zu frühen Fehlentscheidungen führen würden, die später nicht mehr korrigiert werden könnten. Die Normalisierung entfernt alle unerwünschten Variabilitäten aus dem Eingabesignal, die nicht auf den Schreibstil zurückzuführen sind und die Erkennung erschweren. Auf die verwendeten Aufnahmegeräte zurückzuführende Effekte wie unterschiedliche Abtastraten und Abtastfehler werden ebenso beseitigt wie schreiberspezifische Effekte, d.h. Rotationen, Zeichenneigungen, unterschiedliche Größen und Schreibgeschwindigkeiten. Die aus der normalisierten Eingabe für jeden Datenpunkt berechneten Merkmalsvektoren kombinieren Merkmale, die lokale Charakteristiken der Trajektorie kodieren, mit globalen Merkmalen, die eine weiträumigere, von der Zeit unabhängige

Sicht auf die jeweilige Umgebung eines Punktes beinhalten. Diese Repräsentation bezieht so ein Stück weit eine Grauwertbild-Darstellung der Eingabe mit in die Erkennung ein, ohne auf die zeitlichen Informationen verzichten zu müssen.

Aus der vorverarbeiteten Eingaberepräsentation erzeugt eine **neuronale Netzarchitektur** eine Folge von Zeichenhypothesen. Das hierzu verwendete Multi-State Time Delay Neural Network, ursprünglich eine Weiterentwicklung des TDNN für Spracherkennungsaufgaben, zeigt auch bei der Handschrifterkennung, daß es fähig ist, Zeichen bzw. Teile von Zeichen unabhängig von ihrer Position im Eingabesignal und ihrer zeitlichen Länge mit hoher Zuverlässigkeit zu klassifizieren. Neben dieser Verschiebungsinvarianz ist das zweite wesentliche Merkmal des MS-TDNN die Fähigkeit, zeitliche Zusammenhänge aufeinanderfolgender Merkmalsvektoren zu erfassen. Die Klassifikationsentscheidung für einen Ausgabevektor des MS-TDNN basiert nicht auf einem Merkmalsvektor für einen bestimmten Zeitpunkt alleine, sondern auf einem größeren zeitlichen Kontext, der auch die Vorgänger und Nachfolger des Merkmalsvektors einbezieht. Die Experimente haben gezeigt, daß die in dieser Arbeit präsentierten hohen Erkennungsraten erst durch die geeignete Modellierung der Zeichen durch eine Folge von abstrakten Zuständen erzielt werden. Diese Modellierung verteilt die schwierige Klassifikationsaufgabe für jedes Zeichen auf mehrere Ausgabeneuronen, wodurch die Lernaufgabe für das MS-TDNN wesentlich erleichtert wird. Die zeitliche Anpassung der Zustände an die jeweilige Eingabesequenz ist direkt in die MS-TDNN Architektur integriert.

Durch die Realisierung dieser wichtigen Erkennerkomponente mittels eines lernenden neuronalen Netzes wird die Unabhängigkeit des Systems von einem bestimmten Schreibstil und Schreiber erreicht. Durch den Schreibstil oder innerhalb eines Schreibstils durch Länder- oder Schreibergewohnheiten bedingte unterschiedliche Schreibvarianten der Zeichen werden anhand einer großen Menge von Schriftproben automatisch gelernt. Es ist keine explizite manuelle Modellierung der Zeichenvarianten erforderlich, zumal diese aufgrund der Vielzahl der Varianten zwangsläufig unvollständig bleiben müßte. Das hier verwendete mehrstufige Training des MS-TDNN erschließt große Datenmengen, für die die Zeichen- oder Wortgrenzen nicht bekannt sind. Lediglich eine geringe Datenmenge mit manuell segmentierten Daten ist erforderlich. Hat das MS-TDNN anhand dieser kleinen Datenmenge bereits hinreichend gut gelernt, Zeichen zu klassifizieren, wird die Trainingsmenge um die unsegmentierten Daten ergänzt, deren Zeichengrenzen in den folgenden Trainingsiterationen automatisch bestimmt werden können.

Für die Einzelzeichenerkennung ist das MS-TDNN bereits das Ende des Erkennungsprozesses. Mit getrennt für Ziffern, Klein- und Großbuchstaben trainierten MS-TDNNs werden bei allen drei Erkennungsaufgaben jeweils über 90% Erkennungsrate erreicht. Eine einfache Erweiterung des MS-TDNN für die Worterkennung ohne syntaktische Informationen über die zugrundeliegende Sprache führt allerdings zu einer weniger zufriedenstellenden Worterkennungsrate von nur 26%, da bereits ein falsch erkanntes Zeichen eines Wortes zu einer Fehlerkennung des gesamten Wortes führt. Die

