Studienarbeit „Online Handschrifterkennung japanischer Kanazeichen“

Andreas Hanemann

Fakultät für Informatik

Universität Karlsruhe

Institut für Logik, Komplexität und Deduktionssysteme

Prof. Dr. A. Waibel

betreut von Dipl-Inf. Jürgen Reichert und Dr. Stefan Jäger

März 2001

Kurzfassung

Ausgehend von einem Online Handschrifterkenner für englische Wörter, der auf einer Kombination neuronaler Netze mit statistischen Modellen beruht, wurde in dieser Studienarbeit ein schreiberunabhängiger japanischer Handschrifterkenner entwickelt. Dieser Erkenner kann für die Einzelzeichenerkennung von 146 Kanazeichen verwendet werden, die die im Japanischen gebräuchlichen Silben repräsentieren. Ebenfalls ist es möglich, Kanafolgen zu erkennen. Dabei wurde bei Verwendung einer Datenbasis mit 28 Schreibern für den Einzelzeichenerkenner eine Erkennungsrate von 92,9% erzielt, während für den Kanafolgenerkenner eine Rate von 89,3% erreicht wurde. Abschließend wurden zwei weitere Einzelzeichenerkenner für zwei Untermengen der Kanas (Hiragana und Katakana) entwickelt, die Erkennungsraten von 96,4% bzw. 97,0% erreichten.

Danksagung

An dieser Stelle möchte ich mich bei verschiedenen Personen bedanken, deren Unterstützung mir bei der Erstellung dieser Studienarbeit sehr geholfen hat. Mein erster Dank gilt Dr. Stefan Manke, dessen Dissertation „On-line Erkennung kursiver Handschrift bei großen Vokabularen“ die Grundlage dieser Arbeit bildet. Desweiteren möchte ich mich bei Dr. Stefan Jäger und Jürgen Reichert, die diese Studienarbeit betreut haben, und bei Roald Wolff, der mich bei der Einarbeitung in das Erkennungssystem unterstützt hat, für ihre Hilfe bei der Verwendung und Weiterentwicklung des Handschrifterkenners bedanken. Für die Betreuung und Begutachtung der Studienarbeit gilt mein besonderer Dank Prof. Alex Waibel.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Handschrifterkennung

Nachdem in den sechziger Jahren bereits erste Versuche mit Online Handschrifterkennung durchgeführt, aber wegen zu geringer technischer Möglichkeiten nicht weiterverfolgt wurden, ist seit Beginn der neunziger Jahre ein stark wachsendes Interesse aus der Wirtschaft und von Forschungseinrichtungen zu verzeichnen. Bei dieser Art der Handschrifterkennung wird versucht, die Handschrift während des Schreibens laufend aufzuzeichnen, um das Erkannte kurz nach dem Schreiben auszugeben. Das steigende Interesse war nicht zuletzt durch die Entwicklung kleiner tragbarer Computer begünstigt, bei denen sich der Stift als mögliches Eingabegerät anbot. Außerdem kann die Verwendung eines Stiftes dazu dienen, Menschen ohne große Erfahrung mit Computern die Scheu vor dem Umgang mit diesen zu nehmen. Die Verwendung ist ebenfalls bei sogenannten multimodalen Benutzerschnittstellen möglich, bei denen der Benutzer verschiedene Eingabemedien (z.B. Sprache, Tastatur, Maus, Handschrift) zur Verfügung hat, um bei unterschiedlichen Anwendungen möglichst effizient zu arbeiten. Speziell in Japan und Ostasien ist der Einsatz der Handschrifterkennung sinnvoll, da aufgrund der großen Zeichensätze nur ein geringer Teil der Zeichen auf Computertastaturen untergebracht werden kann. Die Eingabe der übrigen Zeichen erfolgt dann über komplizierte Eingabemethoden. Deshalb wäre es günstiger, wenn die Zeichen z.B. mit Hilfe eines Grafiktabletts direkt eingegeben werden könnten.

In den folgenden Unterkapiteln wird auf den Unterschied zwischen Offline und Online Handschifterkennung eingegangen und anschließend werden die wesentlichen Merkmale, die zur Klassifizierung eines Handschifterkenners dienen, vorgestellt.

1.1.1 Offline Erkennung

Bei der *Offline Erkennung* werden Texte, die typischerweise bereits auf Papier gedruckt oder geschrieben wurden, mittels optischer Abtastung als z.B. Grauwertbilder in den Rechner eingelesen, um anschließend erkannt zu werden. Eine weitverbreitete Form ist dabei das sogenannte *OCR (Optical Character Recognition)*, bei dem gedruckte Texte eingelesen werden. Dadurch daß bei der Offline Erkennung nur das Ergebnis eines Schreibvorganges betrachtet wird, fehlen zeitliche Informationen, was die Erkennung in den meisten Fällen erschwert.

1.1.2 Online Erkennung

Im Gegensatz zur Offline Erkennung stehen bei der *Online Erkennung* auch zeitliche Informationen über den Schreibvorgang zur Verfügung, so daß es im allgemeinen nicht empfehlenswert ist, Onlinedaten in entsprechende Offlinedaten umzuwandeln und dann die Erkennung mit Offlinealgorithmen vorzunehmen.

Ein Vorteil der zeitlichen Informationen ist, daß Separationsprobleme zwischen einzelnen Buchstaben, die durch räumliche Überlappungen verursacht werden, einfacher aufgelöst werden können.

Manchmal kann es jedoch auch zu Problemen durch die zusätzlichen dynamischen Informationen kommen, wenn z.B. ein Zeichen wie der Großbuchstabe E mit unterschiedlicher Reihenfolge der Striche geschrieben wird. Bei der Offline Erkennung macht dieses keinen Unterschied, da die Ergebnisse der unterschiedlichen Schreibweisen nicht zu großen Abweichungen führen. Bei der Online Erkennung einer für den Erkenner neuen Schreibvariante kann es jedoch zu Fehlerkennungen kommen. Ein verwandtes Problem ergibt sich z.B. auch bei t-Strichen und i-Punkten, die häufig nicht sofort beim Schreiben des Buchstabens, sondern erst am Ende des Wortes geschrieben werden. Auch hier gibt es ohne zeitliche Informationen keine Probleme, da sich die verzögerten Striche am richtigen Ort befinden, bei Online Erkennern muß dieses jedoch berücksichtigt werden.

Insgesamt bringen die zeitlichen Informationen jedoch einen erheblichen Vorteil bei der Online Erkennung, was auch durch spezielle Untersuchungen [2] nachgewiesen werden konnte.

1.1.3 Merkmale eines Handschrifterkenners

Beim Vergleich der Erkennungsraten unterschiedlicher Online Handschrifterkenner müssen die Einschränkungen, die bei der Eingabe der zu untersuchenden Zeichen oder Wörter gemacht wurden, berücksichtigt werden. Je mehr Einschränkungen nämlich für die Eingabe gemacht werden, um so mehr kann die Erkennungsrate verbessert werden. Folgende Kriterien sind zu berücksichtigen:

Art des Erkenners: Bei der Art des Erkenners wird unterschieden, ob der Erkenner zur Einzelzeichenerkennung, zur Einzelworterkennung oder zur Satzerkennung verwendet werden soll.

Bei der Einzelzeichenerkennung tritt das Problem der Segmentierung, d.h. der Zuordnung von einzelnen Strichen zu verschiedenen Zeichen, nicht auf, da die Zeichen alle einzeln erkannt werden. Ein wichtiges Kriterium ist hierbei, wieviele Zeichen insgesamt unterschieden werden müssen und wie stark sich die Zeichen unterscheiden.

Bei der Worterkennung soll eine Kette von Einzelzeichen eingelesen werden. Dabei tritt nun das Segmentierungsproblem auf, da das Wort in die Einzelzeichen zerlegt werden soll. Bei der Worterkennung bedient man sich häufig eines Wörterbuchs, das Wörter aus der verwendeten Sprache enthält, um bei Wörtern mit gut und schwer lesbaren Teilen doch noch das richtige Wort zu erkennen. Als Ergebnis wird immer ein Wort aus dem Wörterbuch ausgegeben, das der Eingabe am ähnlichsten ist. Dieses führt dazu, daß Wörter, die nicht im Wörterbuch enthalten sind, auf jeden Fall falsch erkannt werden. Ein wichtiges Kriterium ist hierbei die Vokabulargröße, die nicht zu klein sein darf, da ansonsten bei der Anwendung des Erkenners häufig Fehler durch nicht im Vokabular enthaltene Wörter entstehen. Auf der anderen Seite sinkt die Erkennungsrate für die Wörter aus dem Wörterbuch, je größer das Vokabular ist, da es mehr Auswahl- und damit auch Verwechselungsmöglichkeiten gibt.

Bei der Satzerkennung gibt es zusätzlich zu den Problemen der Worterkennung noch das Problem der Segmentierung der Eingabe in verschiedene Wörter, d.h. dem Auffinden von Wortgrenzen. Ergänzend zu dem Vokabular werden hierbei Sprachmodelle der zugrundeliegenden Sprache verwendet.

Schreibstile: Der Schreibstil beeinflußt in großem Umfang die Schwierigkeit der Handschrifterkennung und damit die Erkennungsrate. Hierbei wird unterschieden zwischen Druckschrift, Kursivschrift und Mischformen zwischen Druck- und Kursivschrift. Je mehr Einschränkungen gemacht werden, z.B. nur ein bestimmter Schreibstil und Schreiben in vorgegebene Kästchen, desto leichter sind die Zeichen zu unterscheiden.

Schreiberabhängigkeit: Ein schreiberabhängiges System wird speziell für einen Schreiber eingestellt und wird im allgemeinen besser sein als ein schreiberunabhängiges System, das für beliebige Schreiber geeignet sein soll. Beim schreiberunabhängigen System müssen möglichst viele Schreibvarianten dem System bekannt sein, damit nicht erst eine Anpassung für den speziellen Schreiber erforderlich ist. Eine Möglichkeit die Vorteile dieser beiden Varianten zu verbinden, ist die sogenannte

Adaption [3], bei der sich das System auf den jeweiligen Benutzer einstellt, während dieser das System verwendet. Das System lernt dabei aus Fehlern, die vom Benutzer korrigiert werden.

