Kapitel 5

Strukturoptimierung bei Verwendung kontextabhängiger Modelle

Bezüglich der Modellstruktur sind die verschiedensten Ansätze zur Systemoptimierung denkbar. Dies kann sowohl die Variation der Anzahl der Zustände betreffen, wie auch die Modelltopologie als solche, die durch die Zustandsübergänge vorgegeben wird oder anhand von Trainingsbeispielen gelernt werden kann. Der Schwerpunkt der Betrachtungen liegt in diesem Kapitel jedoch auf der Einführung multipler Modelle. Wie später noch zu verdeutlichen ist, wird mit der Verwendung multipler Modelle eine Strukturoptimierung in der Form erforderlich, dass gewisse Parameter verschiedener Modelle optimal miteinander verknüpft werden müssen.

Bisher wurde stets davon ausgegangen, dass für jedes lexikalische Element *ein* Modell zur Verfügung steht. Als Alternative ist es jedoch auch denkbar, für jedes zu erkennende Zeichen mehrere Modelle einzuführen [Kos99d, Kos97c]. Damit stellt sich zunächst die grundsätzliche Frage, nach welchem Kriterium dies zu erfolgen hat. Betrachtet man dazu das in Abb. 5.1 gezeigte Beispiel und dort insbesondere die markierten mehrfach auftretenden Buchstaben, so wird deutlich, dass selbst bei einem festen Schreiber Realisierungen des selben Buchstabens doch deutlich voneinander abweichen können. Offensichtlich handelt es sich hier um den Einfluß der benachbarten Zeichen. Dieser Effekt kann dadurch erklärt werden, dass der Schreiber versucht, während des Schreibprozesses eine möglichst effiziente Stiftführung zu erzielen und dabei gewisse Inkonsistenzen innerhalb der selben Zeichenklasse auftreten. Das erste auftretende ’e’ im Kontext von ’t’ und ’x’ z. B. ist im Vergleich mit dem zweiten ’e’ (im Kontext von ’k’ und ’i’) deutlich ausgeprägter. Auch liegt der Anfangspunkt des ersten ’e’ höher als bei der zweiten Realisierung. Insgesamt läßt sich bei der ersten Variante eine deutlichere Separation von den benachbarten Zeichen feststellen. In gleicher Form ließen sich offensichtliche Differenzen bei den übrigen mehrfach vorkommenden Zeichen aufzählen.

Dem könnte entgegenhalten werden, dass möglicherweise mehr Gemeinsamkeiten inner-

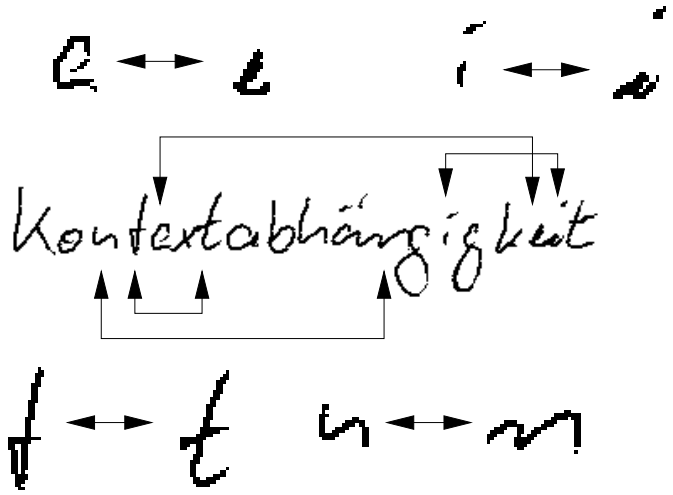


Abbildung 5.1: Kontextbedingte Graphemvariationen

halb einer Zeichenklasse existieren als Unterschiede. Ein Beispiel dazu wären sicherlich die in Abb. 5.1 gezeigten 'g', sowie das zweite und dritte 't', welche jeweils über sehr ähnliche Charakteristika verfügen. Dabei muß man sich allerdings verdeutlichen, in welcher Form die Schrift dem Erkennungssystem präsentiert wird. Die extrahierten Merkmale - insbesondere die gleitende Bitmap - erfassen neben dem lokalen Ausschnitt der Trajektorie auch einen bestimmten Umgebungsbereich (vergl. Abb. 3.7). Zeitlich betrachtet geschieht dies zudem global. Am Beispiel der gezeigten 'g' bedeutet dies, dass selbst bei scheinbar großer Ähnlichkeit erhebliche Unterschiede in der Merkmalssequenz vorliegen. Bei der ersten Realisierung wird durch die gleitende Bitmap ein Teil des vorherigen 'n' erfasst und als Merkmal mitgeführt. Sowohl die kontextbedingten Einflüsse auf die Form der Trajektorie wie auch die kontexterfassende Mermalssequenz der Bitmap sollten sich natürlich um so deutlicher auswirken, je stärker die Buchstaben untereinander verbunden sind (kursive Schreibweise).
Aufgrund dieser Betrachtungen scheint es also naheliegend zusätzliche Modelle für ein Zeichen - in Abhängigkeit des jeweiligen Kontextes - einzuführen. In Anlehnung an die in der Spracherkennung übliche Terminologie für kontextabhängige Modelle wird diese Form von Modellen nachfolgend als Allographeme, bzw. als Bi- und Trigrapheme bezeichnet.

5.1 Bi- und Trigrapheme

Zentrales Problem bei der Einführung von Allographemen ist die große Parameteranzahl der Menge aller Modelle. Bei der Verwendung von N-Graphemen wird ein Kontext von N-1 benachbarten Zeichen berücksichtigt. Dies bedeutet, dass bei einem Grundzeichensatz bestehend aus 80 Modellen letztendlich 80^N Zeichenkombinationen vorhanden sind. Weiterhin muß davon ausgegangen werden, dass die Trainingsdatenbasis und damit die Anzahl

der Trainingsbeispiele nicht beliebig erhöht werden kann oder gar fest vorgegeben ist. Diese beiden Randbedingungen (stark steigende Parameteranzahl und feste Trainingsdatenbasis) erfordern demnach einen sinnvollen Kompromiß zwischen einer möglichst hohen Genauigkeit der Modelle und deren hinreichender Trainierbarkeit [Lee90]. Die Trainierbarkeit ist gegeben, wenn jeder Modellparameter anhand ausreichenden Trainingsmaterials geschätzt werden kann.