volle Leistungsfähigkeit von über 93% Worterkennungsrate erzielt das vorgestellte System erst durch den Einsatz einer vokabulargesteuerten **Suche**, die die Fehlerkennung einzelner deformierter oder unlesbarer Zeichen durch den Kontext der vorangehenden und nachfolgenden Zeichen korrigieren kann. Die Suche der wahrscheinlichsten Ausgabe aus den Zeichenhypothesen des MS-TDNN erzeugt so nur noch legale Wörter des vorgegebenen Vokabulars. Hier hat sich die Repräsentation der Wortmodelle des Vokabulars als Baumstruktur mit einer Strahlensuche als effizienteste Lösung bezüglich der Erkennungsrate und Antwortzeit des Systems erwiesen. Selbst bei Vokabularen mit 50 000 Wörtern werden so noch Worterkennungsraten von über 90% erzielt. Da dieser Teil des Systems keinerlei Training erfordert, die Klassifikation des MS-TDNN lediglich auf der Zustandsebene erfolgt und die Modellierung der Schrift wie das System selbst streng hierarchisch ist, kann das Vokabular ohne Änderungen der Systemparameter oder zusätzliche aufwendige Anpassungsschritte ausgetauscht oder modifiziert werden. Sogar zur Laufzeit des Systems ist durch einfache Erweiterung der Baumstruktur die Änderung des Vokabulars möglich.

Durch Rückkopplung der Baumstruktur kann das System auf die Satzerkennung erweitert werden. Da dann neben Verwechslungen zusätzlich Einfüge- und Auslassungsfehler auftreten, ist die Wortakkuratheit zunächst wesentlich geringer als bei der Einzelworterkennung. Aber ebenso wie beim Schritt von der Einzelzeichen- zur Worterkennung zeigt sich, daß auch hier wiederum syntaktisches Wissen über die zugrundeliegende Sprache helfen kann. Mit **Sprachmodellen** wird der Tatsache Rechnung getragen, daß bestimmte Wortfolgen in der Sprache mit unterschiedlichen Wahrscheinlichkeiten auftreten. Diese Sprachmodelle schätzen die Wahrscheinlichkeit für das Auftreten eines bestimmten Wortes in Abhängigkeit der unmittelbar vorausgehenden ein oder zwei Wörter. Durch den Einsatz der Sprachmodelle erhöht sich die Wortakkuratheit bei der Erkennung von Sätzen von rund 82% auf etwa 86% bei einem Vokabular von 20 000 Wörtern.

Mit Erkennungsraten von über 93% für isolierte Wörter und über 86% für ganze Sätze werden veröffentlichte Erkennungsraten vergleichbarer Systeme um mehr als 5 bzw. 20% übertroffen, wenn außer Betracht bleibt, daß jedes System auf anderen Datenbasen evaluiert wird und somit ein Vergleich schwierig ist. Lediglich für die Erkennung von Einzelzeichen wurden bereits höhere Erkennungsraten publiziert. In diesen Fällen wurde jedoch das jeweilige Erkennungssystem speziell für die Erkennung von Einzelzeichen optimiert und das Training auf weitaus größeren Datenbasen durchgeführt. Die Ergebnisse dieser Arbeit haben gezeigt, daß auch in diesem System durch geeignete Optimierungen für eine bestimmte Erkennungsaufgabe die Leistung weiter gesteigert werden kann. Aber selbst ohne diese gezielten Optimierungen werden mit diesem hybriden System aus neuronalem Netz und Hidden Markov Modellen bei der Wort- und Satzerkennung die derzeit besten publizierten Ergebnisse erreicht.

10.2 Ausblick

Das in dieser Arbeit beschriebene Erkennungssystem hat seine Leistungsfähigkeit nicht nur anhand der Messung auf den drei Datenbasen unter Beweis gestellt, sondern auch durch praktische Anwendung in unterschiedlichen Demonstrationssystemen. Diese Demonstrationssysteme haben aber nicht nur die Leistungsfähigkeit des Handschrifterkenners, sondern auch seine bisherigen praktischen Grenzen aufgezeigt, die sich erst durch die interaktive Benutzung ergeben und in dem System bislang nicht berücksichtigt wurden. Drei wesentliche fehlende Eigenschaften des Systems können dabei ausgemacht werden:

- Robustheit gegenüber Korrekturen
- Schritthaltende Erkennung
- Berücksichtigung des vollständigen Zeichensatzes

Die sich hierdurch ergebende Problematik und Lösungsansätze skizzieren die folgenden Abschnitte. Darüber hinaus werden Erweiterungen vorgeschlagen, die eine weitere Steigerung der Erkennungsrate ermöglichen und die Funktionalität des Erkenners erhöhen.

10.2.1 Robustheit gegenüber Korrekturen

Die in dieser Arbeit verwendeten Daten zur Evaluierung der Erkennungsleistung sind insofern als „saubere" Daten zu bezeichnen, als sie in der Regel keine Schreibfehler oder Korrekturen enthalten. Schreibfehler, wie einzelne fehlende, zusätzliche oder falsche Zeichen innerhalb eines Wortes, können in den meisten Fällen durch die vokabulargesteuerte Suche korrigiert werden und stellen somit ein geringeres Problem als Korrekturen der Eingabe dar. Als Korrekturen werden hierbei alle Aktionen des Schreibers bezeichnet, die einen bereits geschriebenen (aber noch nicht erkannten) Teil der Eingabe nachträglich modifizieren, d.h. die strikte Schreibreihenfolge von Links nach Rechts durchbrechen. Typische Korrekturaktionen sind Durchstreichen (Löschen) und Überschreiben existierender Eingabeteile oder Einfügen zusätzlicher Zeichen. Das Hinzufügen nachträglicher „i"-Punkte oder „t"-Striche kann somit auch als Korrekturaktion, in diesem Fall als Überschreiben, definiert werden.

