Kapitel 6

Textmodellierung

Bisher realisierte Ansätze für die Online-Handschrifterkennung ermöglichen entweder

1. die Erkennung ohne vorgegebenen Wortschatz, was dafür aber die Eingabe *einzelner* Druckschrift-ähnlicher Buchstaben erfordert. Hier werden die Buchstaben einzeln erkannt und später aneinandergereiht. Oder sie ermöglichen

2. die verbundene Eingabe von Buchstabenfolgen, mit der Beschränkung auf bestimmte Worte (festes Vokabular).

Bei der Handschrifterkennung müßte demnach grundsätzlich zwischen einem Lexikon-freien Modus bei Eingabe segmentierbarer Einzelzeichen, und einem Lexikon-basierten Modus mit der Option zur Eingabe verbundener Zeichen ausgewählt werden.
Die zweite Variante wurde in den bisherigen Kapiteln verfolgt. Um diese Beschränkung soweit wie möglich abzuschwächen, wurde dabei versucht die Lexikongröße zu maximieren. Gerade im Deutschen mit der intensiven Nutzung von Komposita wird es hingegen immer wieder auffallen, dass ein Lexikon nie vollständig ist.
Der entscheidende Vorteil des hier vorgestellten Verfahrens ist nun die Kombination der Vorteile beider Modi [Kos99c]. Es kann Text unter Verwendung verbundener Buchstaben eingegeben werden. Der Benutzer ist dabei jedoch nicht auf ein festes Lexikon, bzw. auf einen definierten Wortschatz beschränkt. D. h., dass die Kombination von HMM und Textmodellen hier erstmals eine automatische schreiberunabhängige Handschrifterkennung ermöglicht ohne besondere Voraussetzungen bezüglich des verwendeten Vokabulars oder der Schriftart (Font). Als Textmodelle werden hier statistische Modelle (N-Gramme) bezeichnet, die nicht wie üblich auf Wortbasis, sondern auf Einzelzeichenbasis trainiert werden.

6.1 Verknüpfung von Graphem- und Textmodellen

Für die Erkennung von Wörtern oder Sätzen mit großem Vokabular (einige tausend Wörter) gilt folgender klassischer Ansatz als Stand der Technik:
Die einzelnen Buchstaben werden mit HMM anhand von Trainingsbeispielen modelliert. In einem Lexikon, welches den erlaubten Wortschatz definiert, ist ein gültiges Wort durch die Folge seiner Buchstaben bzw. HMM beschrieben. Mit einer Viterbi-Dekodierung kann bei der Erkennung eines unbekannten Wortes, welches durch die Abtastsequenz X repräsentiert ist, das wahrscheinlichste Wort W^* aus allen Worten W gefunden werden.

$$W^* = \arg\max_W P(X|W) \tag{6.1}$$

Die Suche erfolgt hier unter Berücksichtigung des gewählten Vokabulars um den Berechnungsaufwand zu begrenzen und die Fehlerrate zu minimieren. Für die Erkennung ganzer Sätze oder Wortfolgen werden zusätzlich Sprachmodelle eingesetzt [Mak94, Nat95]. Diese Sprachmodelle sind im wesentlichen Wahrscheinlichkeiten für bestimmte Wortfolgen W:

$$P(W) = \prod_{i=1}^{m} P(w_i|w_{i-N+1}, ..., w_{i-1}). \tag{6.2}$$

$P(w_i|w_{i-N+1}, ..., w_{i-1})$ ist dabei die Wahrscheinlichkeit, dass dem Wort w_i die Wortfolge $w_{i-N+1}, ..., w_{i-1}$ voraus ging. N ist die Kontexttiefe dieses sog. N-Grams. Mit der Viterbi-Dekodierung kann bei der Erkennung das Sprachmodell und die HMM gleichzeitig ausgewertet werden. Der erkannte Satz W^* wird dabei wie folgt bestimmt:

$$W^* = \arg\max_w P(W) \cdot P(X|W) \tag{6.3}$$

Die wesentliche Innovation ist nun die Verwendung eines Textmodells an Stelle eines festen Lexikons und eines optionalen Sprachmodells. Das Textmodell ist im wesentlichen strukturiert wie ein Sprachmodell, beschreibt aber nicht die Wahrscheinlichkeiten von Wortfolgen sondern die von Buchstabensequenzen ($P(c_i|c_{i-N+1}, ..., c_{i-1})$). Diese Beschreibung kann allerdings über eine wesentlich größere Kontexttiefe erfolgen als bei Wort-basierten Sprachmodellen. Während bei Sprachmodellen gewöhnlich Kontexttiefen von zwei oder drei eingesetzt werden, sind bei Textmodellen Kontexttiefen von 12 oder höher realisierbar. Die höheren Kontexttiefen bei Textmodellen beziehen sich allerdings auf die Länge von Buchstabenfolgen, während sich die Kontexttiefe bei Sprachmodellen auf die Wortfolgenlänge bezieht.
Ein systemimmanentes Problem des Viterbi-Dekoders ist der Umgang mit N-Grammen mit großer Kontexttiefe ($N \geq 3$). Um eine effizientere Auswertung solch komplexer Textmodelle zu erzielen, kann z. B. ein *Stack-Dekoder* [Wil98] verwendet werden. Das Textmodell - insbesondere mit hoher Kontexttiefe - macht auf diese Weise ein festes Lexikon überflüssig.

Indem das Textmodell mit entsprechend großer Kontexttiefe über Wortgrenzen hinweg berechnet wird, sind die Auftrittswahrscheinlichkeiten der Wortgrenzen im Textmodell enthalten. D. h., dass sowohl eine Erkennung von Sätzen bzw. Wortfolgen möglich ist, wie auch die Erkennung einzelner Worte. Das potentielle Auftreten von Wortgrenzen wird, wie in Abschnitt 6.2.2 beschrieben, durch eine entsprechende Aufbereitung der Trainingsmenge gesteuert.

6.2 Modellebenen

Wie bereits erwähnt, werden HMM und Textmodelle wie in Glg. (6.3) angegeben, gemeinsam während der Erkennung anhand der beobachteten Handschriftmerkmale X, bzw. im diskreten Fall anhand der vektorquantisierten Merkmale Y, evaluiert. Bei der Dekodierung, bzw. der Erkennung sind daher lediglich die zwei Modellarten Graphemmodell und Textmodell involviert, wie sie in den folgenden Abschnitten beschrieben werden.

6.2.1 Graphemmodelle

Bisher war das Graphem- oder Zeichenmodell als Hidden Markov Modell dargestellt. Anzumerken ist hier, dass statt des HMMs bei diesem Ansatz ebenso ein allgemeines statistisches Graphemmodell verwendet werden kann, welches die bedingte Wahrscheinlichkeit (Likelihood) $P(X|C)$ der Beobachtungsfolge X unter der Bedingung der Wort- bzw. Zeichenfolge C liefert. Das Verfahren ist demnach nicht an die Art der Zeichenmodelle gebunden. Es muß lediglich sichergestellt werden, dass während der Erkennung die verwendeten Graphemmodelle effizient gemeinsam mit dem Textmodell ausgewertet werden können.

6.2.2 Textmodelle

Allgemein formuliert beschreibt ein Textmodell die Wahrscheinlichkeit $P(c_i|c_{i-N+1}, ..., c_{i-1})$ eines Buchstabens c_i unter der Bedingung einer bestimmten ihm vorausgehenden Buchstabenkette oder Historie $c_{i-N+1}, ..., c_{i-1}$ (Glg. (6.2)). Die Struktur ähnelt demnach der eines Sprachmodells, bzw. einer statistischen Grammatik, welche auf Wortfolgen basiert. Mit zunehmender Kontexttiefe vergrößert sich exponentiell die Anzahl der Textmodellparameter. Bei einer Kontexttiefe von $N = 3$ und einem Zeichenvorrat N_c von 80 verschiedenen Zeichen, ergibt sich theoretisch pro Zeichen eine Kontextmenge von $N_c^{N-1} = 80^2 = 6400$ verschiedener Kombinationen. Für eine Kontexttiefe von $N = 6$