Der Kompromiß zwischen Trainierbarkeit und Genauigkeit läßt sich zuerst durch einige naheliegende Maßnahmen anstreben. Bei einer betrachteten Kontexttiefe von N sollte man sich zunächst vor Augen führen, dass viele der 80^N theoretisch möglichen Zeichenkombinationen äußerst ungebräuchlich sind. Darum ist es sinnvoll die Expansion von Monographemen zu Allographemen auf der Basis eines festen Lexikons, bestehend aus gültigen und gebräuchlichen Wörtern, vorzunehmen. Bei der Verwendung sehr großer Lexika in Kombination mit einer hohen berücksichtigten Kontexttiefe kann es dennoch im Verhältnis zur Trainingsdatenbasis zu einer unangemessenen Überhöhung der Modell- und Parameteranzahl kommen. Da die Modellanzahl also bei wachsender Kontexttiefe überproportional ansteigt, gleichzeitig aber der Einfluß weiter entfernter Zeichen abnimmt, ergibt sich als guter Kompromiß mit handhabbarer Modellanzahl eine maximale Kontexttiefe von $N = 3$.

Tabelle 5.1 zeigt neben der Graphemfolge die Entwicklung der Modellanzahl bei variierender Kontexttiefe von $N = 1, 2, 3$. Die Notation bei der Graphemisierung der Lexikoneinträge gibt Auskunft über das zugrundeliegende Modell und den berücksichtigten Kontext.

		Monographeme	Bigrapheme (li.)
Graphemisierung		/K/ /o/ /n/ **/t/** /e/ /x/ **/t/**	/K/ K/o/ o/n/ **n/t/** t/e/ e/x/ **x/t/**
Anzahl Modelle	1k-Lexikon	80	800
	30k-Lexikon	80	1300
	200k-Lexikon	80	2800
	kein Lexikon	80	6400
		Bigrapheme (re.)	Trigrapheme
Graphemisierung		/K/o /o/n /n/t **/t/e** /e/x /x/t **/t/**	/K/o K/o/n o/n/t **n/t/e** t/e/x e/x/t **x/t/**
Anzahl Modelle	1k-Lexikon	800	3000
	30k-Lexikon	1300	8900
	200k-Lexikon	2800	25000
	kein Lexikon	6400	512000

Tabelle 5.1: Modellexpansion bei Verwendung von Allographemen am Beispiel des Wortes 'Kontext'

’/t/’ steht in diesem Zusammenhang für das kontextunabhängige HMM (Monographem) des Buchstabens ’t’. Für den Fall der links-kontextabhängigen Bigrapheme würden für den Buchstaben ’t’ in dem Wort ’Kontext’ zwei HMM eingeführt werden: n/t/ mit dem linken Kontextbuchstaben ’n’ und x/t/ mit dem linken Nachbarn ’x’. Entsprechend würden bei der Verwendung rechts-kontextabhängiger Modelle die HMM /t/e und /t/ gebildet werden. Für Trigrapheme schließlich ergäben sich in analoger Weise die Modelle n/t/e und x/t/ (jeweils in Fettdruck markiert).

5.2 Parameterreduktion

Auch unter der Voraussetzung, dass bei der Erkennung ein festes Lexikon eingesetzt wird, und damit die Kombinationsvielfalt kontextueller Modelle eingeschränkt wird, zeigen die vorangehenden Betrachtungen, dass die hinreichende Trainierbarkeit kontextabhängiger Modelle damit allein nicht unbedingt gegeben sein muß. Im Falle der Trigrapheme werden auf der Grundlage eines 200000 Wörter umfassenden Lexikons statt der ursprünglichen 80 HMM nun 25000(!) HMM generiert.
Jedes dieser HMM besteht weiterhin aus 12 Zuständen. Pro Zustand wiederum setzen sich die diskreten Verteilungen aus insgesamt 142 Parametern zusammen. Insgesamt besteht ein Trigraphem-System also aus $12 \cdot 142 = 1704$ Verteilungsparametern zuzüglich der 24 (pro Zustand ein Selbstübergang und ein Übergang zum Nachfolgezustand) von Null verschiedenen Transitionswahrscheinlichkeiten. Dies bedeutet in der Summe, dass für den Aufbau eines Trigraphem-Systems 42600000 Parameter zu schätzen sind. Dem stehen in der WD-Datenbasis (Abschnitt B.1) ca. 500000 Merkmalsvektoren zur Verfügung. Diese deutliche Diskrepanz zeigt klar die Notwendigkeit die Anzahl der Modellparameter weiter zu reduzieren.
Dazu sind im Prinzip zwei verschiedene Ansätze denkbar, die schließlich auf der Basis von Trigraphemen verglichen werden. Neben dem selektiven Ansatz [Kos97b, Kos97a], der in Abschnitt 5.2.1 beschrieben wird, sind darüber hinaus noch Verfahren zur Parameterverknüpfung (Tying) denkbar [Lee90]. Das Grundprinzip der Parameterverknüpfung ist in Abschnitt 5.2.2 erklärt. Spezielle Clustering-Algorithmen dazu werden in den Abschnitten 5.3 und 5.4 beschrieben. Da das *sparse-data* Problem insbesondere bei Trigraphemen auftritt, diese aber den Vorteil haben, beidseitig den Kontext zu berücksichtigen, werden die verschiedenen Verfahren schließlich anhand von Trigraphemen evaluiert.

5.2.1 Selektives Verfahren

Um die höhere Konsistenz von Trigraphemen auszunutzen, muß also die Parameteranzahl zum Teil deutlich reduziert werden. Dazu ist es zunächst sinnvoll die Häufigkeiten der Trigrapheme zu betrachten. Abb. 5.2 zeigt die Verteilung der Trigraphem-Häufigkeiten in der Trainingsdatenbasis auf der Grundlage des 1K-Lexikons. N bezeichnet dabei die Anzahl der Trainingsbeispiele für ein HMM. $M(N)$ ist die Anzahl der Trigrapheme, welche durch genau N Trainingsbeispiele in der Trainingsdatenbasis repräsentiert sind. Erstaunlich ist hier, dass - selbst bei einem mittelgroßen Vokabular von 1000 Wörtern - die meisten erzeugten Trigrapheme absolut unterrepräsentiert sind. So existieren 310 Trigrapheme, die lediglich anhand von zwei Beispielen ($N = 2$) in der Trainingsmenge trainiert werden könnten. Für $N = 1$ würden sogar 860 HMM existieren und für $N = 0$ nochmals 300 Modelle. Diese 300 'ungesehenen' Trigrapheme resultieren daher, dass das Lexikon über Worteinträge verfügt, die ausschließlich für den Test benötigt werden, wofür es aber keine Trainingsbeispiele gibt. Zusammen würden also bereits rund die Hälfte (1470 von insgesamt 3000) aller erzeugten Trigrapheme mit keinem, einem oder nur anhand von zwei Beispielen trainiert werden.
Anhand dieser Verteilungen läßt sich nun eine minimal erforderliche Häufigkeit definieren, anhand derer die Generierung von Trigraphemen gesteuert werden kann. Dazu wird zunächst eine Expansion des Monographem-Lexikons auf Trigrapheme durchgeführt mit einer anschließenden Berechnung der Häufigkeitsverteilung. Die Kontextkombinationen deren Häufigkeit der Trainingsbeispiele N einen Schwellwert N_{min} überschreitet werden schließlich als Trigrapheme eingeführt.
Mit einer hinreichend großen Wahl von N_{min} kann somit die Trainierbarkeit jedes einzelnen Modells garantiert werden. Ein Nachteil dieses Verfahrens ist, dass die erzeugten Trigraph-