Länderspezifische Besonderheiten: Manchmal muß bei schreiberunabhängigen Systemen auf unterschiedliche Schreibweisen in verschiedenen Ländern geachtet werden. So wird beispielsweise die Eins in den USA nur durch einen senkrechten Strich geschrieben und die Sieben ohne den in Deutschland verwendeten Mittelstrich. Deshalb könnte es zu Verwechselungen zwischen einer deutschen Eins und einer amerikanischen Sieben kommen.

Verwendete Hardware: Für die Eingabe der Handschrift können verschiedene Arten von Hardware verwendet werden, insbesondere Grafiktabletts, drucksensitive Bildschirme oder LCD-Tabletts.

1.2 Übersicht über die japanischen Schriftzeichen

Im Japanischen werden unterschiedliche Schriftsysteme nebeneinander verwendet (siehe [4]):

Kanji: Kanji sind aus China stammende Schriftzeichen oder Ideogramme, die jeweils einen Begriffsinhalt wiedergeben. Die meisten Kanji haben mindestens zwei verschiedene Lesungen. Die sogenannte On-Lesung gibt die dem Chinesischen nachempfundene japanische Aussprache wieder, die Kun-Lesung ist die rein japanische Aussprache für ein Kanji. Ein Kanji besteht aus mehreren Komponenten, von denen die Hauptkomponente als Radikal bezeichnet wird und die Aussprache des Kanji bestimmt. Es gibt insgesamt mehr als 50.000 Kanji, von denen das japanische Bildungsministerium 1945 als wissenswert einstufte und für die Schulen als Lerninhalt vorgab. Für die Suche nach einem speziellen Kanji gibt es besondere Kanjilexika, die die Kanji nach den 214 Radikalen und der Strichanzahl geordnet auflisten.

Die Verwendung der chinesischen Zeichen für die japanische Schrift ist historisch zu erklären: Im 3. Jahrhundert nach Christus begannen die Japaner, die bis dahin noch kein eigenes Schriftsystem entwickelt hatten, die Kanji von den Chinesen zu übernehmen, da diese bereits über ein im Laufe von 2000 Jahren entwickeltes Schriftsystem verfügten. Da die Aussprache des Japanischen sich deutlich vom Chinesischen unterscheidet, entwickelte sich zunächst eine japanisch gefärbte Aussprache der Kanji (On-Lesung), bis sich parallel dazu eine japanische Lesung (Kun-Lesung) etablierte.

Hiragana: Hiragana ist eine phonetische Silbenschrift, die keine geraden Striche enthält. Sie besteht aus 46 Grundzeichen und wird heute für

ursprünglich japanische Begriffe verwendet. Dabei kann jedes Kanji durch (mindestens) eine Folge von Hiragana dargestellt werden. Die Silbenschrift, die im 9. Jahrhundert vermutlich von Mönchen entwickelt wurde, galt lange Zeit als Frauenschrift und wurde vor allem von Damen des Hochadels am Kaiserhof von Kyoto verwendet.

Katakana: Katakana ist ebenfalls eine phonetische Silbenschrift, die jedoch ausschließlich aus geraden Strichen besteht. Sie wird heute für Fachworte und ausländische Begriffe verwendet. Katakana entstand wie Hiragana im 9. Jahrhundert und wurde vorwiegend von buddhistischen Mönchen verwendet. Hiragana und Katakana werden zusammen als Kana bezeichnet. Die Tabellen 1.2 und 1.3 zeigen die Grundzeichen und Aussprachevarianten der beiden Schriftsysteme.

Romaji: Diese Schrift gibt die Aussprache des Japanischen als lateinische Buchstaben wieder und ist als Hilfe für Ausländer, die keine japanischen Zeichen lesen können, entwickelt worden.

Die verschiedenen Arten von Zeichen[1] werden in japanischen Sätzen nebeneinander verwendet, wie dies die Abbildung 1.1 zeigt.

Kanji	**Watashi** 私				**i-** 行	
Hiragana		**wa** は		**ni** に		**kimasu** きます
Katakana			**Kanada** カナダ			

Abbildung 1.1: Aufbau eines japanischen Beispielsatzes (Übersetzung: Ich fahre nach Kanada)

1.3 Gliederung

Dieses Kapitel gab zunächst eine Einführung in den Bereich der Handschrifterkennung und die zu behandelnden Probleme. Außerdem wurde der Aufbau der japanischen Schriftzeichen betrachtet. In Kapitel 2 geht es um verschiedene Ansätze für die Erkennung und um grundlegende Techniken der Sprach- und Handschrifterkennung. Kapitel 3 stellt das hier verwendete Handschrifterkennungssystem und die erzielten Erkennungsraten vor, während Kapitel 4 andere Ansätze im Bereich der Online Erkennung japanischer Handschrift beschreibt. Im letzten Kapitel folgt eine Zusammenfassung der Studienarbeit und ein Ausblick auf weitergehende Entwicklungsmöglichkeiten.

[1] Für Zahlen werden im Japanischen häufig die arabischen Zahlen verwendet.

c \ v	a あ (ア)	i い (イ)	u う (ウ)	e え (エ)	o お (オ)
k	ka か (カ)	ki き (キ)	ku く (ク)	ke け (ケ)	ko こ (コ)
s	sa さ (サ)	shi し (シ)	su す (ス)	se せ (セ)	so そ (ソ)
t	ta た (タ)	chi ち (チ)	tsu つ (ツ)	te て (テ)	to と (ト)
n	na な (ナ)	ni に (ニ)	nu ぬ (ヌ)	ne ね (ネ)	no の (ノ)
h	ha は (ハ)	hi ひ (ヒ)	fu ふ (フ)	he へ (ヘ)	ho ほ (ホ)
m	ma ま (マ)	mi み (ミ)	mu む (ム)	me め (メ)	mo も (モ)
y	ya や (ヤ)	[i い (イ)]	yu ゆ (ユ)	[e え (エ)]	yo よ (ヨ)
r	ra ら (ラ)	ri り (リ)	ru る (ル)	re れ (レ)	ro ろ (ロ)
w	wa わ (ワ)	[i い (イ)]	[u う (ウ)]	[e え (エ)]	o を (ヲ)
n, m	— ん (ン)				

Abbildung 1.2: Grundsilben in Romaji, Hiragana und Katakana

g	ga が (ガ)	gi ぎ (ギ)	gu ぐ (グ)	ge げ (ゲ)	go ご (ゴ)
z	za ざ (ザ)	ji じ (ジ)	zu ず (ズ)	ze ぜ (ゼ)	zo ぞ (ゾ)
d	da だ (ダ)	ji ぢ (ヂ)	zu づ (ヅ)	de で (デ)	do ど (ド)
b	ba ば (バ)	bi び (ビ)	bu ぶ (ブ)	be べ (ベ)	bo ぼ (ボ)
p	pa ぱ (パ)	pi ぴ (ピ)	pu ぷ (プ)	pe ぺ (ペ)	po ぽ (ポ)

Abbildung 1.3: Variierte Silben in Romaji, Hiragana und Katakana

Kapitel 2

Grundlagen der Handschrifterkennung

2.1 Einleitung

In diesem Kapitel werden grundlegende Techniken der Handschrifterkennung vorgestellt. Sie bilden die Grundlage zum Verständnis für den Aufbau des im nächsten Kapitel vorgestellten Erkenners.

2.2 Erkennungsstrategien

Bei den Erkennungsstrategien wird zunächst einmal zwischen ganzheitlichen und analytischen Ansätzen unterschieden.

Bei ganzheitlichen Ansätzen wird in einem ersten Schritt eine Eingabefolge normalisiert und ein Merkmalsvektor berechnet. Im zweiten Schritt wird der Merkmalsvektor mit den Einträgen eines Lexikons verglichen und die wahrscheinlichste Folge ausgegeben. Daraus folgt, daß nur Wörter erkannt werden können, die sich im Lexikon befinden. Sollen neue Wörter hinzugefügt werden, ist ein Training auf Wortebene erforderlich, was aufwendig ist und damit den Ansatz unflexibel macht. Er wird deshalb nur selten angewendet, insbesondere dann, wenn sich das Vokabular nur selten ändert.

Im Gegensatz zu den ganzheitlichen Ansätzen wird bei den analytischen Ansätzen versucht, ein Wort in Untereinheiten z.B. in einzelne Zeichen zu zerlegen. Hierbei kann weiter unterschieden werden in Ansätze mit expliziter Segmentierung und solche mit impliziter Segmentierung.

Bei der expliziten Segmentierung (siehe Abbildung 2.1) wird zunächst versucht, ein Wort in seine Untereinheiten zu zerlegen, dann die einzelnen Untereinheiten isoliert voneinander zu erkennen und schließlich eventuell mit Hilfe eines Lexikons oder mit statistischen Informationen (N-Gramme) zu einem Wort zusammenzufügen.

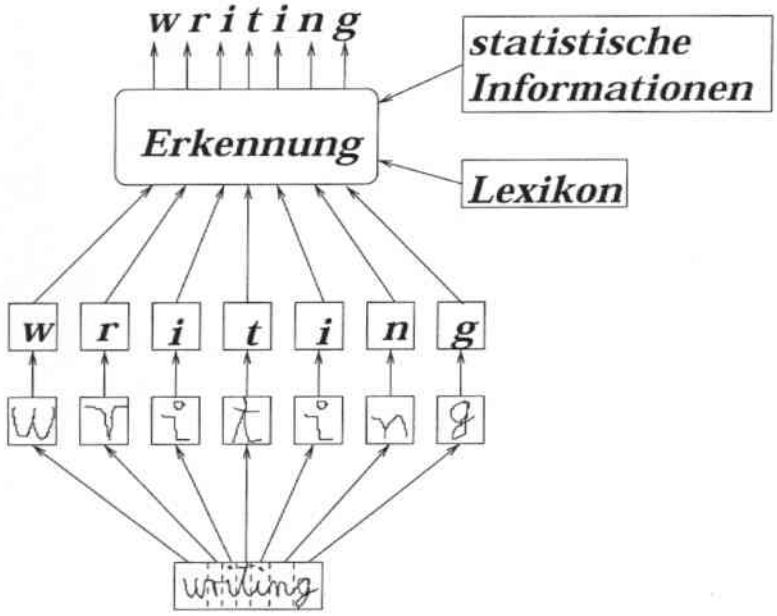

Abbildung 2.1: Explizite Segmentierung des Wortes „Writing“

Bei Ansätzen mit impliziter Segmentierung (siehe Abbildung 2.2) wird die Aufteilung des Wortes in Untereinheiten erst im Suchprozeß vorgenommen. Die Segmentierung und Erkennung finden gleichzeitig statt. Wissen über erlaubte Zeichenfolgen wird im Suchprozeß mit statistischen Informationen oder durch ein Wörterbuch berücksichtigt.