erhöht sich dieser Wert bereits auf $80^5 \approx 3,3 \cdot 10^9$. Um die Vielzahl der Parameter zuverlässig zu schätzen, werden solche Modelle i. d. R. auf sehr großen Textdatenbasen trainiert, die mehrere Millionen Sätze beinhalten. Zudem werden zur Glättung der Wahrscheinlichkeiten und zur Vermeidung von *sparse data*-Effekten, *Cut-Off-, Diskontierungs*- und *Back-Off*-Strategien [Cla97] bei der Parameterschätzung der Textmodelle angewendet. Prinzipiell können hier sämtliche Verfahren zur Verfeinerung und Glättung der Textmodelle verwendet werden, wie sie auch von der Optimierung von Sprachmodellen bekannt sind.
Der Unterschied zwischen Textmodellen zur Satz- und zur Worterkennung soll nachfolgend anhand eines Beispiels erläutert werden. Dazu wird angenommen, dass die Trainingsmenge für das Textmodell aus folgendem Satz besteht:

```
Ein Trigramm-Sprachmodell besteht aus relativen
Trigramm-, Bigramm- und Unigramm-Häufigkeiten.
```

Für die Satzerkennung wird jeweils eine gesamte Textpassage verarbeitet. Satzanfang und -ende werden speziell behandelt, was hier nicht weiter vertieft werden soll. Bei einem Trigramm ($N = 3$) ergeben sich für die exemplarische Trainingsmenge beginnend bei dem ersten 'n' folgende Kontextkombinationen:

```
(n|i,E), (<sp>|n,i), (T|<sp>,n), (r|T,<sp>), (i|r,T),
(g|i,r), (r|g,i), (a|r,g), (m|a,r), (m|m,a),
(-|m,m), (S|-,m), (p|S,-), ...
```

Dieses Beispiel macht deutlich, dass bei entsprechend hoher Kontexttiefe statistische Zusammenhänge über Wortgrenzen (hier mit $< sp >$ bezeichnet) hinweg erfasst werden können. Sogar der Einfluß von Satzzeichen und Wortgrenzen wird in diese spezielle Form der statistischen Orthografie einbezogen.
Für den Einzelwortmodus wird die Datenbasis in modifizierter Form zum Training verwendet. Die einzelnen Textpassagen werden dazu an den auftretenden *white spaces* in einzelne Wörter zerlegt:

```
Ein
Trigramm-Sprachmodell
besteht
aus
relativen
Trigramm-,
Bigramm-
und
Unigramm-Häufigkeiten.
```

Unter spezieller Behandlung der Wortanfänge und -Enden werden für jedes einzelne Wort mit anhängenden Satzzeichen die Kontextkombinationen extrahiert und deren Häufigkeiten analysiert. Unter Vernachlässigung der Sonderbehandlung von Wortanfang und -Ende ergeben sich im Vergleich zum ersten Beispiel folgende Kontextkombinationen:

```
(n|i,E), (i|r,T), (g|i,r), (r|g,i), (a|r,g), (m|a,r),
(m|m,a), (-|m,m), ...
```

Ein Wortende ($< sp >$) inmitten einer Äußerung wird somit nicht ermöglicht, wohl aber die zusätzliche Eingabe von Satzzeichen nach den Einzelworten. Die letztendliche Struktur des Textmodells bleibt die gleiche, da es sich hier lediglich um Häufigkeiten, bzw. um Wahrscheinlichkeiten von Kontextkombinationen handelt. In den Erkenner muß bei einer Umschaltung zwischen den beiden Modi lediglich das relevante Textmodell nachgeladen werden.

6.3 Dekodierung

Wie bereits oben angedeutet, stellt die Verarbeitung komplexer N-Gramme eine besondere Herausforderung an den als Erkenner eingesetzten Dekoder. Zur Verarbeitung komplexer N-Gramme bietet sich z. B. der Einsatz eines *Stack-Decoders* an, wie er in [Ren95a, Wil98] beschrieben wurde. Exemplarisch soll nachfolgend eine Übersicht zur Arbeitsweise des Dekoders im zeitsynchronen Modus gegeben werden. Abb. 6.1 stellt den grundlegenden Algorithmus des Stack-Decoders mit zeitsynchroner Stackexpandierung dar. Wie in Schritt a) beschrieben, wird zunächst jedem Zeitpunkt k, zu dem ein Merkmalsvektor existiert, ein

a) Initialisierung des nullten Stacks mit der leeren Hypothese und der Stacks 1 bis K als leere Stacks, $k := 0$
b) Auswahl des Stacks des Zeitpunktes k
c) Falls das Äußerungsende erreicht ist ($k = K$), ist die beste Hypothese des Stacks aus b) die gesuchte Wortfolge
d) Erweitere die Hypothesen des gewählten Stacks um weitere Worte unter Berücksichtigung der bedingten Wortwahrscheinlichkeiten und der Graphemmodelle. Potentielle Wortenden definieren neue Hypothesen, sie werden dem jeweiligen Stack zugefügt.
e) $k := k + 1$
f) weiter bei b)