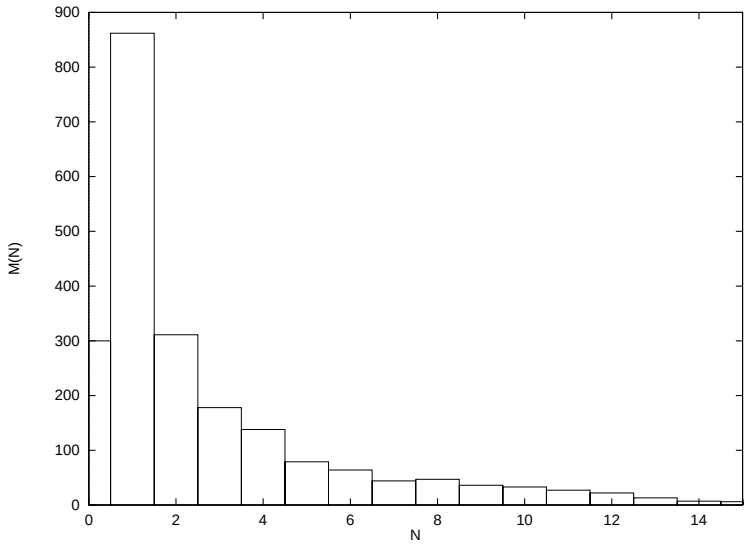


Abbildung 5.2: Trigraphem-Häufigkeiten der Trainingsdatenbasis (1K-Lexikon)

eme keine Generalisierungsfähigkeiten aufweisen, da sie für eine ganz bestimmte Kontextkombination vorgesehen sind. Es kann also der Fall eintreten, dass die Erzeugung zweier Trigrapheme, z. B. n/a/c und m/a/c, aufgrund unzureichenden Trainingsmaterials zurückgewiesen wird, obwohl deren Kontextkombination zu sehr ähnlichen Realisierungen führt. Liegt eine gewisse Ähnlichkeit vor, würde dies wiederum bedeuten, dass beide Trigrapheme mit Realisierungen aus beiden Kontextkombinationen trainiert werden könnten. Die Parameter vieler gering okkupierter, jedoch ähnlicher Grapheme könnten gemeinsam und damit robuster geschätzt werden, als dies bei separaten Trainingsbeispielen der Fall wäre.

5.2.2 Parameter-Tying

Das Training verschiedener HMM anhand identischer Trainingsbeispiele führt natürlich zu absolut identischen Modellparametern. Als Konsequenz dazu können diese Trigrapheme ebenso zu einem einzigen generalisierten Trigraphem vereinigt werden. Dies führt bei dem genannten Beispiel zu einem Trigraphem $\{$m,n$\}$/a/c, bei dem der rechte Kontext des Buchstabens 'a' ein 'c' ist, und der linke Kontext entweder aus einem 'm' oder einem 'n' bestehen kann. Führt man diese Idee weiter, gelangt man zu Strukturen, wie sie in Abb. 5.3 gezeigt sind. Die gestrichelt dargestellten Zustände bedeuten logische Zustände, die im Prinzip lediglich aus einem Zeiger auf physikalische Zustände bestehen. Solche Überkreuz-Verknüpfungen ermöglichen die gemeinsame Parameternutzung über verschiedene Trigrapheme des gleichen Basisgraphems, wobei die Anfangszustände des Modells d/e/s mit einem völlig anderen Modell verknüpft werden, als dessen Endzustände.
Verschiedene Allographeme des gleichen Stammgraphems benutzen damit einen Pool von Parametern (insbesondere Zustände) gleichzeitig und bilden so eine Form verallgemeinerter Allographeme. Diese verallgemeinerten Allographeme sollten dann so strukturiert werden, dass sie ggf. bei vorhandenem Trainingsmaterial spezielle Graphemcharakteristika abbilden können oder aber die zuverlässig schätzbaren Parameter ähnlicher Allographeme nutzen können. Konkret bedeutet dies, dass Allographeme mit gleichem Stammgraphem - im Beispiel in Abb. 5.3 das /e/ - über die selbe Transitionsmatrix verfügen. Diese Transitionsmatrix wird dem entsprechend auch anhand aller Trainingsbeispiele $*/e/* = /e/$ geschätzt.
Das Tying kann sowohl auf die Transitionsmatrizen angewendet werden, wie auch auf einzelne merkmalsspezifische Verteilungen verschiedener Zustände oder gar auf ganze HMM-Zustände. Geht man davon aus, dass insbesondere innerhalb einer Graphemklasse gewisse Korrelationen in den Merkmalen auftreten und diese sich wiederum in den verschiedenen Verteilungen widerspiegeln, ist es angemessen das Tying allgemein auf ganze HMM-Zustände zu beziehen. Vor allem im Zusammenhang mit großen und sehr großen Lexika und der damit einhergehenden großen Anzahl von Parametern erfordert das Tying selbstorganisierende Clustering-Verfahren, wie das datengetriebene State-Clustering oder das Entschei-