Bei dem hier vorgestellten Erkenner für Kanafolgen wird ein Ansatz mit impliziter Segmentierung verwendet.

2.3 Hidden Markov Modelle

Eine wichtige Modellierungstechnik der Handschrifterkennung bilden die *Hidden Markov Modelle* (für eine exakte Definition und die drei Hauptprobleme der Hidden Markov Modelle siehe [5]). Diese Modelle fanden in den achtziger Jahren weite Verbreitung unter anderem im Bereich der Spracherkennung, was durch ihre Fähigkeit, zeitliche Varianzen in der Eingabe zu behandeln, bedingt war.

In der Handschrifterkennung wird jedes zu erkennende Zeichen durch ein statistisches Modell mit einer Folge von Zuständen und Zustandsübergängen repräsentiert, wobei die Übergänge immer nur in einer Richtung erfolgen und nicht zu große Sprünge beinhalten dürfen. Verschiedene Modellierungen, wie in Abbildung 2.3 dargestellt, können dabei verwendet werden. Passende Gewichte für die Wahrscheinlichkeiten der einzelnen Übergänge werden durch das Training (Baum-Welch-Algorithmus) bestimmt. Bei der Erkennung ist

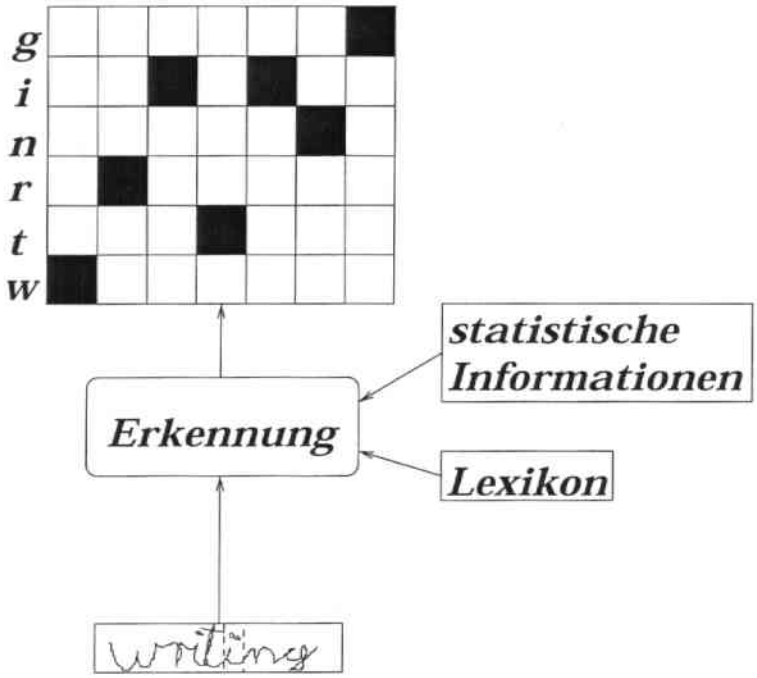

Abbildung 2.2: Implizite Segmentierung des Wortes „Writing"

es dann das Ziel, ein Zeichen z_i aus dem Zeichenmenge $Z = z_1, z_2, ..., z_n$ mit der größten a posteriori Wahrscheinlichkeit $P(z_i|X)$ unter der Eingabe X zu erhalten. Diese läßt sich bestimmen durch

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

mit
$P(X)$: Eingabewahrscheinlichkeit, die keinen Beitrag zur Entscheidung leistet
$P(X|z_i)$: Wahrscheinlichkeit für das Zeichen z_i die Eingabe X zu erhalten, was durch die Hidden Markov Modelle modelliert werden soll
$P(z_i)$: a priori Wahrscheinlichkeit des Zeichens z_i

Bei der Worterkennung könnten Modelle durch Konkatenation der Modelle der Einzelzeichen gewonnen werden.

2.4 Neuronale Netze

Die *neuronalen Netze*, die auf die Modellierung einer Nervenzelle durch McCulloch und Pitts im Jahr 1943 [6] zurückgehen, werden in der Sprach- und Handschrifterkennung eingesetzt. Ein neuronales Netz ist dabei aus verschiedenen *Neuronen* (siehe Abbildung 2.4) aufgebaut, die n Eingänge und einen

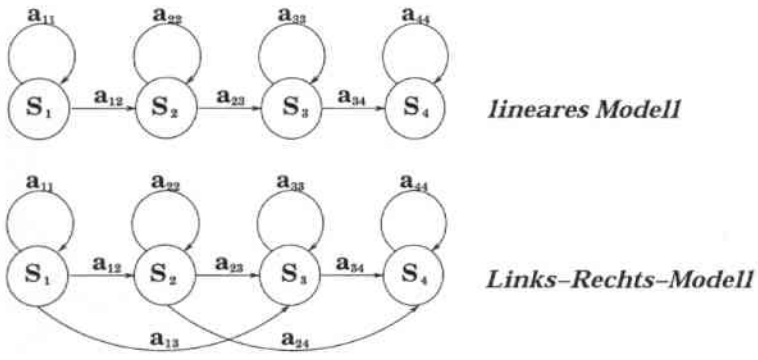

Abbildung 2.3: Verschiedene Hidden Markov Modelle

Ausgang haben. Die Eingänge können entweder aktiv oder passiv sein, was durch die Binärwerte 1 oder 0 ausgedrückt wird. Im Neuron selbst wird eine gewichtete Summe der Werte der Eingänge gebildet und mit einem Schwellwert verglichen. Ist der errechnete Wert größer als der Schwellwert, so erhält der Ausgang den Wert 1, anderenfalls 0.

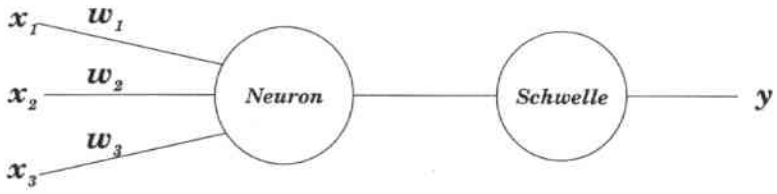

Abbildung 2.4: Aufbau eines Neurons mit drei Eingängen

Im Jahre 1962 wurde dann durch Rosenblatt [7] das sogenannte *einschichtige Perzeptron* eingeführt. Es besteht aus N Neuronen, die jeweils eine Klasse repräsentieren und die gleichen M Eingangsvariablen erhalten. An die Eingänge der Neuronen werden Muster angelegt, die aus den N verschiedenen Klassen stammen. Das Ziel ist es nun, passende Gewichte für die Eingänge der Neuronen zu finden, so daß jedes Neuron in der Lage ist zu unterscheiden, ob ein Muster zu der ihm zugeordneten Klasse gehört. Beim späteren Einsatz als Klassifikator wird dann in jedem Neuron die folgende Berechnung durchgeführt und man erhält entsprechende Aktivierungen der Ausgänge der Neuronen.

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

mit
y_n: Ausgabe des Neurons n
$f(x)$: Schwellwertfunktion mit Wert 1 für x >= 0 und 0 sonst
w_{ni}: Gewichtung des Eingangs i am Neuron n

x_i: Wert des Eingangs i

Ein großer Rückgang des Interesses an der Weiterentwicklung neuronaler Netze erfolgte 1969, als von Minsky und Papert [8] gezeigt wurde, daß ein solches einschichtiges Perzeptron nicht in der Lage ist, eine Exklusiv-oder-Funktion darzustellen.

Erst am Anfang der achtziger Jahre entstand neues Interesse an neuronalen Netzen durch die Einführung des Lernverfahrens *Backpropagation* [9] für mehrschichtige Perzeptrons (*multi-layer perceptrons*, siehe Abbildung 2.5), die in der Lage sind, auch komplexere Funktionen zu realisieren. Während die einschichtigen Perzeptrons nämlich nur solche Funktionen darstellen können, die linear separierbar sind, können durch zweischichtige konvexe Funktionen und durch dreischichtige beliebige stetige Funktionen approximiert werden, wodurch ein Ausweg aus den von Minsky und Papert aufgezeigten Einschränkungen gefunden war.

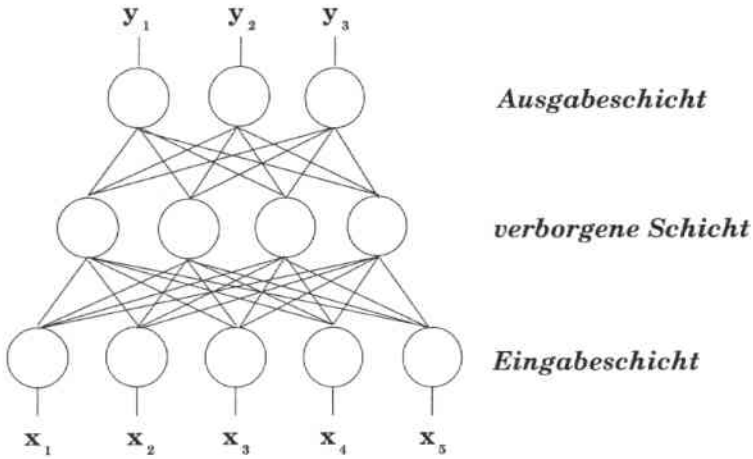

Abbildung 2.5: Aufbau eines dreischichtigen neuronalen Netzes

Die mehrschichtigen Perzeptrons, die heute sehr häufig verwendet werden, bestehen aus einer Eingabeschicht, einer oder mehrerer verborgener Schichten und einer Ausgabeschicht. Der Ausgang jedes Neurons einer Schicht wird mit allen Eingängen der Neuronen der folgenden Schicht verbunden. Ähnlich wie bei dem einschichtigen Perzeptron wird das Muster an die Neuronen der Eingabeschicht angelegt. An der Aktivität der Neuronen in der Ausgabeschicht, in der sich genausoviele Neuronen wie zu unterscheidende Klassen befinden, kann dann das Klassifikationsergebnis ermittelt werden. Als Schwellwertfunktion f(x) wird hier oftmals die Sigmoidfunktion

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

verwendet, um einen harten Übergang zwischen Aktivität und Nichtaktivität

des Neurons zu vermeiden.