Eine allgemeine Lösung dieses Problems ist äußerst schwierig. Zunächst muß anhand der Eingabe festgestellt werden, daß ein bestimmter zeitlicher Ausschnitt der Eingabesequenz eine Korrektur und kein normales weiteres Zeichen darstellt. Anschließend muß die Frage beantwortet werden, um welche Korrekturaktion es sich handelt, d.h. ob ein Durchstreichen, Überschreiben oder Einfügen stattgefunden hat, und welche Teile der Eingabeaktion durch diese Aktion betroffen sind. Diese Teile müssen dann je nach Korrekturaktion entweder gelöscht, ersetzt oder ergänzt werden. Einen ersten Versuch, dieses Problem für das hier beschriebene System auf der Ebene der Vorverarbeitung zu lösen, beschreibt [Hür97].

10.2.2 Schritthaltende Erkennung

Die Verwendung der Datenbasen mit bereits gesammelten Schriftproben für die Evaluierung des Systems erlaubt es, bei der Erkennung jeweils ein vollständig geschriebenes Wort bzw. einen ganzen Satz zu betrachten. Für den interaktiven Einsatz eines Handschrifterkenners ist es jedoch sinnvoll, nicht erst nach Vollendung eines Wortes oder Satzes mit der Erkennung zu beginnen, sondern bereits während des Schreibens. So können frühzeitig erste Erkennungshypothesen für den bereits geschriebenen Teil der Eingabe ausgegeben werden, und das Erkennungsergebnis steht ohne größere zeitliche Verzögerungen unmittelbar nach Beendigung des Schreibvorgangs zur Verfügung.

Da das MS-TDNN für die Berechnung der Ausgabevektoren lediglich einen begrenzten Kontext der Eingabesequenz betrachtet und aus der Baumstruktur für jeden Zeitpunkt die bis dahin beste Ausgabehypothese direkt abgelesen werden kann, reduzieren sich die für eine schritthaltende Erkennung erforderlichen Erweiterungen auf die Vorverarbeitung. Nur dort wird bislang eine Annahme darüber gemacht, daß jeweils ein vollständiges Wort oder ganzer Satz vorliegt. Dies betrifft unmittelbar die Bestimmung der Schreiblinien, die umso genauer funktioniert, je länger die Eingabe ist, und dadurch indirekt auch die Neuabtastung, Rotations- und Größennormalisierung. Die bisherige Normalisierung der gesamten Eingabe muß also durch eine abschnittsweise Normalisierung ersetzt werden, die jeweils nur einen begrenzten Teil der Eingabe berücksichtigt. Eine Anpassung ist auch bei den Kontextbitmaps erforderlich, da sie bei der Berechnung für einen bestimmten Punkt in der jetzigen Form alle zukünftigen Punkte mit einbeziehen. Sie müßten dementsprechend so modifiziert werden, daß sie nur noch zeitlich zurückliegende Punkte betrachten. Eine erste Anpassung dieses Systems für schritthaltende Erkennung einzelner Wörter ist in [Gro96] beschrieben. Die schritthaltende Erkennung wird allerdings durch eine niedrigere Erkennungsrate erkauft, da die Normalisierung durch den begrenzten Kontext an Genauigkeit verliert.

10.2.3 Vergrößerung des Zeichensatzes

Aufgrund der zur Verfügung stehenden Datenbasen beschränkt sich das beschriebene System auf die Erkennung von Ziffern, Klein- und Großbuchstaben. Für einen praktischen Einsatz müssen jedoch auch Sonderzeichen und ggf. länderspezifische Zeichen wie Umlaute erkannt werden können. Dies erfordert hier lediglich eine Erweiterung des MS-TDNN um zusätzliche Ausgabeklassen und ein Neutraining mit entsprechenden Daten, da an allen anderen Stellen des Systems keine Annahmen über die Größe des zugrundeliegenden Zeichensatzes gemacht werden. Insgesamt ist allerdings eine leichte Verschlechterung der Erkennungsrate zu erwarten, da durch die Erhöhung der Ausgabeklassen die Verwechslungsgefahr steigt und so die Lernaufgabe erschwert wird.