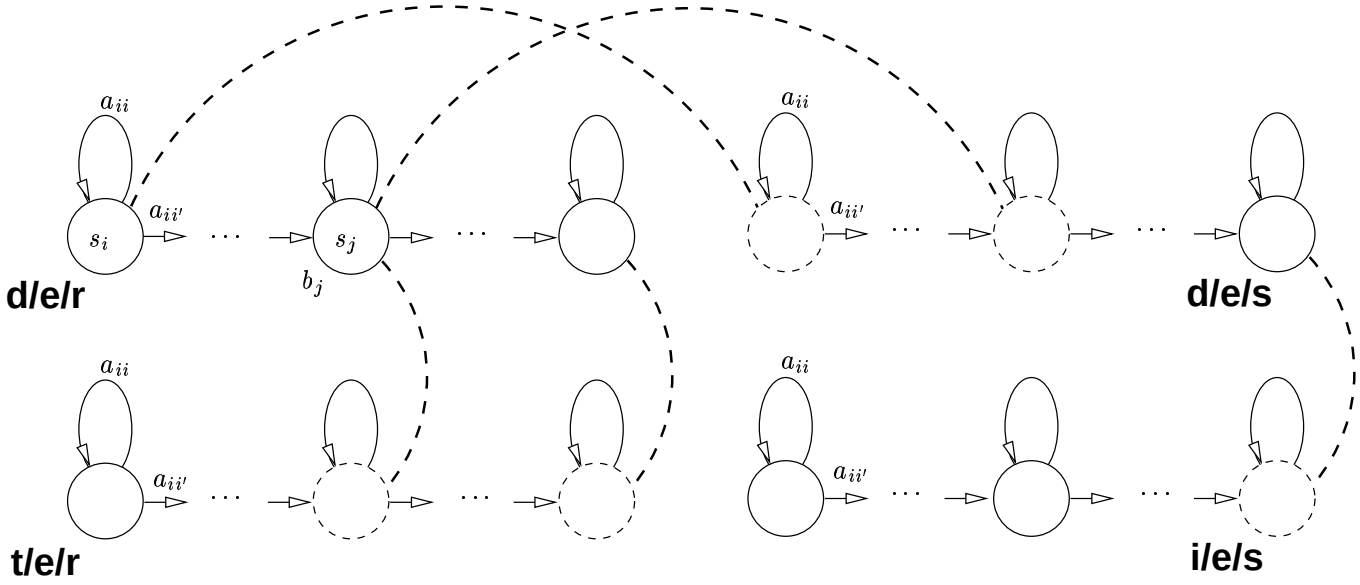


Abbildung 5.3: Parameterverknüpfung bei Trigraphemen

dungsbaum basierte Clustering.

5.3 Datengetriebenes State-Clustering

Die Grundidee des datengetriebenen Clusterings ist es, Zustände aufgrund eines gewissen Abstandsmaßes zusammenzufassen [You92, You93]. Da es sich hier um diskrete Wahrscheinlichkeitsdichtefunktionen handelt, kann die Ausgabeverteilung als Vektor der Dimension $N_C^{(z)}$ betrachtet werden, wenn das Codebuch des z-ten Merkmalsstromes die Größe N_C hat. Die einzelnen Vektorelemente entsprechen demnach den zustandsbedingten Wahrscheinlichkeiten einzelner VQ-Partitionen. Ein Vergleich verschiedener Verteilungen b_i und b_j des selben Merkmalsstromes z in den Zuständen i und j kann dann anhand des Euklid'schen Abstandes zwischen den Ausgabeverteilungen (bzw. Vektoren) b_i und b_j erfolgen:

$$d(i,j) = \sum_{n=1}^{N_C} (b_i(n) - b_j(n))^2. \tag{5.1}$$

Das Abstandsmaß der Ausgabeverteilungen kann dabei die direkte Auftrittswahrscheinlichkeit $P(n)$ der Partition n eines Vektorquantisierers berücksichtigen, wie auch deren logarithmierte Form $\log P(n)$. Für den gebräuchlicheren Fall der logarithmierten Wahrscheinlichkeiten ergibt sich somit

$$d(i,j) = \sum_{n=1}^{N_C} (\log\{P_i(n)\} - \log\{P_j(n)\})^2. \tag{5.2}$$

Weiterhin ist zu berücksichtigen, dass verschiedene Merkmalsströme unterschiedliche Codebuchgrößen aufweisen. Soll nun das Tying anstatt auf einzelne Verteilungen auf ganze

HMM-Zustände angewendet werden, ist die Codebuchgröße $N_C^{(z)}$ des Vektorquantisierers z zu berücksichtigen, sodass sich der Gesamtabstand zweier Zustände i und j mit Z unabhängigen und ggf. unterschiedlich dimensionierten Codebüchern wie folgt formulieren läßt:

$$d(i,j) = \frac{1}{Z}\sum_{z=1}^{Z}\sqrt{\frac{1}{N^{(z)}}\sum_{n=1}^{N_C^{(z)}}(\log\{P_i^{(z)}(n)^{\gamma_z}\} - \log\{P_j^{(z)}(n)^{\gamma_z}\})^2}. \tag{5.3}$$

Unter Einbeziehung der Merkmalsgewichtungen γ_z lassen sich für das Clustering zusätzlich noch Gewichtungen der einzelnen Merkmalsströme berücksichtigen. Als Standardeinstellung gilt jedoch $\gamma_z = 1$.

Zur Initialisierung wird zunächst jedes Cluster mit genau einem Zustand besetzt. Um festzustellen welche Cluster zusammengefasst werden, muß zuerst für jedes Cluster-Paar aus den Zustandsabständen nach Glg. 5.3 ein Clusterabstand ermittelt werden. Der Abstand zweier Cluster ist die maximale Distanz, die beliebige Zustände innerhalb der beiden Cluster untereinander aufweisen. D. h. dass die Distanz zweier Cluster i und j, welche die Zustände k und l beinhalten aus der Maximumsuche unter den Abständen aller beteiligter Zustandsabstände ermittelt werden kann. Der Maximalabstand

$$d(k^*, l^*) = \max_{k,l} d(k,l). \tag{5.4}$$

zwischen den Zuständen k^* und l^* führt somit zu der gesuchten Distanz d_C zwischen den Clustern i und j mit

$$d_C(i,j) = d(k^*, l^*). \tag{5.5}$$

Eine anschließende Minimumsuche

$$(i^*, j^*) = \arg\min_{i,j} d_C(i,j) \tag{5.6}$$

unter den Abständen d_C aller Cluster-Paare i, j führt schließlich auf die zu verschmelzenden Cluster i^* und j^*. Abb. 5.4 illustriert diese Vorgehensweise.

Würde dieser Algorithmus sukzessive in der Form weitergeführt werden, hätte dies zur Folge, dass am Ende wieder alle Zustände in einen Cluster kollabieren würden. Darum ist es zudem erforderlich, eine Abbruchbedingung einzubringen. Dies läßt sich z. B. in der Form realisieren, dass eine minimale Anzahl verbleibender Cluster vorgegeben wird. Da dies aber nicht unbedingt eine Vorhersage über die Clustergröße zuläßt, kann alternativ dazu eine maximale Clusterdistanz $d_{C,max}$ definiert werden, bei deren Erreichen der Clustering-Algorithmus abgebrochen wird. Weiterhin kann das Clustering in der Weise verfeinert werden, dass in einem letzten Durchlauf besonders gering frequentierte Cluster entsprechend des definierten Abstandsmaßes mit ihrem jeweils nächsten Nachbarn zusammengefasst werden. In Kombination sorgen beide Maßnahmen für eine ausbalancierte Clustergröße, wodurch ein guter Kompromiß zwischen Trainierbarkeit und Genauigkeit erreicht werden kann.