Wie beim einschichtigen Perzeptron müssen passende Gewichte für die Eingänge der Neuronen ermittelt werden, um die gewünschte Klassifikation zu erreichen. Dieses kann mit Hilfe des Backpropagationverfahrens erreicht werden, welches das Ziel verfolgt, für eine Eingabe einer bestimmten Klasse eine möglichst hohe Aktivität des entsprechenden Neurons der Ausgabeschicht und eine möglichst geringe der anderen Neuronen zu erreichen. In der Trainingsphase werden dem neuronalen Netz verschiedene Muster bestimmter Klassen als Eingabe präsentiert und die tatsächliche Ausgabe mit der gewünschten Klassifizierung verglichen. Dazu wird eine Fehlerfunktion E eingeführt und ein Gesamtfehler ermittelt:

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

mit
y_n: tatsächliche Ausgabe des Neurons n
d_n: gewünschte Ausgabe des Neurons n, d.h. $d_n = 1$ für die richtige Klasse n, sonst 0

Die am häufigsten verwendete Fehlerfunktion ist der quadratische Fehler:

$$E_{MSE} = \frac{1}{2} \sum_{n=1}^{N} (y_n - d_n)^2$$

Um den an der Ausgabeschicht auftretenden Fehler zu minimieren, werden beim Backpropagationverfahren Gradienten des Fehlers nach den einzelnen Gewichten berechnet. Dann erfolgt mittels Gradientenabstieg eine Neuberechnung der Gewichte. Mit diesem Verfahren kann ein lokales Minimum der Fehlerfunktion gefunden werden, wobei dieses nicht das globale Minimum sein muß.

2.5 Time Delay Neural Network

Die *Time Delay Neural Networks (TDNN)* sind vorwärtsgerichtete Neuronale Netze, die erstmals im Jahr 1987 in der Spracherkennung eingesetzt wurden [10]. Das Ziel ihrer Entwicklung war es, in einer Sequenz von Eingabevektoren bestimmte Muster zu erkennen. Die Erkennung sollte nicht an vorgegebene Startpunkte der Muster gebunden sein und auch bei unterschiedlichen Längen der Muster funktionieren.

Das TDNN besteht aus vier verschiedenen Schichten (siehe Abbildung 2.6). In der Eingabeschicht werden die Merkmalsvektoren eingegeben. In der verborgenen Schicht werden die Aktivierungen der Neuronen zu einem gewissen Zeitpunkt nicht nur aus dem gewichteten Eingabevektor zu diesem

Zeitpunkt berechnet, sondern der zeitliche Kontext wird mit verwendet. Die Berechnung der Aktivierungen in der Zustandsschicht erfolgt ebenfalls mit Hilfe des Kontextes der Neuronen aus der verborgenen Schicht. Dieses dient dazu, Veränderungen der Eingabe im Zeitablauf besser erfassen zu können. Anschließend wird in der Ausgabeschicht eine Summierung der Aktivierungen für die einzelnen Zeichen vorgenommen und das wahrscheinlichste Zeichen ausgegeben.

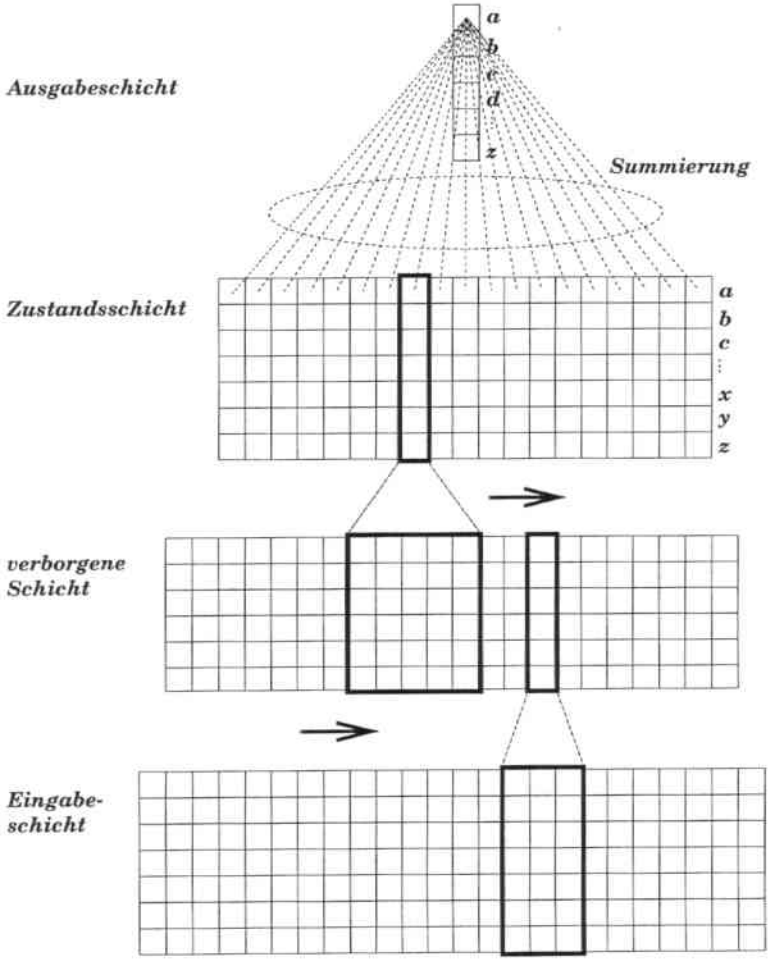

Abbildung 2.6: Architektur des Time Delay Neural Networks (TDNN)

2.6 Multi-State Time Delay Neural Network

In diesem Abschnitt geht es um das *Multi-State Time Delay Neural Network (MS-TDNN)*, das eine Erweiterung des TDNN darstellt. Dabei werden einzelne Zeichen in der Zustandsschicht nicht mehr durch einen Zustand

dargestellt, sondern durch eine Zustandsfolge. Diese Zerlegung diente in der Spracherkennung, aus der der Ansatz stammt [11], dazu, einzelne Buchstaben nicht als Ganzes, sondern durch ihre sprachlichen Untereinheiten (Phoneme) zu erkennen. Die Architektur des MS-TDNN ist in Abbildung 2.7 dargestellt und wird im folgenden beschrieben.

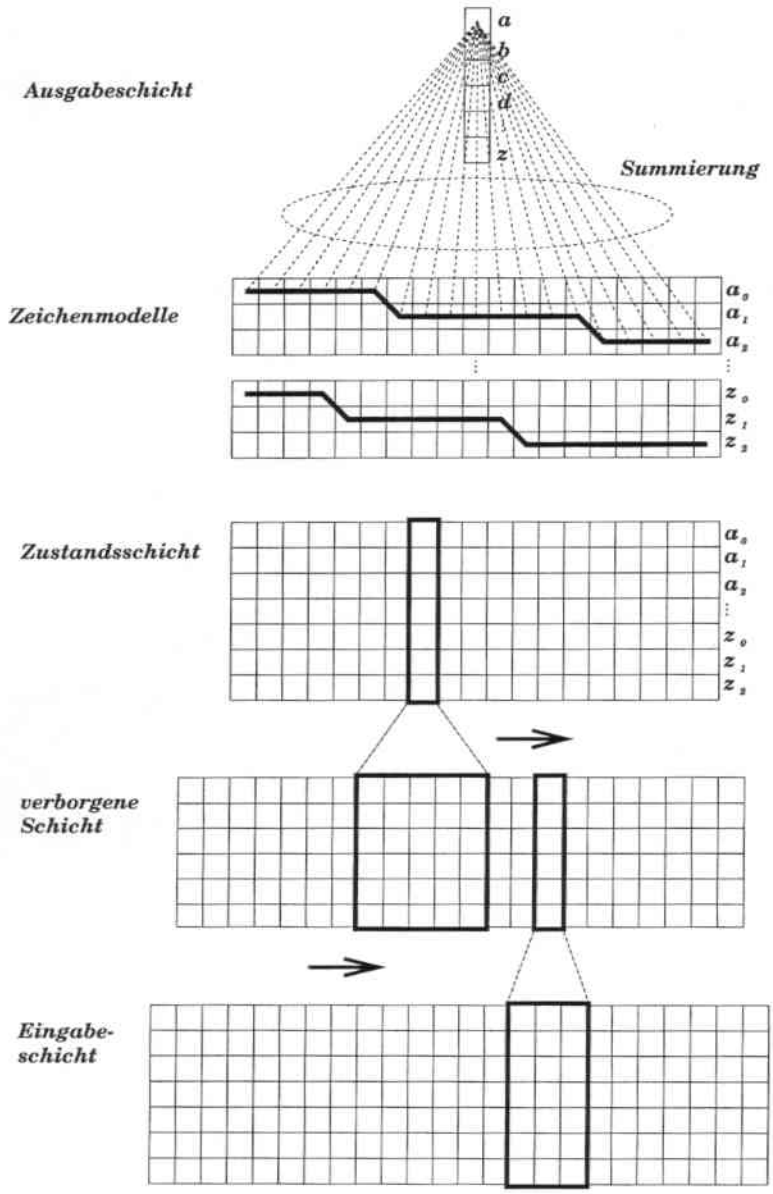

Abbildung 2.7: Architektur des Multi State Time Delay Neural Networks (MS-TDNN)

Das MS-TDNN ist aus fünf Schichten aufgebaut, wobei die ersten drei

Schichten mit denen aus dem TDNN vergleichbar sind. In der dritten Schicht erhält man jedoch Aktivierungen für die Zustände der Zeichen und nicht für die Zeichen selbst. Die Zeichenmodellschicht (Schicht 4) besteht aus linearen Modellen (siehe Abbildung 2.3), die die Zeichen repräsentieren. Diese werden verwendet, um in jedem Zeitpunkt und für jeden Zeichenzustand eine Wahrscheinlichkeit zu berechnen. Mit diesen Wahrscheinlichkeiten, die auch Aktivierungen genannt werden, wird nun für jedes Zeichen mit Hilfe des Viterbialgorithmus (siehe [12]) ein wahrscheinlichster Pfad von Start- zum Endzustand des Zeichens berechnet und eine Gesamtwahrscheinlichkeit für das Zeichen ausgegeben. Am Ende kann in der Ausgabeschicht das Zeichen mit der maximalen Gesamtwahrscheinlichkeit ausgegeben werden.

2.7 Worterkennung mit Vokabularen

In diesem Abschnitt geht es darum, wie bei einem Worterkenner ein Vokabular eingesetzt und in die Suche integriert werden kann.