Anstatt lediglich die Größe des Zeichensatzes zu ändern, ist auch ein vollständiger Austausch des Zeichensatzes denkbar, z.B. für Russisch, Griechisch, Arabisch, Chinesisch, Japanisch oder Koreanisch. Während für Russisch, Griechisch und Arabisch

durch die ähnliche Struktur der einzelnen Zeichen lediglich geringfügige Änderungen des Systems vonnöten wären, sind für asiatische Sprachen größere Änderungen erforderlich. Dies betrifft sowohl die Vorverarbeitung, da asiatische Zeichen z.B. keine Unter- und Oberlängen und damit auch keine Schreiblinien haben, als auch die Modellierung der Zeichen und Wörter durch abstrakte Zustände. Ein Zeichen dort entspricht von der Komplexität her eher einem Wort in unserem Sinne. Wie anstelle getrennter Erkenner für unterschiedliche Zeichensätze ein multi-lingualer Erkenner, z.B. durch hierarchische Anordnung von HMMs, entwickelt werden kann, ist in [LK96] dargestellt.

Auch die Eingabe mathematischer Formeln erfordert die Erkennung spezieller Symbole, die als Erweiterung des Zeichensatzes betrachtet werden können. Hier kommt jedoch noch ein schwierigeres Segmentierungsproblem hinzu, da Formeln in der Regel nicht mehr nur von links nach rechts erzeugt werden, sondern eine komplexere geometrische Anordnung der Symbole vorliegt. Darüber hinaus bekommt in diesem Fall auch die Größe der Zeichen eine Bedeutung, die zu der Syntaktischen Struktur der Formel beiträgt. Ein erster Ansatz zur Erkennung mathematischer Formeln, der auf dem hier beschriebenen Handschrifterkenner aufbaut, ist in [SH97] beschrieben. Weitere Ansätze zur Formelerkennung sind z.B. in [Win96, WL97] zu finden.

10.2.4 Kombination unterschiedlicher Erkenner

Die Ergebnisse in Kapitel 9.2.6 haben gezeigt, daß bei nahezu 50% aller Fehlerkennungen des Systems die korrekte Antwort sich bereits unter den drei besten Hypothesen des Systems befindet. Dies und die Beobachtung, daß bei Fehlerkennungen die Bewertungen der Wörter in der N-Besten-Liste sehr dicht beieinander liegen, das System also „unsicher" über seine Entscheidung ist, führt zu der Idee, vor der Ausgabe der endgültigen Ausgabehypothese eine Neubewertung der Wörter in der N-Besten Liste durch einen zweiten Erkenner durchzuführen. So wird entweder die Entscheidung des ersten Erkenners bestätigt oder aber die Entscheidung entsprechend dem Ergebnis des zweiten Erkenners geändert. Idealerweise sollte der zweite Erkenner seine Entscheidung nach möglichst vom ersten Erkenner verschiedenen Merkmalen treffen, um zu vermeiden, daß er dieselben Fehler wie der erste Erkenner macht. Der zweite Erkenner könnte z.B. ein Off-line Erkenner sein, der seine Entscheidung anhand einer Grauwertbild-Repräsentation der Eingabe durchführt. Ein erstes Experiment mit dem in [Bur97] vorgestellten Off-line Erkenner hat gezeigt, daß auf diese einfache Art und Weise der Neubewertung bereits eine Steigerung der Erkennungsrate bei Einzelworten um 1% möglich ist. Eine komplexere Technik der Kombination mehrerer Erkenner ist das z.B. in [DSS93] beschriebene „Boosting", das aber eine weitaus größere Trainingsmenge erfordert. Eine theoretische Betrachtung der Kombination mehrerer Klassifikatoren zur Verbesserung der Erkennungsraten enthält [Kit96]. Weitere praktische Realisierungen der Erkennerkombination für die Handschrifterkennung sind in [Gor96, YNT+96, LBK96] zu finden.

10.2.5 Modellierung von Schreibvarianten

Eine weitere Möglichkeit, die Erkennungsleistung zu verbessern, ist die getrennte Behandlung der Schreibstile Kursiv- und Druckschrift, um die Erkennungsaufgabe zu vereinfachen. Zu diesem Zweck können entweder zwei Erkenner jeweils für einen speziellen Schreibstil trainiert werden wie in den Experimenten in Kapitel 9.2.2 oder aber die Schreibstile in einem einzigen Erkenner getrennt modelliert werden. Im ersten Fall müßte dann der Schreibstil vor der Erkennung bekannt sein oder aber automatisch bestimmt werden. Erste Ansätze für eine automatische Bestimmung des Schreibstils sind in [VS96] zu finden. Zusätzlich ist auch eine kontext-abhängige Modellierung der einzelnen Zeichen wie in [KRR97] möglich, die auch in der Spracherkennung zur Kompensierung von Koartikulationseffekten eingesetzt wird.

10.2.6 Der UNIPEN Benchmark

Neben weiteren Verbesserungen der Erkennungsrate und einer Erhöhung der Funktionalität des Erkenners ist der Vergleich mit anderen Systemen bei internationalen Benchmarks eines der wichtigsten Ziele. Für die Spracherkennung und Off-line Handschrifterkennung werden solche Benchmarks auf standardisierten Datenbasen bereits seit einigen Jahren regelmäßig durchgeführt, um jeweils den aktuellen „State-of-the-art" zu bestimmen. Für die On-line Handschrifterkennung ist ein erster Benchmark dieser Art erst in der nahen Zukunft im Rahmen des UNIPEN Projekts zu erwarten.