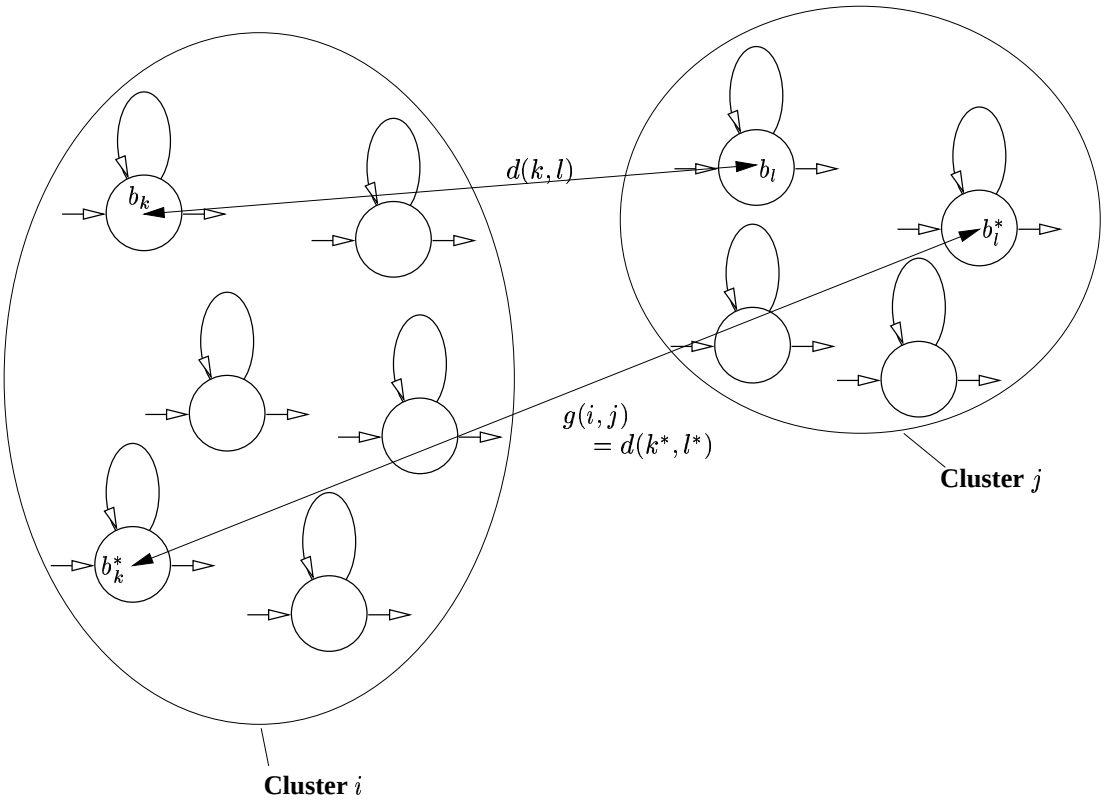


Abbildung 5.4: Datengetriebenes State-Clusterings

Nachdem bekannt ist, zu welchen Gruppen die HMM-Zustände jeweils gehören, können diese nun miteinander verschmolzen werden. Sämtliche Zustände innerhalb eines Clusters greifen dann auf einen repräsentativen Zustand zu, der die Eigenschaften aller beteiligten Zustände vereinigen sollte. Dies wird dadurch realisiert, dass ein beliebiger Zustand aus dem Cluster als physikalischer Zustand beibehalten wird, während alle anderen Zustände lediglich als Zeiger auf diesen einen physikalischen Zustand weiter geführt werden. Per Definition sind alle in einem Cluster enthaltenen Zustände und insbesondere deren Verteilungen recht ähnlich, wodurch diese vereinfachte Auswahl des physikalischen Zustandes möglich ist. Zudem werden nach dem Clustering die Modelle mit einigen Iterationen des Forward-Backward Algorithmus nachtrainiert. Die geclusterten Zustände werden dabei auf Grund der Verknüpfung von logischen und physikalischen Zuständen automatisch mit all den Zustands-bezogenen Beispielen trainiert die dem gesamten Cluster zuzuordnen sind. Damit wird schließlich garantiert, dass der den Cluster repräsentierende physikalische Zustand alle Charakteristika der beteiligten Zustände - in Abhängigkeit der jeweiligen Frequentierung - berücksichtigt.
Der Algorithmus zum datengetriebenen Zustands-Clustering läßt sich also in folgender Form zusammenfassen:

1. Erzeuge n Cluster für n Zustände

2. Suche ein (i, j) mit $(i, j) \in \{1, 2, \cdots, n\}$ welches $d_C(i, j)$ minimiert

3. Falls $d_C(i, j) < d_{C,max}$ und $n > 1$

   (a) verschmelze Cluster i mit Cluster j

   (b) dekrementiere n

   (c) weiter mit 2.

Theoretisch sind mit diesem Verfahren Verknüpfungen von Verteilungen oder Zuständen über verschiedene Zustandspositionen oder sogar Graphemklassen möglich. Da sich aber i. A. die diskriminativen Eigenschaften der Modelle verschlechtern, wenn deren Parameter über Graphembasisklassen hinweg verknüpft werden, ist es sinnvoll das Clustering nur auf verschiedene Modelle des gleichen Basisgraphems zu beschränken.
Mit dem datengetriebenen Clustering sollen die Ähnlichkeiten von Zuständen gefunden werden. Da sich rechtsseitige (linksseitige) Kontexteinflüsse insbesondere an den Modellanfängen (Modellenden) widerspiegeln, ist es zudem naheliegend, nur die Zustände zu verknüpfen, die an gleicher Position innerhalb verschiedener Grapheme des gleichen Basistyps liegen. Wie das Beispiel in Abb. 5.3 zeigt, werden nur Zustände in Position 1 mit anderen Zuständen in Position 1 verknüpft, während potenziell nur Zustände der Positionen 2 miteinander verknüpft werden usw.
Wenngleich dieses Verfahren bereits eine elegante Möglichkeit zur Konstruktion verallgemeinerter Allographeme darstellt, bleibt eine wesentliche Frage zunächst offen: was geschieht mit den kontextabhängigen Modellen, für die keine (oder nur sehr wenig) Trainingsdaten in entsprechender Kontextkombination existieren? Die Antwort auf diese Frage wird im nachfolgenden Abschnitt gegeben.
Um dies jedoch zu klären, sollte zunächst einmal der gesamte Ablauf zur Erzeugung geclusterter Allographeme zusammengefasst werden.

- Ausgehend von vollständig trainierten Monographemen werden auf Basis des später bei der Erkennung einzusetzenden Lexikons die Monographeme in entsprechende Allographeme kopiert. Praktischer Weise kann bereits hier ein Tying der Transitionsmatrizen von Allographemen des gleichen Basistyps erfolgen.

- Es folgen einige Iterationen des Forward-Backward-Algorithmus auf den Allographemen.

- Die Allographeme werden geclustert.

- Es folgen erneut einige Iterationen des Forward-Backward-Algorithmus auf den geclusterten Allographemen.