Eine Möglichkeit der Verwendung eines Wörterbuches ist es, stets ein Wort aus dem Wörterbuch auszugeben. Diese Vorgehensweise führt zu einer erhöhten Erkennungsrate für die Wörter des Vokabulars, da einige der Verwechselungsmöglichkeiten ausgeschlossen werden, andererseits werden unbekannte Wörter auf jeden Fall falsch erkannt.

Wenn unbekannte Wörter jedoch häufig sind, kann man zusätzlich die Wahrscheinlichkeit für das Wort, was ausgegeben werden soll, betrachten. Sollte die Wahrscheinlichkeit gering sein, kann man vermuten, daß es sich um ein neues Wort handelt. Man kann dann versuchen, dieses ohne das Vokabular zu erkennen.

Eine einfache Art die Suche in einem Wörterbuch zu realisieren, ist die *flache Suche*. Dabei wird für jedes Wort aus dem Vokabular ein Wortmodell durch Konkatenation der Buchstabenmodelle für die einzelnen Zeichen gewonnen. Mit dem Viterbialgorithmus wird dann für jedes der Modelle ein optimaler Pfad berechnet und anschließend aus den Gesamtwahrscheinlichkeiten die beste Hypothese ermittelt. Diese Art der Suche nutzt zwar die vorhandenen Informationen optimal aus, doch wächst die Rechenzeit proportional mit der Vokabulargröße. Dieses macht eine solche Suche zu aufwendig, zumal bei gleichen Wortanfängen die gleichen Berechnungen wiederholt durchgeführt werden.

Mit der *Baumsuche* soll daher die Rechenzeit verkürzt werden, ohne die Erkennungsleistung zu sehr zu beeinträchigen. Hierbei werden identische Wortpräfixe nur einmal berechnet und das Vokabular in einer Baumstruktur dargestellt. Im Suchprozeß wird im Baum eine Breitensuche durchgeführt, die an den jeweils letzten Knoten eines Pfades endet. Um nicht Pfade, die sehr unwahrscheinlich sind, stets bis zum Ende weiterzuverfolgen, kann *Pruning* verwendet werden. Dabei wird zu bestimmten Zeitpunkten für jeden

Knoten des Baumes, der noch an der Suche beteiligt ist, untersucht, inwieweit sich die Wahrscheinlichkeit des Pfades zu diesem Knoten von der bisher besten Pfadwahrscheinlichkeit unterscheidet. Ist der Unterschied zu groß, wird der Pfad nicht mehr weiterverfolgt.

2.8 Zusammenfassung

In diesem Kapitel wurden grundlegende Vorgehensweisen und Modellierungstechniken der Handschrifterkennung vorgestellt. Besondere Bedeutung kommt den analytischen Ansätzen mit expliziter oder impliziter Segmentierung zu, die aufgrund ihrer höheren Flexibilität für die meisten Aufgaben geeigneter erscheinen als ganzheitliche Ansätze. Bei der expliziten Segmentierung wird gleich zu Beginn der Erkennung versucht, die Eingabe in unterschiedliche Bestandteile zu zerlegen, während bei Ansätzen mit impliziter Segmentierung die Erkennung gleichzeitig mit der Segmentierung erfolgt und so unter Umständen frühzeitige Fehlentscheidungen vermieden werden.

Als Modellierungstechniken wurden Hidden Markov Modelle und Neuronale Netze sowie TDNNs und MS-TDNNs vorgestellt. Eine Eigenschaft der Hidden Markov Modelle ist es, daß sie für die Berücksichtigung zeitlicher Varianz der Eingabe geeignet sind. Mit Hilfe von mehrschichtigen neuronalen Netzen können stetige Funktionen beliebig genau approximiert und die a posteriori Wahrscheinlichkeiten der Trainingsmuster geschätzt werden. In dem vorliegenden Erkenner wird ein MS-TDNN verwendet.

Das Wissen über die Sprache, aus der die zu erkennenden Worte stammen, kann mit Hilfe von Vokabularen in der Erkennung berücksichtigt werden.

Kapitel 3

Gesamtübersicht und Leistungsmessung

3.1 Einleitung

Dieses Kapitel dient dazu, einen Gesamtüberblick über das in dieser Arbeit verwendete und für japanische Zeichen weiterentwickelte System [1] zu geben. Hierbei werden zunächst das Ziel und der grobe Aufbau des Erkenners beschrieben, bevor genauer auf die Vorverarbeitung und das MS-TDNN eingegangen wird. Zum Schluß werden die verwendete Datenbasis und die Erkennungsraten vorgestellt sowie eine Fehleruntersuchung vorgenommen.

3.2 Ziel

Das Ziel dieser Arbeit bestand darin, einen Online Einzelzeichenerkenner für die japanischen Kanazeichen zu entwickeln. Dieser sollte dann zu einem Erkenner für Kanafolgen weiterentwickelt werden, um auf diese Weise auch die Eingabe von Kanjizeichen durch Nachsehen in einem Wörterbuch zu ermöglichen. Schließlich sollten noch zwei spezielle Einzelzeichenerkenner für die Hiragana- und Katakanazeichen entwickelt werden.

3.3 Aufbau des Erkenners

Bei diesem System wird eine Verarbeitungsreihenfolge verwendet, wie sie bei heutigen Erkennern im allgemeinen üblich ist. Zunächst wird die Eingabe, die aus Punkten mit x- und y-Koordinaten besteht, in den Erkenner eingegeben. Die Punkte werden in regelmäßigen Zeitabständen aufgenommen. Die Eingabe wird normalisiert und es werden verschiedene Merkmale berechnet, was zusammen als Vorverarbeitung bezeichnet wird. Anschließend wird die Sequenz der Merkmalsvektoren als Eingabe für den eigentlichen Erkenner

verwendet, wo durch ein MS-TDNN Zustandshypothesen erzeugt werden. Bei der Einzelzeichenerkennung wird dann die wahrscheinlichste Hypothese mit Hilfe des Viterbialgorithmus berechnet, während das Finden der Hypothese für die Folgenerkennung mit Hilfe einer Baumsuche im Wörterbuch durchgeführt wird.

Folgende Leitlinien wurden bei dem Entwurf des zugrundeliegenden Erkenners festgelegt:

- Es findet eine implizite Segmentierung statt.
- Das Wörterbuch wird direkt im Erkennungsprozeß verwendet.
- Dieselbe Architektur wird für alle Erkennungsaufgaben, d.h. für den Einzelzeichen- und den Folgenerkenner, benutzt.

3.4 Normalisierung

Bei der Eingabe der Zeichen treten verschiedene Störeinflüsse auf, die die Erkennung beeinträchtigen und deshalb so gut wie möglich unterdrückt werden müssen. Beispiele dafür sind:

Unterschiedliche Punktabstände: Diese können durch Unterschiede in der Schreibgeschwindigkeit, Überlastung der Hardware oder zu geringe Abtastraten der Hardware verursacht werden.

Rotation und Größenunterschiede: Ohne vorgegebene Schreiblinien kann es durch unterschiedliche Schreibgewohnheiten zu solchen Unterschieden kommen.

3.4.1 Größennormalisierung

Um Varianzen der Größe der Eingabe auszugleichen, wird eine Normalisierung vorgenommen, bei der die Eingabezeichen auf eine vorgegebene Höhe skaliert werden. Dabei wird die Breite der Zeichen variabel gelassen, um nicht das Wissen über das Verhältnis der Breite zur Höhe der Eingabe zu verlieren. Dabei tritt beim Japanischen das Problem auf, daß es von manchen Zeichen z.B. den Vokalen a,e,i,o,u große und kleine Varianten gibt, die sich nur in der Größe voneinander unterscheiden. Durch die Normalisierung wird deren Unterscheidung unmöglich, weshalb es bei der Einzelzeichenerkennung für jedes solche Zeichenpaar nur eine Klasse gibt.

3.4.2 Interpolation fehlender Datenpunkte mittels Bézierkurven

Falls zwischen zwei Punkten der Eingabe ein zu großer euklidischer Abstand besteht, soll dieses durch das Einfügen zusätzlicher Punkte ausgeglichen werden. Dabei werden zunächst Kontrollpunkte zwischen den beiden Punkten

berechnet, die auch die Eingangs- und Ausgangswinkel der beiden Punkte berücksichtigen. Anschließend wird eine Bézierkurve durch die Endpunkte und die Zwischenpunkte gelegt und entsprechend einem vorgegebenen Abstand neue Punkte eingefügt.

3.4.3 Glättung

Um kleinere Abweichungen, die beispielsweise durch Zittern entstehen, auszugleichen, wird eine Glättung durchgeführt. Dabei wird ein Punkt $(x(t), y(t))$ mit Hilfe von n vorangegangen und n nachfolgenden Punkten folgendermaßen verschoben:

$$x_{neu}(t) = \frac{x_{alt}(t-n) + ... + x_{alt}(t-1) + \alpha x_{alt} + x_{alt}(t+1) + ... + x_{alt}(t+n)}{2n + \alpha}$$

$$y_{neu}(t) = \frac{y_{alt}(t-n) + ... + y_{alt}(t-1) + \alpha y_{alt} + y_{alt}(t+1) + ... + y_{alt}(t+n)}{2n + \alpha}$$

Dabei hängt der Gewichtsfaktor α vom Winkel ab, den der Punkt mit den beiden benachbarten Punkten bildet, um ein unerwünschtes Abrunden bei signifikanten Richtungsänderungen zu verhindern.

3.4.4 Neuabtastung

Durch unterschiedliche Schreibgeschwindigkeiten oder unterschiedliche Abtastraten erhält man verschieden viele Datenpunkte. Um dieses auszugleichen, wird eine Neuabtastung vorgenommen, bei der die neuen Datenpunkte alle denselben räumlichen Abstand voneinander haben. Die Berechnung der neuen Punkte erfolgt mittels einer linearen Interpolation. Die Anzahl der Punkte muß sorgfältig ausgewählt werden, da bei zuvielen Punkten der Rechenaufwand unnötig erhöht wird, während bei zuwenigen Punkten Information verloren geht und sich damit die Erkennungsrate verschlechtert.