Abbildung 6.1: Stackalgorithmus mit zeitabhängigen Stacks und zeitsynchroner Stackbearbeitung

leerer Stack zugeordnet. Dieser Stack wird später im Laufe der Dekodierung mit Hypothesen aufgefüllt. Als Hypothese H wird eine Teildekodierung der Beobachtungssequenz X bis zum Zeitpunkt k_H bezeichnet. Die Hypothese H besteht im wesentlichen aus der hypothesenspezifischen Historie der Zeichenfolge C_H und wird als Objekt zusammen mit dem Wahrscheinlichkeitswert der Hypothese S_{C_H} (*Score*) auf dem Stack des entsprechenden Zeitpunktes k_H abgelegt (Abb. 6.2). Der Score S_{C_H} wird entsprechend Glg. (6.2) und (6.3) für Einzelzeichenfolgen berechnet: $S_{C_H} = P(C) \cdot P(X|C)$. Die Auftrittswahrscheinlichkeit $P(X|c)$ für ein einzelnes Zeichen kann z. B. mit einer Viterbi-Approximation und einer zeitsynchronen Strahlsuche [Rab89] bestimmt werden. Die Hypothesen werden auf den jeweiligen Stacks nach ihrem Score in aufsteigender Reihenfolge abgelegt, sodass die wahrscheinlichste Hypothese ($H_i^{(k_H)}$ in Abb. 6.2) oben aufliegt. Anschließend wird ein zu bearbeitender Zeitpunkt ausgewählt (Schritt b)). Ist das Ende der Beobachtungssequenz erreicht (Schritt c)), wird die oberste Hypothese zum Zeitpunkt K als das Erkennungsergebnis ausgegeben. Andernfalls wird mit der Stackexpandierung fortgefahren, indem die Hypothese des nächsten Stacks unter Berücksichtigung von Graphem- und Textmodell um ein Zeichen $c_i^{(k')}$ verlängert wird ($||c_i^{(k')}$). Die so entstandenen neuen Hypothesen werden - entsprechend der geschätzten zusätzlichen Zeichendauer - auf nachfolgenden Stacks abgelegt (Schritt d)). Die Schleife wird anschließend für nachfolgende Stacks wiederholt (Schritte e) und f)). Dabei ist jedoch zu beachten, dass i. A. nicht jeder nachfolgende Stack untersucht werden muß. Es können, wie in Schritt b) angedeutet, gezielt Auswahlmechanismen eingesetzt werden, die den Dekodierungsprozeß wesentlich beschleunigen [Wil98]. So werden z. B. erst die nachfolgenden Stacks selektiert, deren beste Hypothesen einen einstellbar abfallenden Wahrscheinlichkeitswert überschreiten. Außerdem kann davon ausgegangen werden, dass sich in zeitlich benachbarten Stacks überwiegend ähnliche Hypothesen befinden, deren Auswertung vernachlässigt werden kann ohne allzu große Fehler zu erzeugen, was wiederum zu einer beschleunigten Erkennung führt. Zusätzliche Pruningmechanismen werden auch in der Strahlsuche zur Einzelzeichenerkennung eingesetzt, sodass auch auf dieser Ebene eine Beschleunigung erzielt werden kann.