Die Expansion von Monographemen zu Allographemen wird zunächst ohne Kenntnis der Trainingsdaten und der damit verbundenen Graphemhäufigkeiten durchgeführt. Bei den nachfolgenden Forward-Backward Iterationen werden die Parameter der Allographeme mit unzureichendem oder fehlendem Trainingsmaterial unverändert mitgeführt und in dieser Form auch in den Clustering-Prozeß eingebracht. D. h., verglichen mit Monographemen ist der Gewinn an Genauigkeit bezogen auf die Testmenge auf solche Allographeme beschränkt, die sich in einer Schnittmenge aus Trainings- *und* Testdaten befinden. Sicherlich läßt sich annehmen, dass die am häufigsten vorkommenden Graphemkombinationen auch in der Trainingsmenge wiederzufinden sein müßten. Dennoch nimmt das Verhältnis von Schnittmenge aus Trainings- und potenziellen Testgraphemen zur Gesamtanzahl potenzieller Testgraphemen bei wachsendem Lexikon stark ab (vergl. Tab. 5.1).

5.4 Entscheidungsbaum basiertes State-Clustering

Es zeigt sich also, dass es bei der Verwendung kontextabhängiger Modelle in Verbindung mit einem sehr großen Vokabular vorteilhaft ist, die Vorzüge kontextueller Modellierung bei vorgegebener Trainingsmenge auch auf ungesehene Kontextkombinationen zu übertragen. Ein vielversprechender Ansatz dafür ist das Entscheidungsbaum-basierte State-Clustering [You94]. Mit der Verwendung von Entscheidungsbäumen wird es möglich, in kodierter Form a priori Wissen über den Einfluß von Kontextkombinationen auf die Graphemrealisierung in den Clustering-Prozeß einzubringen.
Verglichen mit dem datengetriebenen Clustering geht man hier genau einen entgegengesetzten Weg: Anstatt ähnliche Zustände zusammenzufassen, wird durch ein *split and merge*-Algorithmus versucht, möglichst unterschiedliche Zustände zuerst zu trennen und am Schluß die in einem Cluster verbleibenden Zustände zu verschmelzen (Abb. 5.5). Man nimmt dazu zunächst an, dass alle Zustände s in einem Cluster zu einem Zustand zusammengefasst werden. Anhand von Fragen q, die bestimmte Kontextgrapheme c der in einem Cluster enthaltenen Zustände betreffen, werden die Zustände dann in zwei Unter-Cluster aufgeteilt.
Eine solche aufspaltende Frage könnte z. B. lauten: 'Ist der rechte Kontext ein $/E/$ oder ein $/F/$?'. Zustände, für die eine angewendete Frage mit 'ja' ('nein') zu beantworten ist, werden den Teil-Clustern $\mathcal{S}^{(\mathcal{J})}$ (bzw. $\mathcal{S}^{(\mathcal{N})}$) zugeordnet. Der entstandene Cluster $\mathcal{S}^{(\mathcal{J})}$ enthält Zustände aus Graphemen die den nachgefragten Kontext c aufweisen. Der Cluster $\mathcal{S}^{(\mathcal{N})}$ nimmt alle übrigen Zustände der Menge $\mathcal{S}$ auf, d. h. die Zustände, deren Allographeme nicht den Kontext c besitzen. Nach wiederholter Aufteilung der entstehenden Unter-Cluster werden die in einem Ausgangsknoten des Baumes verbleibenden Zustände verschmolzen.
Jede Aufteilung sollte so vorgenommen werden, dass möglichst unterschiedliche Zustände getrennt werden. Nimmt man an, dass alle in einem Cluster enthaltenen Zustände zu einem

repräsentativen Zustand $\bar{s}$ verknüpft werden, bedeutet jede Aufsplittung des Clusters $\mathcal{S}$ eine vergrößerte Anzahl Modellparameter. Diese größere Anzahl führt zu einer detaillierteren Abbildung der Trainingsdaten und damit i. d. R. auch zu einer erhöhten Likelihood der Modelle auf den zugrunde liegenden Trainingsdaten. Um die Relevanz einer Aufsplittung und der damit verbundenen Frage q zu bewerten, eignet sich daher der log-Likelihood-Gewinn ΔL_q, der durch die Aufsplittung des Clusters $\mathcal{S}$ durch Frage q erzielt wurde.

$$\Delta L_q = L(\mathcal{S}^{(J)}(q)) + L(\mathcal{S}^{(N)}(q)) - L(\mathcal{S}) \tag{5.7}$$

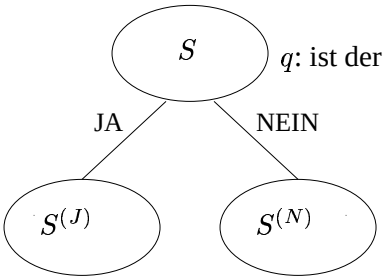
Die Bewertungsfunktion $L(\mathcal{S})$ eines Clusters ist die bedingte logarithmierte Wahrscheinlichkeit $\log p(X|\mathcal{S})$ dafür, dass die aus K Abtastwerten bestehende Trainingsmenge X durch das Cluster $\mathcal{S}$ erzeugt wurde. Dabei ist es zunächst unerheblich, ob X aus kontinuierlichen, oder aus diskreten, vektorquantisierten Merkmalssequenzen besteht.
Natürlich hängt $L(\mathcal{S})$ maßgeblich von den beteiligten Zuständen s ab. Wären innerhalb eines Clusters alle Zustände in gleichem Maße relevant, ließe sich $L(\mathcal{S})$ durch Summation der einzelnen bedingten log-Wahrscheinlichkeiten $\log p(X|s)$ bestimmen:

$$L(\mathcal{S}) = \sum_{s \in \mathcal{S}} \gamma \cdot \log p(X|s). \tag{5.8}$$

Dem Faktor γ kommt dabei die Rolle einer (konstanten) Gewichtung der beteiligten Wahrscheinlichkeiten $p(X|s)$ zu. Verallgemeinert man die Zusammensetzung des Clusters in der Form, dass die beteiligten Zustände unterschiedliche Relevanz für $L(\mathcal{S})$ einbringen können, kann die konstante Gewichtung durch eine zustandsabhängige Gewichtung γ_s ersetzt werden. $L(\mathcal{S})$ ließe sich dann approximieren durch

$$L(\mathcal{S}) = \sum_{s \in \mathcal{S}} \gamma_s \log p(X|s) = \sum_{s \in \mathcal{S}} \gamma_s \sum_{k=1}^{K} \log p(x_k|s). \tag{5.9}$$

Naheliegend ist es, die zustandsabhängige Gewichtung γ_s über die jeweilige Zustandsfrequentierung zu definieren [You94]. Unter der Voraussetzung, dass die Segmentgrenzen der



q: ist der rechte Kontext ein $/E/$ oder $/F/$?