Bei der Einzelzeichenerkennung hat es sich als günstig erwiesen, jedes Zeichen auf eine feste Anzahl von Punkten zu normieren, anstatt einen festen Abstand der einzelnen Punkte vorzugeben und entsprechend viele Punkte zu erzeugen. Der beste Einzelzeichenerkenner für Hiragana mit einem festen Abstand der Punkte erreichte eine Erkennungsrate von 93,4%, während mit dem besten Einzelzeichenerkenner, der mit einer festen Anzahl von Punkten arbeitet, eine Erkennungsrate von 96,4% erreicht wurde.

Beim Kanafolgenerkenner ist es in der Vorverarbeitung nicht möglich, jede Folge auf eine feste Anzahl von Punkte zu normieren, da man schon a priori wissen müßte, aus wievielen Zeichen die Folge besteht. Sonst würden Folgen mit zwei Zeichen durch genausoviele Punkte dargestellt wie Folgen mit zehn Zeichen, was offensichtlich nicht sinnvoll ist. Deshalb wird die Eingabe des Folgenerkenners durch eine variable Anzahl von Punkten dargestellt.

3.4.5 Veränderungen der Normalisierung gegenüber dem ursprünglichen Erkenner

Einige Normalisierungsschritte, die Besonderheiten der Schreibweise englischer oder deutscher Wörter ausnutzen, wurden in diesem System nicht verwendet. Dazu gehören die Bestimmung verschiedener Schreiblinien, die Normalisierung von Neigungswinkeln und die besondere Behandlung sogenannter verzögerter Striche, die wie beispielsweise beim „t“ erst am Ende des Wortes geschrieben werden.

3.5 Merkmalsextraktion

Ausgehend von der Normalisierung wird bei der Merkmalsextraktion für jeden Datenpunkt ein Merkmalsvektor berechnet, der als Eingabe für den eigentlichen Erkenner dient. Die verwendeten Merkmale haben das Ziel, die charakteristischen Eigenschaften aus der Trajektorie zu extrahieren. Dabei werden die im Merkmalsvektor enthaltenen Merkmale in lokale und globale Merkmale eingeteilt.

3.5.1 Lokale Merkmale

Diese Merkmale werden für einen Datenpunkt aus der lokalen zeitlichen Umgebung, also aus vorangehenden und nachfolgenden Punkten, erzeugt. Folgende Merkmale werden dabei berechnet:

Vertikale Position: Die y-Position relativ zur Gesamthöhe des Zeichens wird bestimmt.

Schreibrichtung: Als Merkmale für die Schreibrichtung α (siehe Abbildung 3.1) werden drei Merkmale berechnet:

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

Krümmung: Für die Krümmung (siehe Abbildung 3.1) am aktuellen Punkt $(x(t), y(t))$ werden der Sinus und der Cosinus des Winkels β zwischen den Punkten $(x(t-2), y(t-2))$, $(x(t), y(t))$ und $(x(t+2), y(t+2))$ angegeben.

Erscheinungsbild: Dieses und die drei folgenden Merkmale [13] beziehen sich auf eine Umgebung des aktuellen Punktes mit zwei vorangehenden und zwei nachfolgenden Punkten, wie sie in der Abbildung 3.2 dargestellt ist. Das Erscheinungsbild gibt das Verhältnis der vertikalen zur

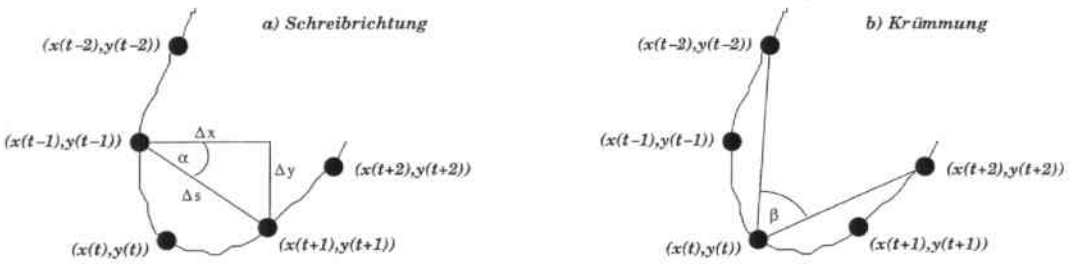

Abbildung 3.1: Berechnung von Schreibrichtung (a) und Krümmung (b)

horizontalen Ausdehnung an:

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

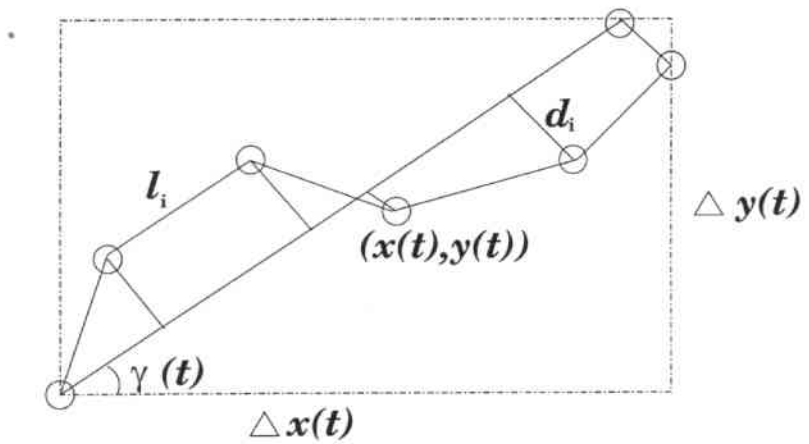

Abbildung 3.2: Definitionen für die Berechnung weiterer Merkmale

Gewelltheit: Dieses Merkmal ist ein Maß für den Unterschied zwischen der Länge des Liniensegmentes und einer Geraden. Es wird berechnet durch:

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

Linearität: Dieses Merkmal, das ebenfalls die Linearität des Kurvenverlaufs beschreibt, wird folgendermaßen bestimmt:

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

Neigung: Das Neigungsmerkmal enthält den Cosinus des Winkels γ zwischen der Horizontalen und der Verbindung des ersten mit dem letzten Punkt.

Bei den lokalen Merkmalen wurde das „Hut"-Merkmal, das im Erkenner für englische Wörter zur Detektion verzögerter t-Striche und i-Punkte eingesetzt wurde, weggelassen. Bei den japanischen Zeichenfolgen kommt ein Zurückspringen zu vorherigen Zeichen im allgemeinen nicht vor, was das Merkmal überflüssig macht.

3.5.2 Globale Merkmale

Die globalen Merkmale versuchen, auch Zusammenhänge zwischen zeitlich auseinanderliegenden Punkten aufzuzeigen. In diesem Fall geht es besonders darum, Schnittpunkte zweier zeitlich auseinanderliegender Segmente zu erkennen, wozu die sogenannten *Kontext-Bitmaps* verwendet werden. Dabei wird für jeden Punkt ein Grauwertbild, das aus einer (3,3)-Matrix besteht, berechnet, um das räumliche Umfeld des Punktes abzubilden. Die Berechnung eines Unter- und Oberlängenmerkmals wurde hier weggelassen, da dieses Merkmal für die japanischen Zeichen nicht geeignet erscheint und durch die Auslassung der Basisliniennormalisierung auch nicht sinnvoll berechnet werden kann.

3.6 Aufbau des MS-TDNN

Für die Erkennung der Einzelzeichen und Kanafolgen wurde ein MS-TDNN (siehe [1]) eingesetzt, dessen grundlegende Architektur in Abschnitt 2.6 beschrieben wurde.

3.6.1 Zeichenmodellierung

Im Gegensatz zum TDNN werden beim MS-TDNN die Zeichen durch mehrere Zustände modelliert. Der ursprüngliche Erkenner verwendete für jedes Zeichen drei Zustände für die Modellierung, was sich für die englischen Wörter als optimal erwiesen hatte und bei diesem Erkenner zunächst beibehalten wurde. Durch die Erhöhung der Zustandsanzahl auf fünf für jedes Zeichen konnte dann aber für die Kanazeichen eine Verbesserung von 0,5% (absolut) erreicht werden. Um zusätzlich das Wissen über die im Japanischen vorgegebene Strichanzahl zu nutzen, wurden in einem weiteren Versuch die Zeichen durch genausoviele Zustände wie das Doppelte der Strichanzahl modelliert. Dieses führte zu einer weiteren Verbesserung der Erkennungsleistung um 0,9%.

3.6.2 Details zum Erkenneraufbau

Beim Übergang von der Eingabe- zur verborgenen Schicht sowie von der verborgenen Schicht zur Zustandsschicht wurde jeweils eine Kontextbreite

von 7 verwendet. Als Transferfunktion wurde die Sigmoidfunktion und als Fehlermaß die Cross Entropy benutzt.

3.6.3 Training des MS-TDNN

Beim Training werden die Gewichte des neuronalen Netzes mit Hilfe des Backpropagationverfahrens durch Gradientenabstieg eingestellt. Beim Lernen werden nach jeder Präsentation eines Lernmusters und Rückpropagierung des Fehlers alle Gewichte im neuronalen Netz aktualisiert. In jeder Iteration werden alle Muster dem Netz in zufälliger Reihenfolge präsentiert. Der Zeitpunkt des Wechsels von einer Trainingsstufe zur nächsten wird heuristisch festgelegt.

Das Training für die Einzelzeichenerkennung erfolgt in zwei Schritten.

Training auf der Zustandsebene: Hierbei sollen zunächst die Gewichte von der Eingangs- zur verborgenen Schicht und von der verborgenen Schicht zur Zustandsschicht trainiert werden. Dabei wird als Zielvorgabe für die Zustandsschicht angenommen, die Zustände seien gleichmäßig über die Zeit verteilt (siehe Abbildung 3.3). Nach zwei Iterationen wird dann zum Training auf der Zeichenebene gewechselt.

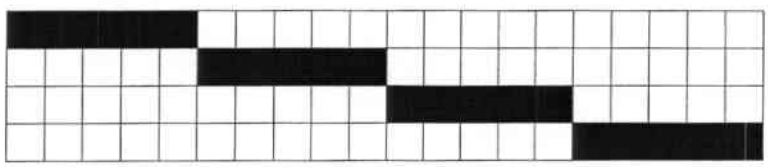

Abbildung 3.3: Sollwerte beim Training auf der Zustandsebene

Training auf der Zeichenebene: In diesem Trainingsschritt soll eine optimale Aufteilung der Zustände über die Zeit erreicht werden. Die optimalen Zeitpunkte des Übergangs von einem Zustand zum nächsten wird nun mit Hilfe des Viterbialgorithmus vom MS-TDNN selbst bestimmt (siehe Abbildung 3.4). Das vorherige Training auf der Zustandsebene lenkt das Finden des optimalen Pfades so, daß es nicht zu großen Ungleichgewichten zwischen den Zuständen kommt.