Das Projekt UNIPEN [GSP+94] wurde 1993 vom Technical Committee 11 der International Association for Pattern Recognition (IAPR) ins Leben gerufen, um dem Mangel einer standardisierten Datenbasis und dem Fehlen internationaler Vergleiche entgegenzuwirken. Diesem Projekt haben sich über 40 internationale Teilnehmer aus Forschung und Industrie angeschlossen. Jeder Teilnehmer mußte als Beitrittsbedingung eine festgelegte Mindestmenge (mind. 12 000 Zeichen) On-line Handschriftdaten sammeln und für das Projekt frei zur Verfügung stellen. Insgesamt wurde dadurch eine Datenbasis mit mehr als 5 Millionen Zeichen (in Form von Einzelzeichen, Wörtern oder Sätzen) zusammengestellt. Derzeit wird diese Datenbasis vom National Institute of Standards and Technology (NIST) in 4 disjunkte Mengen unterteilt (Trainingsmenge, Kreuzvalidierungsmenge und 2 Testmengen), um zwei internationale Vergleichswettbewerbe durchzuführen. Die Trainings- und Kreuzvalidierungsmenge werden allen Teilnehmern gleichermaßen zur Entwicklung der Erkenner zur Verfügung gestellt. Die beiden Testmengen werden vom NIST unter Verschluß gehalten, um im Abstand von ca. einem Jahr zwei internationale Vergleiche der Erkenner zu organisieren. Erst durch diesen Vergleich wird eine zuverlässige Aussage über den derzeitigen Leistungsstand heutiger On-line Handschrifterkennungssysteme möglich.

Literaturverzeichnis

[AA93] I. S. I. Abuhaiba and P. Ahmed. Restoration of Temporal Information in Off-line Arabic Handwriting. *Pattern Recognition*, 26(7):1009–1017, July 1993.

[AM91] S. Abramowski and H. Müller. *Geometrisches Modellieren.* Reihe Informatik, Band 75. BI Wissenschaftsverlag, Mannheim/Wien/Zürich, 1991. Hrsg. von K.H. Böhling, U. Kulisch und H. Maurer.

[BBNN94] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo, and K. S. Nathan. A Fast Statistical Mixture Algorithm for On-Line Handwriting Recognition. *IEEE Transactions on Patter Analysis and Machine Intelligence*, 16(12):1227–1233, December 1994.

[BCCM93] G. Boccignone, A. Chianese, L. P. Cordella, and A. Marcelli. Recovering Dynamic Information from Static Handwriting. *Pattern Recognition*, 26(3):409–418, March 1993.

[Bei96] H. S. M. Beigi. Pre-Processing the Dynamics of On-line Handwriting Data, Feature Extraction and Recognition. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[BH95] U. Bodenhausen and H. Hild. Automatic Construction of Neural Networks for Special Purpose Speech Recognition Systems. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Detroit, May 1995.

[BL93] S. Bercu and G. Lorette. On-line Handwritten Word Recognition: An Approach Based on Hidden Markov Models. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Buffalo, USA, May 1993.

[BM93] U. Bodenhausen and S. Manke. Automatically Structured Neural Networks for Handrwitten Character and Word Recognition. In *Proceedings of the International Conference on Neural Networks*, San Francisco, March 1993.

[BMW93] U. Bodenhausen, S. Manke, and A. Waibel. Connectionist Architectural Learning for High Performance Character and Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, April 1993.

[Bur97] T. Burkhardt. Methoden der Merkmalsgenerierung bei Off-line Handschriften. Studienarbeit, Universität Karlsruhe, Fakultät für Informatik, April 1997.

[DHU97] J. G. A. Dolfing and R. Haeb-Umbach. Signal Representations for Hidden Markov Model Based On-line Handwriting Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, 1997.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-Likelihood from Incomplete Data via the EM Algorithm. *Journal of Royal Statistical Society*, 39:1-38, 1977.

[DR95] D. S. Doerman and A. Rosenfeld. Recovery of Temporal Information from Static Images of Handwriting. *International Journal of Computer Vision*, 15:143–164, 1995.

[DSS93] H. Drucker, R. Schapire, and P. Simard. Boosting Performance in Neural Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705–720, 1993.

[DW92] C. E. Dunn and P. S. P. Wang. Character Segmentation Techniques for Handwritten Text — A Survey. In *Proceedings of the International Conference on Pattern Recognition*, pages 577–582, 1992.

[Ear62] L. D. Earnest. Machine Recognition of Cursive Writing. In C. Cherry, editor, *Information Processing*, pages 462–466. Butterworth, London, 1962.

[EK75] R. W. Ehrich and K. J. Koehler. Experiments in the Contextual Recognition of Cursive Script. *IEEE Transactions on Computers*, 24:182–194, 1975.

[Elm96] A. J. Elms. *The Representation and Recognition of Text Using Hidden Markov Models*. PhD thesis, Department of Electronic and Electrical Engineering, University of Surrey, April 1996.