$S^{(J)}$: Zustände aus Modellen, deren rechter Kontext ein $/E/$ oder ein $/F/$ ist

$S^{(N)}$: Zustände aus Modellen, deren rechter Kontext kein $/E/$ und kein $/F/$ ist

Abbildung 5.5: Cluster-Splitting

Zustände durch das Clustering quasi nicht verschoben werden, ließe sich die jeweilige Zustandsfrequentierung aus einem - dem Clustering vorausgehenden - Viterbi-Alignment bestimmen. Als $\gamma_s(x_k)$ kann damit die Wahrscheinlichkeit betrachtet werden, dass das Merkmal x_k von dem Zustand s erzeugt wurde. Unter Anwendung der Viterbi-Segmentierung ergäbe sich ein $\gamma_s(x_k)$ von 1 oder 0, abhängig davon, ob ein Frame x_k dem Zustand s zugeordnet wurde oder nicht.
Führt man sich den gesamten allgemeinen Ablauf des Clusterings vor Augen (vergl. Abschnitt 5.3), ergibt sich eine sehr elegante Alternative zu dem zusätzlichen (relativ teuren) Viterbi-Alignment: Da vor dem eigentlichen Clustering ohnehin einige Iterationen des Forward-Backward-Algorithmus durchgeführt werden, sollte es doch möglich sein, die Frame-Zustands-Zuordnung aus der letzten Iteration des FB-Algorithmus vor dem Clustering zu nutzen. Damit verließe man den deterministischen Standpunkt, den eine Viterbi-Segmentierung mit sich brächte und erhielte so 'echte', d. h. kontinuierliche Wahrscheinlichkeitswerte für $\gamma_s(x_k)$ zwischen $0,0$ und $1,0$.
Werden die Wahrscheinlichkeiten $\gamma_s(x_k)$ nun über alle K Abtastwerte aufsummiert, ergibt sich die Zustandsfrequentierung γ_s mit

$$\gamma_s = \sum_{k=1}^{K} \gamma_s(x_k) \tag{5.10}$$

für den Zustand s, mit der Normierungsbedingung

$$\sum_{s} \gamma_s(x_k) = 1, \tag{5.11}$$

wobei über alle Zustände s des Gesamtsystems summiert wird. D. h. γ_s ist letztlich die Häufigkeit mit der Zustand s besucht wurde. Wird die log-Likelihood des Clusters $\mathcal{S}$ durch

$$L(\mathcal{S}) = \sum_{s \in \mathcal{S}} \log p(X|s) \sum_{k=1}^{K} \gamma_s(x_k) = \sum_{s \in \mathcal{S}} \sum_{k=1}^{K} \log p(x_k|s) \gamma_s(x_k) \tag{5.12}$$

geschätzt, ergibt sich für den Fall kontinuierlicher Merkmale und Ausgabeverteilungen als weitere Vereinfachung, dass $L(\mathcal{S})$ mit

$$L(\mathcal{S}) = -\frac{1}{2}(\log[(2\pi)^n |\Sigma(\mathcal{S})|] + n) \sum_{s \in S} \sum_{k=1}^{K} \gamma_s(x_k), \tag{5.13}$$

nur noch von der Varianz $\Sigma(\mathcal{S})$ des Clusters $\mathcal{S}$, von $\gamma_s(x_k)$ und von der Dimensionalität n der Merkmalsvektoren abhängt [You94]. Der Wert für die Zustandsfrequentierung γ_s entsprechend Glg. 5.10 läßt sich für jeden Zustand s aus der vorangegangenen FB-Iteration aus den Vorwärts- und Rückwärtsvariablen berechnen und temporär speichern. $\Sigma(\mathcal{S})$ wiederum kann aus den Mittelwerten und Varianzen der beteiligten Zustände unter Berücksichtigung von γ_s bestimmt werden, wodurch eine effiziente Berechnung von $L(\mathcal{S})$ möglich wird.

Für den diskreten Fall kann wiederum von der in Glg. 5.12 skizzierten Grundidee ausgegangen werden, wobei der Merkmalsvektor $\vec{x}(k)$ zum Abtastzeitpunkt k von dem Vektorquantisierer in das diskrete Codewort $y_n(k)$ transformiert wird. Die n-te VQ-Partition bringt damit den Anteil $L_n(\mathcal{S})$ in die Gesamt-Likelihood der Menge $\mathcal{S}$ ein. Mit

$$L_n(\mathcal{S}) = \sum_{s\in\mathcal{S}} \sum_{k=1}^{K} \log p(y_n(k)|s)\gamma_s(y_n(k)) \tag{5.14}$$

resultiert für den diskreten Fall ein

$$L(\mathcal{S}) = \sum_{n=1}^{N_C} L_n(\mathcal{S}) = \sum_{s\in\mathcal{S}} \sum_{n=1}^{N_C} \sum_{k=1}^{K} \log p(y_n(k)|s)\gamma_s(y_n(k)). \tag{5.15}$$

Die Summe über k kann wiederum zusammengefasst werden, sodass die diskrete Ausgabeverteilung $b_s(y_n) = p(y_n|s)$, gewichtet mit der Zustandsfrequentierung $\gamma_s(y_n)$ des Zustands s durch Codewort y_n ein

$$L(\mathcal{S}) = \sum_{s\in\mathcal{S}} \sum_{n=1}^{N} \log b_s(y_n)\gamma_s(y_n) \tag{5.16}$$

ergibt. Hierbei ist zu betonen, dass es sich bei $L(\mathcal{S})$ um die Gesamt-Likelihood des Clusters *vor* dem Tying der einzelnen Zustände handelt. D. h. das Cluster ist zunächst nichts weiter als eine Gruppe einzelner Zustände mit separaten Parametersätzen. Durch ein Tying würden die einzelnen Verteilungen $b_s(y_n)$ durch eine repräsentative Verteilung $b_{\bar{s}}(y_n)$ ersetzt, die zu einer log-Likelihood

$$L(\bar{s}) = \gamma_{\bar{s}} \sum_{n=1}^{N_C} \log b_{\bar{s}}(y_n) \tag{5.17}$$

des Zustandes $\bar{s}$ führt.

Aus der Summation über alle N_C Codewörter in Glg. 5.16 resultiert des weiteren die totale Zustandsfrequentierung γ_s mit

$$\gamma_s = \sum_{n=1}^{N_C} \gamma_s(y_n). \tag{5.18}$$

Die Normierung der codewortbedingten Zustandsfrequentierung $\gamma_s(y_n)$ mit der totalen Zustandsfrequentierung γ_s liefert umgehend die diskrete Ausgabeverteilung $b_s(y_n)$ von Zustand s:

$$b_s(y_n) = \frac{\gamma_s(y_n)}{\gamma_s}. \tag{5.19}$$

In analoger Weise läßt sich die repräsentative Wahrscheinlichkeitsdichtefunktion $b_{\bar{s}}(y_n)$ berechnen

$$b_{\bar{s}}(y_n) = \frac{\sum_{s\in\mathcal{S}} \gamma_s(y_n)}{\sum_{s\in\mathcal{S}} \gamma_s}, \tag{5.20}$$

die sich praktischer Weise - unter Ausnutzung von Glg. 5.19 - als gewichtete Summe der Einzel-WDF bestimmen läßt:

$$b_{\bar{s}}(y_n) = \frac{\sum_{s\in\mathcal{S}} \gamma_s \cdot b_s(y_n)}{\sum_{s\in\mathcal{S}} \gamma_s}. \tag{5.21}$$