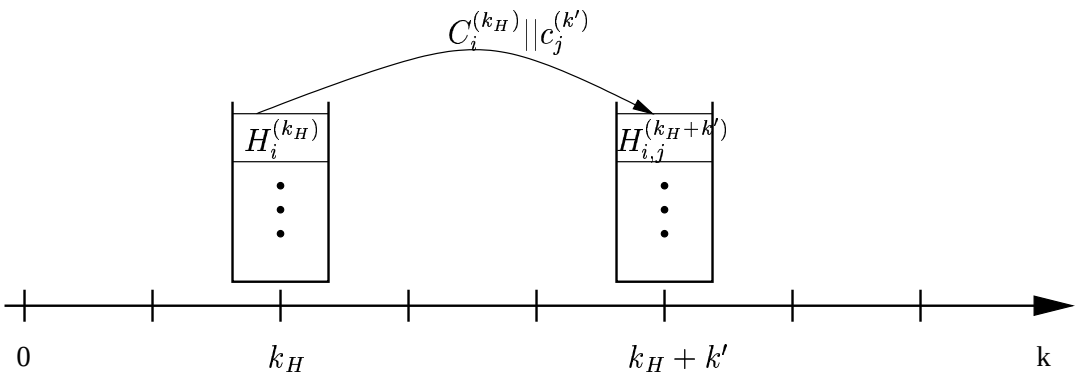

Abbildung 6.2: Anordnung der Stacks und Zuordnung der Hypothesen

6.4 Ergebnisse und Kapitelzusammenfassung

Abschließend bleibt festzuhalten, dass der Vorteil des Stack-Dekoders für diese Aufgabe konzeptionell zu begründen ist, wobei die vielen verschiedenen Pruningmöglichkeiten in adäquater Adjustierung das Dekodierverhalten bezüglich der Laufzeit positiv beeinflussen. Ein wesentlicher Vorteil, den dieses Konzept mit sich bringt, ist die Tatsache, dass der Zugriff auf das Textmodell unabhängig von der Strahlsuche zur Einzelzeichenerkennung erfolgt. Bei einer Expandierung einer Hypothese wird zunächst die Wahrscheinlichkeit der Graphemmodelle $P(X|C)$ zu dem bekannten aktuellen Score multipliziert und abschließend wird der Textmodellanteil aufmultipliziert. Im Vergleich dazu würde der Speicher- und Berechnungsaufwand exponenziell wachsen, wenn derart komplexe N-Gramme direkt in die Strahlsuche integriert würden, da der Suchraum über sämtliche gegebene Kontextkombinationen aufgebaut werden müßte. Hier ließe sich die Auswertung von Graphem- und Textmodell nicht so elegant entkoppeln. Diese Tatsache dürfte im wesentlichen begründen, warum mit einer Strahlsuche maximal Kontexttiefen von $N = 3$ verarbeitet werden können.
Auf der anderen Seite zeigt Tab. 6.1, dass die Ausweitung des betrachteten Kontextes auf mehr als drei Zeichen unerlässlich ist, um die Erkennungsrate zu maximieren. Der Bedarf nach effizienten Dekodierverfahren, die in der Lage sind solche Kontextbereiche zu verarbeiten, wird anhand dieser Ergebnisse deutlich. Während bei dem schreiberunabhängigen System bei einer Kontexttiefe von $N = 3$ die Korrektheit der insgesamt 27025 Einzelzeichen im Testset der 4134 Einzelwörter noch bei 81,3 % liegt, kann dieser Wert bei der Verwendung von 9-Grammen auf 91,4 % gesteigert werden. Bei weiterer Vergrößerung der Kontexttiefe auf 12 nimmt die Erkennungsrate wiederum leicht ab. Die Verbesserung der Akkuratheit, bei der zusätzlich zur Korrektheit (Glg. (3.34)) noch die Anzahl der fehlerhaften Einfügungen berücksichtigt wird, spielt sich in der gleichen Größenordnung ab. Die jeweils etwa 50-prozentige relative Fehlerreduktion bei Erhöhung der Kontexttiefe von 3 auf 9 wird insbesondere bei der Worterkennungsrate relevant. Hier kann die Korrektheit von 39,2 % auf 73,5 % erhöht werden.

	3-Gramm	6-Gramm	9-Gramm	12-Gramm
Korrektheit (27025 Zeichen)	81.3 %	90.6 %	91.4 %	91.3 %
Akkuratheit (27025 Zeichen)	77.1 %	88.4 %	89.3 %	89.2 %
Worterkennungsrate (4134 Wörter)	39.2 %	70.7 %	73.5 %	73.4 %

Tabelle 6.1: Erkennungsergebnisse - verschiedene Kontexttiefen, schreiberunabhängig, Lexikon-frei