[Far79] R. F. H. Farag. Word Level Recognition of Cursive Script. *IEEE Transactions on Computers*, 28:172–175, 1979.

[FH61] L. S. Frishkopf and L. D. Harmon. Machine Reading of Cursive Script. In *4th London Symposium on Information Theory*, pages 300–316, 1961.

[FM97] J. Flanagan and I. Marsic. Issues in Measuring the Benefits of Multimodal Interfaces. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[FNCB93] T. Fujisaki, K. Nathan, W. Cho, and H. Beigi. On-line Unconstrained Handwriting Recognition by a Probabilistic Method. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Buffalo, USA, May 1993.

[Fre61] H. Freeman. On the Encoding of Arbitrary Geometric Configurations. *IRE Transactions on Electronic Computers*, EC 10:260–281, 1961.

[FS93] N. S. Flann and S. Shekhar. Recognizing On-line Cursive Handwriting Using a Mixture of Cooperating Pyramid-Style Neural Networks. In *World Congress on Neural Networks*, Oregon, 1993.

[GAL+91] I. Guyon, P. Albrecht, Y. LeCun, J. Denker, and W. Hubbard. Design of A Neural Network Character Recognizer for A Touch Terminal. *Pattern Recognition*, 24(2):105–119, February 1991.

[GHA+92] I. Guyon, D. Henderson, P. Albrecht, Y. LeCun, and J. Denker. Writer Independent and Writer Adaptive Neural Network for On-line Character Recognition. In S. Impedovo and J.C. Simon, editors, *From Pixels to Features III: Frontiers in Handwriting Recognition*. Elsevier Science Publishers, 1992.

[Gor96] N. Gorski. Practical Combination of Multiple Classifiers. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[GP93] W. Guerfali and R. Plamondon. Normalizing and Restoring On-line Handwriting. *Pattern Recognition*, 26(3):419–431, March 1993.

[Gro96] R. Groß. Inkrementelle Vorverarbeitung und Erkennung für die On-Line Handschrifterkennung. Studienarbeit, Universität Karlsruhe, Fakultät für Informatik, 1996.

[GS90] V. K. Govindan and A. P. Shivaprasad. Character Recognition — A Review. *Pattern Recognition*, 23(7):671–683, July 1990.

[GSP+94] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Stanet. UNIPEN Project of On-line Data Exchange and Recognizer Benchmarks. In *Proceedings of the International Conference on Pattern Recognition*, pages 29–31, Jerusalem, October 1994.

[GWS92] V. Govindaraju, D. Wang, and S. N. Srihari. Using Temporal Information in Off-Line Word Recognition. *USPS 5th Advanced Technology Conference*, November 1992.

[Har72] L. D. Harmon. Automatic Recognition of Print and Script. *Proceedings of the IEEE*, 60(10), 1972.

[HBT96] J. Hu, M. K. Brown, and W. Turin. HMM Based On-Line Handwriting Recognition. *IEEE Transactions on Patter Analysis and Machine Intelligence*, 18(10):1039–1045, October 1996.

[HFW91] P. Haffner, M. Franzini, and A. Waibel. Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, May 1991.

[Hil97] H. Hild. *Buchstabiererkennung mit neuronalen Netzen in Auskunftssystemen*. PhD thesis, Universität Karlsruhe, Fakultät für Informatik, Mai 1997.

[HOKK93] J. Y. Ha, S. C. Oh, J. H. Kim, and Y. B. Kwon. Unconstrained Handwritten Word Recognition with Interconnected Hidden Markov Models. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Buffalo, USA, 1993.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multi-layer Feed Forward Networks Are Universal Approximators. *Neural Networks*, 2:359–368, 1989.

[Hür97] W. Hürst. Repair in Off-Line Handwriting Recognition. Diplomarbeit, Carnegie Mellon University, School of Computer Science, March 1997.

[HW92] P. Haffner and A. Waibel. Multi-State Time Delay Neural Networks for Continuous Speech Recogniton. In *Advances in Neural Information Processing Systems*. Morgan Kaufmann, Volume 4, April, 1992.

[IP90] J. B. Hampshire II and B. Pearlmutter. Equivalence Proofs for Multilayer Perceptron Classifiers and the Bayesian Discriminant Function. In D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G.E. Hinton, editors, *Proc. of the 1990 Connectionist Models Summer School*. Morgan Kaufmann, San Mateo, CA, April 1990.

[KAM+94] F. Kubala, A. Anastasakos, J. Makhoul, L. Nguyen, R. Schwartz, and G. Zavaliagkos. Comparative Experiments on Large Vocabulary Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, April 1994.

[Kas95] R. H. Kassel. *A Comparison of Approaches to On-Line Handwritten Character Recognition*. PhD thesis, Massachusetts Institute of Technology, June 1995.

[Kit96] J. Kittler. Improving Recognition Rates by Classifier Combination: A Theoretical Framework. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[Koh88] T. Kohonen. *Self-Organization and Associative Memory*. Springer Verlag, New York, second edition edition, 1988.