Das Training für die Kanafolgenerkennung umfasst auch die Trainingsschritte auf der Zustands- und Zeichenebene, doch wird es durch ein Training auf der Wortebene ergänzt.

Training auf Wortebene: Beim Training mit unsegmentierten Kanafolgen werden mit dem Viterbialgorithmus Zielpfade auch über Zeichengrenzen hinweg berechnet. Dieses wird durch das vorausgegangene

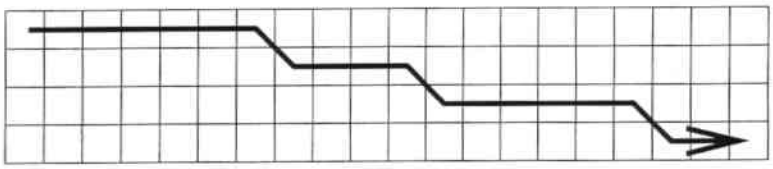

Abbildung 3.4: Bestimmung des Viterbipfades beim Training auf der Zeichenebene

Training auf Zustands- und Zeichenebene unterstützt, da sich dort bereits Pfade für die einzelnen Zeichen herausgebildet haben. Das Training auf Wortebene erfolgt abwechselnd mit dem Training auf Zeichenebene.

3.7 Verwendete Daten und Erkennungsergebnisse

Für diesen Erkenner wurde eine Datenbasis mit 28 Schreibern verwendet, bei der jeder der Schreiber alle Kanazeichen (128 einmal und 18 in der großen und kleinen Variante) geschrieben hat. Zum Training des Erkenners wurden dann die Daten von 22 Schreibern ausgewählt, während die Daten der anderen 6 ausschließlich zum Testen verwendet wurden. Die Tabelle 3.1 zeigt eine Übersicht der verwendeten Zeichenmengen.

Erkenner	Trainingsmenge	Testmenge	Summe
Kana	3608 Zeichen	984 Zeichen	4592 Zeichen
Hiragana	1804 Zeichen	492 Zeichen	2296 Zeichen
Katakana	1804 Zeichen	492 Zeichen	2296 Zeichen

Tabelle 3.1: Übersicht über die verwendeten Daten

Die Tabelle 3.2 zeigt die erreichten Erkennungsraten. Bei der Folgenerkennung wurde das Edict Wörterbuch von Jim Breen [14] verwendet und Kanjizeichen, deren Aussprachen nur mit Hiraganazeichen geschrieben wurden, herausextrahiert. Mit den Aussprachefolgen wurden dann durch Kontatenation der Einzelzeichen aus der Datenbasis Eingaben für die Folgenerkennung erzeugt. Hierbei wird unterschieden zwischen einem auf Kanazeichen basierenden und einem auf Hiraganazeichen basierenden Folgenerkenner. Da nur sehr wenige Folgen mit Katakanazeichen im Wörterbuch enthalten sind, wurden diese beim Folgenerkenner nicht berücksichtigt und deshalb auch kein auf Katakanazeichen basierender Erkenner entwickelt.

Es wurde auch noch untersucht, wie schwierig die Klassifikation eines Zeichens als Hiragana- oder Katakanazeichen ist. Hierbei sollte der Kanaeinzelzeichenerkenner anzeigen, ob es sich bei einem eingegebenen Zeichen um

Zeichenmenge	Einzelzeichenerkennung	Folgenerkennung
Kana (146 Zeichen)	92,9%	89,3%
Hiragana (73 Zeichen)	96,4%	88,3%
Katakana (73 Zeichen)	97,0%	-

Tabelle 3.2: Erkennungsraten der verschiedenen Erkenner

ein Hiragana- oder ein Katakanazeichen handelte. Dieses gelang in 96,59% der Fälle.

3.8 Fehleranalyse bei der Einzelzeichenerkennung

Um die Fehler zu untersuchen, die der Erkenner bei der Einzelzeichenerkennung macht, wurden verschiedene Untersuchungen vorgenommen.

Zunächst wurden bei den drei Einzelzeichenerkennern die Hypothesen und Referenzen bei Fehlerkennungen untersucht. Hier stellte sich heraus, daß es vor allem zu Verwechselungen zwischen der b- und der p-Reihe (siehe Abbildung 1.3) kommt. Dabei wird dasselbe Grundzeichen aus der h-Reihe einmal mit zwei Strichen (b-Reihe) und einmal mit einem kleinen Kreis (p-Reihe) geschrieben, d.h. die Unterscheidung ist nur über einen sehr kleinen Teil des Zeichens möglich. Bei den übrigen Fehlern konnte keine weitere Systematik festgestellt werden.

Nun wurde weiter untersucht, ob sich die Referenz unter den Zeichen befindet, die der Erkenner neben der Hypothese für am wahrscheinlichsten hält (n-best). Die Tabelle 3.3 zeigt die Ergebnisse für die drei Erkenner, wobei beim Hiraganaerkenner zusätzlich die Daten des Erkenners mit festem Punktabstand und damit variabeler Punktanzahl (siehe Abschnitt 3.4.4) angegeben wurden. Hierbei zeigt sich, daß die zweitbeste Hypothese bei den Fehlererkennungen sehr häufig das richtige Zeichen enthält, was durch die Verwechselungen zwischen der b- und der p-Reihe zu erklären ist.

Zeichenmenge	1-best	2-best	3-best	4-best	5-best
Kana	92,87%	98,39%	99,00%	99,10%	99,30%
Hiragana (feste Punktanzahl)	96,38%	98,12%	98,79%	99,40%	99,80%
Hiragana (var. Punktanzahl)	93,36%	97,98%	98,19%	98,59%	99,40%
Katakana	96,99%	99,60%	99,60%	99,60%	99,60%

Tabelle 3.3: Erkennungsraten für n-best Hypothesen

3.9 Fehleranalyse bei der Folgenerkennung

Im Bereich der Folgenerkennung wurde untersucht, ob sich die gesuchten Kanafolgen innerhalb der n-best Liste (Tabelle 3.4) befinden.

Zeichenmenge	1-best	2-best	3-best	4-best	5-best
Kana	89,33%	91,67%	93,17%	94,33%	95,00%
Hiragana	88,33%	91,83%	93,50%	93,83%	94,50%

Tabelle 3.4: Erkennungsraten für n-best Hypothesen

Desweiteren wurden die Längen der Hypothesen und Referenzen miteinander verglichen und die Anzahl der aufgetretenen Fehler aufsummiert (siehe Tabelle 3.5). Dieses erfolgte mit dem Ziel festzustellen, ob die Fehler durch Einfügungen, Auslassungen oder Ersetzungen entstanden sind.

Zeichenmenge	-4	-3	-2	-1	0	1	2
Kana	0	2	15	16	20	10	1
Hiragana	2	4	18	17	20	9	0

Tabelle 3.5: Differenz zwischen der Länge der Hypothese und der Länge der Referenz

Zuletzt wurde noch untersucht, welchen Einfluß die Vokabulargröße (siehe Tabelle 3.6) auf die Erkennungsrate hat.

Zeichenmenge	5000	10000	20000	39664
Kana	92,50%	90,83%	90,50%	89,33%
Hiragana	92,33%	90,83%	90,67%	88,33%

Tabelle 3.6: Erkennungsraten in Abhängigkeit von der Vokabulargröße

Kapitel 4

Vergleich mit anderen Erkennungssystemen

4.1 Einleitung

In diesem Kapitel werden zwei andere japanische Handschrifterkenner vorgestellt. Die Systeme beschränken sich im Gegensatz zu dem in dieser Studienarbeit vorgestellten Erkenner nicht auf die Kanazeichen, sondern erlauben auch die Erkennung von Kanji und arabischen Zahlen.

4.2 Einzelzeichenerkenner für verschiedene Eingabequalitäten

Dieser Einzelzeichenerkenner [15], der von M. Hamanaka und K. Yamada auf der IWFHR 2000 vorgestellt wurde, hatte das Ziel, nicht nur sorgfältig geschriebene Zeichen, sondern auch schneller geschriebene und nicht mit der vorgegebenen Strichanzahl geschriebene Zeichen zu erkennen.

Hierbei wurde zunächst eine Datenbasis erstellt, bei der Schreiber die Zeichen in einer sorgfältigen, einer normalen und einer undeutlichen Art schreiben sollten. Während es bei den sorgfältigen Schreibweisen nur selten zu Abweichungen von der vorgegebenen Strichanzahl kommt, sind bei der undeutlichen Art häufig Striche miteinander verbunden. Aus diesen Daten wurden Verteilungen für die Anzahl der tatsächlich geschriebenen Striche im Verhältnis zur Anzahl der vorgegebenen Striche berechnet.

Bei der Bedienung des Erkenners hat der Schreiber die Möglichkeit, manuell zwischen den Modellen für die drei Qualitäten zu wählen oder die Qualität automatisch erkennen zu lassen. Bei der automatischen Variante wird die Schreibgeschwindigkeit aus der Anzahl der aufgezeichneten Punkte bestimmt. Hierbei wird angenommen, daß die Eingabe umso ungenauer ist, je schneller geschrieben wird.

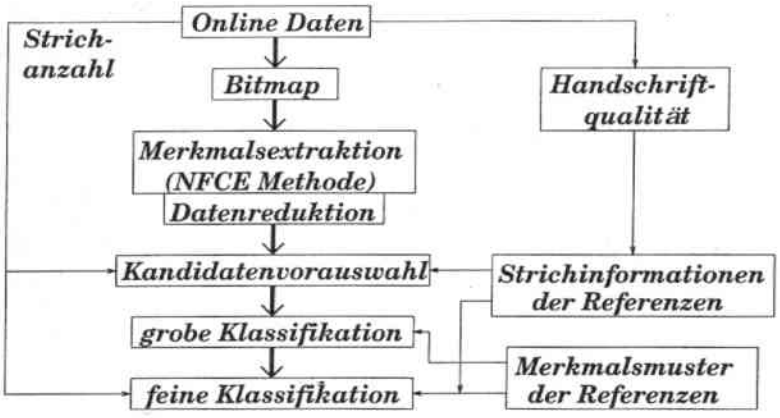

Abbildung 4.1: Handschrifterkennungssystem mit der Flexible-Pattern-Matching Technik

Die eigentliche Erkennung erfolgt mit Hilfe der *Flexible-Pattern-Matching* Technik [16], die in Abbildung 4.1 dargestellt ist. Am Anfang werden die Online Daten auf eine vorgegebene Größe skaliert, neu abgetastet und geglättet. Dann wird aus den Daten ein Bitmap erzeugt, aus dem durch Anwendung der *Normalization-Cooperative Feature Extraction* [17] Merkmale für verschiedene Richtungen berechnet werden. Schließlich erhält man ein Muster mit 256 Elementen. Durch Mittelwertbildung von vier benachbarten Elementen wird dann noch zusätzlich ein Muster mit 64 Elementen erzeugt.