[KR89] M. Kadirkamanathan and P. J. W. Rayner. A Unified Approach to Online Cursive Script Segmentation and Feature Extraction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1989.

[KR91] J. Keeler and D. E. Rummelhart. A Self-Organizing Integrated Segmentation And Recognition Neural Net. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*. Morgan Kaufman, volume 4, Denver, 1991.

[KRR97] A. Kosmala, J. Rottland, and G. Rigoll. An Investigation of the Use of Trigraphs for Large Vocabulary Cursive Handwriting Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, 1997.

[LB93] E. Lecolinet and O. Baret. Cursive Word Recognition: Methods and Strategies. In *Fundamentals in Handwriting Recognition*, Bonas, July 1993. NATO-ASI Workshop.

[LB94] Y. LeCun and Y. Bengio. Word-Level Training of a Handwritten Word Recognizer Based on Convolutional Neural Networks. In *Proceedings of the International Conference on Pattern Recognition*, pages 88–92, Jerusalem, October 1994.

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989.

[LBK96] R. Lindwurm, T. Breuer, and K. Kreuzer. Multi Expert System for Handprint Recognition. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[LK96] J. J. Lee and J. H. Kim. A Unified Network-Based Approach for On-Line Recognition of Multi-Lingual Cursive Handwriting. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[LWH90] K. J. Lang, A. Waibel, and G. E. Hinton. A Time Delay Neural Network Architecture for Isolated Word Recognition. *Neural Networks*, 3(1):23–43, 1990.

[MB94] S. Manke and U. Bodenhausen. A Connectionist Recognizer for On-Line Cursive Handwriting Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 596–598, Adelaide, April 1994.

[MBLD92] O. Matan, C. J. C. Burges, Y. LeCun, and J. S. Denker. Multi-Digit Recognition Using A Space Displacement Neural Network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*. Morgan Kaufman, volume 4, Denver, 1992.

[MBPV93] P. Morasso, L. Barberis, S. Pagliano, and D. Vergano. Recognition Experiments of Cursive Dynamic Handwriting with Self-Organizing Networks. *Pattern Recognition*, 26(3):451–460, March 1993.

[ME64] P. Mermelstein and M. Eden. Experiments on Computer Recognition of Connected Handwritten Words. *Information and Control*, 7:255–270, 1964.

[Mey94] A. Meyer. Pen Computing – A Technology Overview and a Vision. Semesterarbeit in Informatik, Multi Media Laboratory, Department of Computer Science, University of Zürich, July 1994.

[MFW94] S. Manke, M. Finke, and A. Waibel. Combining Bitmaps with Dynamic Writing Information for On-Line Handwriting Recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 596–598, Jerusalem, October 1994.

[MFW95a] S. Manke, M. Finke, and A. Waibel. NPen++: A Writer Independent, Large Vocabulary On-Line Cursive Handwriting Recognition System. In *Proceedings of the International Conference on Document Analysis and Recognition*, Montreal, August 1995.

[MFW95b] S. Manke, M. Finke, and A. Waibel. The Use of Dynamic Writing Information in a Connectionist On-Line Cursive Handwriting Recognition System. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge MA, 1995.

[MFW96] S. Manke, M. Finke, and A. Waibel. A Fast Search Technique for Large Vocabulary On-Line Handwriting Recognition. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[Mor89] P. Morasso. Neural Models for Cursive Script Handwriting. In *IEEE Intl. Conference on Neural Networks*, pages 539–542, 1989. Volume 2.

[MP43] W. S. McCulloch and W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[MP69] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[MSSC94] J. Makhoul, T. Starner, R. Schwartz, and G. Chou. On-line Cursive Handwriting Recognition Using Hidden Markov Models and Statistical Grammars. In *Proceedings of the Human Technology Workshop*, pages 432–435, Plainsboro, New Jersey, 1994.

[NBNB93] K. S. Nathan, J. R. Bellegarda, D. Nahamoo, and E.J. Bellegarda. On-line Handwriting Recognition Using Continuous Parameter Hidden Markov Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, April 1993.

[NP90] F. Nouboud and R. Plamondon. On-Line Recognition of Handprinted Characters: Survey and Beta Tests. *Pattern Recognition*, 23(9):1031–1040, September 1990.

[NWF86] R. Nag, K. H. Wong, and F. Fallside. Script Recognition Using Hidden Markov Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo, Japan, April 1986.

[ONA97] S. Ortmanns, H. Ney, and X. Aubert. A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition. *Computer Speech and Language*, 11:43–72, 1997.

[PL89] R. Plamondon and G. Lorette. Automatic Signature Verification and Writer Identification – State of the Art. *Pattern Recognition*, 22(2):107–131, February 1989.

[Rab89] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285, February, 1989.

[Rao95] P. V. S. Rao. A Knowledge-Based Approach for Script Recognition without Training. *IEEE Transactions on Patter Analysis and Machine Intelligence*, 17(12):1233–1239, December 1995.

[RHW86] D. E. Rummelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323(9):533–536, October 1986.

[RL91] M. D. Richard and R. P. Lippmann. Neural Network Classifiers Estimate Bayesian A Posteriori Probabilities. *Neural Computation*, 3(4):461–483, 1991.

[RNM96] E. H. Ratzlaff, K. S. Nathan, and H. Maruyama. Search Issues in the IBM Large Vocabulary Unconstrained Handwriting Recognizer. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[Ros58] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organisation in the Brain. *Psychological Review*, 65:386–408, 1958.

[Ros62] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, Washington, DC, 1962.

[SBM80] C. Y. Suen, M. Berthod, and S. Mori. Automatic Recognition of Handprinted Characters — The State of the Art. *Proceedings of the IEEE*, 68(4), 1980.

[Sch93] L. Schomaker. Using Stroke- or Character-Based Self-Organizing Maps in the Recognition of On-line, Connected Cursive Script. *Pattern Recognition*, 26(3):443–450, March 1993.

[Sch94] L. Schomaker. User-interface Aspects in Recognizing Connected-Cursive Handwriting. In *Proceedings of the IEEE Colloquium on Handwriting and Pen-based Input*, London, March 1994.

[Sch95] M. E. Schenkel. *Handwriting Recognition Using Neural Networks and Hidden Markov Models*, volume 45 of *Series in Microelectronics*. Hartung-Gorre, Konstanz, 1995.

[Sen95] G. Seni. *Large Vocabulary Recognition of On-line Handwritten Cursive Words*. PhD thesis, Department of Computer Science of the State University of New York at Buffalo, August 1995.

[SGH94] M. Schenkel, I. Guyon, and D. Henderson. On-Line Cursive Script Recognition Using Time Delay Neural Networks and Hidden Markov Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, April 1994.

[SH97] A. Schulz-Heyn. Maschinelle Erkennung von handgeschriebenen, mathematischen Ausdrücken. Diplomarbeit, Universität Karlsruhe, Fakultät für Informatik, June 1997.

[SMSC94] T. Starner, J. Makhoul, R. Schwartz, and G. Chou. On-line Cursive Handwriting Recognition Using Speech Recognition Methods. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, April 1994.

[SN97] A. Senior and K. Nathan. Writer Adaptation of a HMM Handwriting Recognition System. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, 1997.

[SNBK93] R. K. Srihari, S. Ng, C. M. Baltus, and J. Kud. Use of Language Models in On-line Sentence/Phrase Recognition. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Buffalo, USA, May 1993.

[SSE96] R. Seiler, M. Schenkel, and F. Eggiman. Cursive Handwriting Recognition: Off-line versus On-line Recognition. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Coulchester, England, September 1996.

[ST92] L. R. B. Schomaker and H. L. Teulings. Stroke- versus Character-based Recognition of On-line, Connected Cursive Script. In S. Impedovo and J.C. Simon, editors, *From Pixels to Features III*. North-Holland, 1992.

[TS92] H. L. Teulings and L. R. B. Schomaker. Leraning Prototypes in Cursive Handwriting. In S. Impedovo and J.C. Simon, editors, *From Pixels to Features III*. North-Holland, 1992.

[TSW90] C. C. Tappert, C. Y. Suen, and T. Wakahara. The State of the Art in On-Line Handwriting Recognition. *IEEE Transactions on Patter Analysis and Machine Intelligence*, 12(8):787–808, 1990.

[VS96] L. Vuurpijl and L. Schomaker. Coarse Writing-Style Clustering Based on Simple Stroke-Related Features. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.

[WF74] R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *Journal of the ACM*, 21(1):168–178, January 1974.

[WHH+87] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme Recognition Using Time-Delay Neural Networks. Technical report, Technical Report TR-1-0006, ATR Interpreting Telephony Research Laboratories, October 1987.

[WHH+89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, March 1989.

[Win96] H. J. Winkler. HMM-Based Handwritten Symbol Recognition Using On-line and Off-line Features. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Atlanta, May 1996.

[WL90] A. Waibel and K. F. Lee. *Readings in Speech Recognition.* Morgan Kaufmann, San Mateo, CA, 1990.

[WL97] H. J. Winkler and M. Lang. On-line Symbol Segmentation and Recognition in Handwritten Mathematical Expressions. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, 1997.

[WSG+94] H. Weissman, M. Schenkel, I. Guyon, C. Nohl, and D. Henderson. Recognition-Based Segmentation of On-Line Run-On Handprinted Words: Input vs. Output Segmentation. *Pattern Recognition*, 27(3):405–420, March 1994.

[WSVY97] A. Waibel, B. Suhm, M.T. Vo, and J. Yang. Multimodal Interfaces for Multimedia Information Agents. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[WVDM96] A. Waibel, M. T. Vo, P. Duchnowski, and S. Manke. Multimodal Interfaces. *Artificial Intelligence Review*, 10:299–319, 1996.

[YNT+96] S. Yamaguchi, K. Nagata, T. Tsutsumida, F. Kimura, and A. Iwata. Study on Multi-Expert Systems for Handprinted Numeral Recognition. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Colchester, England, September 1996.