Die Klassifikation selbst besteht aus drei Stufen: Kandidatenvorauswahl, Grob- und Feinklassifikation. Bei der Kandidatenvorauswahl und der Feinklassifikation werden die Informationen über die tatsächliche Anzahl der Striche und die geschätzte Qualität der Handschrift verwendet. Die Qualität bestimmt dabei, welche der früher berechneten Verteilungsfunktionen (sorgfältig, normal oder undeutlich) verwendet werden soll. Bei der groben Klassifikation wird die aktuelle Eingabe mit den 64-elementigen Referenzmustern verglichen, während bei der Feinklassifikation die 264-elementigen Muster verwendet werden.

Als Datenbasis wurden hier Daten von 51 Schreibern benutzt, die jeweils die 2965 Kanji aus dem japanischen Industriestandard JIS und 232 weitere Zeichen (71 Hiragana, 71 Katakana, 62 alphanumerische Zeichen und 28 weitere Symbole) geschrieben haben. Dabei wurde für die sorgfältig geschriebenen Daten eine Erkennungsrate 94,3%, für die normalen 90,53% und für die undeutlichen 76,61% erreicht.

Dieser Ansatz unterscheidet sich von dem in dieser Studienarbeit vorgestellten Erkenner darin, daß hier am Anfang zwar einige Informationen aus den Online Daten entnommmen werden (Strichanzahl und Schreibgeschwindigkeit), dann jedoch die Daten in Offline Daten (Bitmap) gewandelt und die extrahierten Informationen als Parameter verwendet werden. Dage-

gen gehen beim MS-TDNN die Informationen über die zeitliche Abfolge im Laufe der Erkennung nicht verloren.

4.3 Folgenerkenner für Kana, Kanji und Zahlen

Der Folgenerkenner von T. Fukushima und M. Nakagawa [18], der auf der ICPR 2000 vorgestellt wurde, hatte das Ziel, Folgen, die aus Kana, Kanji und Zahlen bestehen, zu erkennen.

Es wird in zwei Schritten versucht, für eine Eingabe, die nicht in vorgegebene Kästchen erfolgt, mögliche Zeichengrenzen zu finden.

Hierbei werden im ersten Schritt mit Hilfe der Distanz vom Ende des Schreibens des einen Striches zum Anfang des nächsten Striches Hypothesen für die Zeichengrenzen erzeugt. Anschließend wird mit Hilfe von Histogrammen die Höhe und Breite der vermuteten Zeichen geschätzt. Durch Mittelwertbildung erhält man dann eine Schätzung für die durchschnittliche Größe der Zeichen.

Diese durchschnittliche Größe dient nun dazu, im zweiten Schritt neue Hypothesen für die Zeichengrenzen zu erzeugen. Dabei berechnet man den Schwerpunkt aller Striche und definiert die neue Distanz Dx von Strichen durch die Differenz der x-Koordinaten der Schwerpunkte geteilt durch die durchschnittliche Zeichengröße. Ist Dx größer als ein Schwellwert, wird segmentiert, ist Dx unter einem anderen Schwellwert, wird niemals segmentiert, bei den anderen Fällen wird die Stelle als Kandidat vermerkt.

Die eigentliche Erkennung erfolgt dann mit Hilfe eines Maximum Likelihood Klassifikators $L(C|X)$ mit der Zeichenfolge $C = C_1C_2\cdots C_N$ und der Eingabefolge $X = X_1X_2\cdots X_N$.

$$\begin{aligned} L(C|X) &= \sum_{i=0}^{N} logP(C_{i+1}|C_i) \\ &\quad + \sum_{i=1}^{N}\Big(logP(Size_i|C_i) + \sum_j logP(d_j|C_i) + logP(X_i|C_i)\Big) \\ &\quad + \sum_{i=1}^{N-1} logP(D_i|Segmentierung) \end{aligned}$$

mit
$P(C_{i+1}|C_i)$: Wahrscheinlichkeit des Übergangs von Zeichen C_i zu Zeichen C_{i+1}
$P(Size_i|C_i)$: Wahrscheinlichkeit der Größe der Eingabe an der Stelle i bei geschätzter Größe des Zeichens C_i
$P(d_j|C_i)$: Wahrscheinlichkeit des innerhalb des Zeichens aufgetretenen Strichabstandes d_j
$P(X_i|C_i)$: Maß für die Ähnlichkeit von X_i und C_i [19]

$P(D_i|Segmentierung)$: Wahrscheinlichkeit des Abstandes D_i zwischen den einzelnen Zeichen

Mit Hilfe der dynamischen Programmierung wird dann eine Zeichenfolge C_{max} gefunden, die die Gleichung maximiert.

Als Datenbasis wurden hier Daten von acht Schreibern verwendet, die jeweils neun Sätze mit 205 Zeichen geschrieben haben. Dabei wurde eine Erkennungsrate von 78,8% erreicht.

Bei diesem Erkennungsansatz wird versucht, aus der Eingabe Hypothesen für mögliche Segmentierungen abzuleiten und dann Zeichenketten mit Zeichen, die zu den Segmentierungen passen, zu finden. Bei dem Erkenner aus dieser Studienarbeit wird eine implizite Segmentierung durchgeführt, bei der die Erkennung und Segmentierung gleichzeitig stattfinden.

Kapitel 5

Zusammenfassung und Ausblick

In der vorliegenden Studienarbeit wurde ein japanischer Handschrifterkenner entwickelt, der für die Erkennung von einzelnen Kanazeichen sowie für die Erkennung von Kanafolgen geeignet ist. Durch die Erkennung der Kanafolgen (39664 verschiedene) können auch Kanjizeichen eingegeben werden.

Die Erkennung erfolgt mit Hilfe eines MS-TDNN (siehe Kapitel 2.6), das aus einem neuronalen Netz und statistischen Modellen besteht. Dabei kann aufgrund des hierarchischen Aufbaus des Erkenners dieselbe Architektur sowohl für die Einzelzeichenerkennung als auch für die Folgenerkennung verwendet werden. Bei der Folgenerkennung wurde ein Ansatz mit impliziter Segmentierung gewählt und die Daten eines Wörterbuches schon während des Erkennungsprozesses verwendet.

Für die Zukunft sind verschiedene Möglichkeiten für die Verbessung des Systems denkbar:

- Der für die Erkennung verwendete Merkmalsvektor umfasst allgemeine Merkmale von Handschrift. Hier könnte durch Definition spezieller Merkmale für die japanische Handschrift eventuell eine Verbesserung erzielt werden.

- Das Wissen über die für jedes Zeichen vorgeschriebene Anzahl und Reihenfolge der Striche wurde in diesem Erkenner bei der Wahl der Anzahl der Zustände für die Zeichenmodellierung genutzt. Beim Einzelzeichenerkenner von M. Hamanaka und K. Yamada wurde es für die Wahl der geeigneten Verteilungsfunktion verwendet. Weitere Möglichkeiten, dieses Wissen im Erkenner zu verwenden, könnten untersucht werden.

- Am besten für den praktischen Einsatz wäre es, wenn man die Kanjizeichen auch direkt eingeben könnte anstatt den Umweg über die Kanafolgen machen zu müssen. Eine Erweiterung des Erkenners für

diese direkte Eingabe muß jedoch so erfolgen, daß Trainings- und Erkennungszeiten akzeptabel blieben, wozu umfangreiche Veränderungen der Erkennerarchitektur notwendig wären.

Literaturverzeichnis

[1] S. Manke. *On-line Erkennung kursiver Handschrift bei großen Vokabularen.* Dissertation, Universität Karlsruhe, 1998.

[2] 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, Colchester, England, September 1996.

[3] 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, München, 1997.

[4] Association for Japanese-Language Teaching. *Japanisch im Sauseschritt 1.* Doitsu Center Ltd., Seiten 9-20, 2000.

[5] L. R. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, Feb. 1989.

[6] W. S. McCulloch and W. Pitts. *A Logical Calculus of Ideas Immanent in Nervous Activity.* Bulletin of Mathematical Biophysics, 5:115-133, 1943.

[7] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, Washington, DC, 1962.

[8] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[9] D. E. Rummelhart, G. E. Hinton and R. J. Williams. *Learning Representations by Back-Propagating Errors.* Nature, 323(9):533-536, October 1986.

[10] 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.

[11] 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.

[12] T. Cormen, C. Leiserson and R. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[13] M. E. Schenkel. *Handwriting Recognition Using Neural Networks and Hidden Markov Models*, volume 45 of Series in Microelectronics. Hartung-Gorre, Konstanz, 1995.

[14] Website: www.csse.monash.edu.au/~jwb

[15] M. Hamanaka and K. Yamada. *On-Line Character Recognition Adaptively Controlled by Handwriting Quality.* In Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, pp. 23-32, September 2000.

[16] M. Hamanaka, K. Yamada and J. Tsukumo. *On-Line Japanese Character Recognition Experiments By an Off-Line Method Based on Normalization-Cooperative Feature Extraction.* Proceedings of the 2nd ICDAR, pp. 204-207, 1993.

[17] M. Hamanaka, K. Yamada and J. Tsukumo. *Handprinted Kanji Character Recognition Using Normalization-Cooperated Feature Extraction.* Proceedings of the 3rd IWFHR, pp. 343-348, 1993.

[18] T. Furushima and M. Nakagawa. *On-Line Writing-box-free Recognition of Handwritten Japanese Text Considering Character Size Variations.* Proceedings of the 15th ICPR, Vol. 2, pp. 359-363, September 2000.

[19] M. Nakagawa and K. Akiyama. *A Linear-time Elastic Matching for Stroke Number Free Recognition of On-line Handwritten Characters.* Proceedings of the 4th IWFHR, pp. 423-430, 1994.