Umgestellt nach $\gamma_s(y_n)$ kann der Zusammenhang aus Glg. 5.19 in Glg. 5.16 verwendet werden, wodurch sich folgende interessante Berechnungsvorschrift für $L(\mathcal{S})$ mit

$$L(\mathcal{S}) = \sum_{s \in \mathcal{S}} \gamma_s \underbrace{\sum_{n=1}^{N_C} b_s(y_n) \log b_s(y_n)}_{-H_s}, \tag{5.22}$$

bzw. für $L(\bar{s})$ eines geclusterten Zustandes $\bar{s}$ zeigt:

$$L(\bar{s}) = \gamma_{\bar{s}} \underbrace{\sum_{n=1}^{N_C} b_{\bar{s}}(y_n) \log b_{\bar{s}}(y_n)}_{-H_{\bar{s}}}. \tag{5.23}$$

Die zweite Summe von Glg. 5.22 läßt sich eindeutig als negative Entropie H_s der diskreten Verteilung aus Zustand s identifizieren. Auch hier schließt sich also der Kreis: Der Likelihood-Verlust, der sich aus der Verschmelzung der Zustände $\mathcal{S}$ zu dem repräsentativen Zustand $\bar{s}$ ergibt, entspricht einem Informationsverlust

$$\Delta H = -\sum_{s \in \mathcal{S}} \gamma_s H_s + \gamma_{\bar{s}} H_{\bar{s}}. \tag{5.24}$$

Umgekehrt gilt natürlich, dass der Likelihood-Gewinn durch eine Aufspaltung von $\mathcal{S}$ in $\mathcal{S}^{(J)}$ und $\mathcal{S}^{(N)}$ mittels Frage q durch

$$\Delta L_q = \gamma_{\mathcal{S}^{(J)}} H_{\mathcal{S}^{(J)}} + \gamma_{\mathcal{S}^{(N)}} H_{\mathcal{S}^{(N)}} - \gamma_{\mathcal{S}} H_{\mathcal{S}} \tag{5.25}$$

ermittelt werden kann. Auch hier zeigt sich also, dass die für das Clustering erforderlichen Berechnungen allein auf der Grundlage der Ausgabeverteilungen der beteiligten Zustände und deren Frequentierungen durchgeführt werden können. Zusammenfassend kann die log-Likelihood $L(\bar{s})$ eines aus der Menge $\mathcal{S}$ hervorgegangenen Zustandes $\bar{s}$ mittels der Gleichungen 5.23 und 5.21 berechnet werden.

Um eine optimierte Modellstruktur zu finden, ist es sinnvoll eine Reihe optionaler Fragen für das Splitting eines Knotens anzubieten. Unter den möglichen Fragen q wird die für die nächste Aufspaltung anzuwendende Frage q^* ausgewählt, indem unter allen möglichen Fragen diejenige selektiert wird, die den Likelihood-Gewinn ΔL maximiert:

$$q^* = \arg\max_q \Delta L_q. \tag{5.26}$$

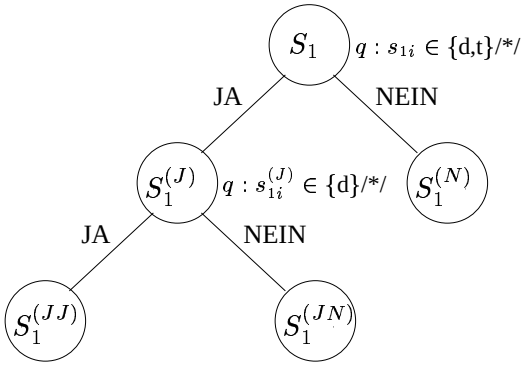
Die ausgewählte Frage kommt dann zur Anwendung, wenn der Likelihood-Gewinn einen gewissen minimalen Schwellwert überschreitet:

$$\Delta L_{q^*} \geq \hat{\Delta L}. \tag{5.27}$$

Ist diese Bedingung für eine optimale Frage nicht mehr erfüllt, wird das Splitting für den aktuellen Knoten abgebrochen und die dort verbliebenen Zustände werden verknüpft.

Ähnlich wie bei dem datengetriebenen Clustering kann auch bei dem Baum-basierten Clustering eine minimale Zustandsanzahl pro Cluster gefordert werden, um die Trainierbarkeit der generalisierten Modelle zu gewährleisten. Diese minimale Zustandsanzahl kann als zusätzliches Abbruchkriterium für das Clustering genutzt werden, indem für die potenziellen Unter-Cluster $\mathcal{S}^{(J)}$ und $\mathcal{S}^{(N)}$ die Prüfung auf minimale Zustandsanzahl in logischer UND-Verknüpfung positiv ausfallen muß. Andernfalls wird auch hierbei das Splitting abgebrochen.
Wurde nun ein Baum für eine Menge von Zuständen durch sukzessive Aufspaltung expandiert, kann die Struktur weiter verfeinert werden, indem für jede Paarung der entstandenen Terminalknoten geprüft wird, ob deren Verknüpfung mit einem vertretbaren Likelihood-Verlust erkauft werden könnte. Ist dies der Fall, erfolgt deren Zusammenfassung. Dazu wird wiederum der Schwellwert $\hat{\Delta L}$ verwendet, der schon als Abbruchkriterium bei der Aufspaltung verwendet wurde.

Nachdem die Frage nach der Bewertung einer Frage und deren Auswahl geklärt ist, ist die praktische Vorgehensweise bei dem Entscheidungsbaum-basierten Clustering noch näher zu beleuchten. Abb. 5.6 zeigt dazu den Vorgang an einem konkreten Beispiel. Auch bei dem Baum-basierten Clustering gilt im Regelfall, dass die Zustände positionsweise geclustert werden. Das Beispiel bezieht sich auf die Trigrapheme $d/e/r$, $d/e/s$, $i/e/s$ und $t/e/r$. Die Zustände aus den jeweiligen HMM mit der Position i werden in der Menge bzw. dem Knoten $\mathcal{S}_i$ gesammelt. Auf der Basis jedes Wurzelknotens wird anhand der Fragen $\mathcal{Q}$ ein Binärbaum gebildet. In Abb. 5.6 wird auf den Knoten $\mathcal{S}_1$ zuerst eine Frage q angewendet. In dem gezeigten Beispiel spaltet die Frage nach den rechten Kontextgraphemen $(q : s_{1i} \in \{d, t\}/*/)$ den Knoten $\mathcal{S}_1$ in eine Menge $\mathcal{S}_1^{(J)}$ und eine komplementäre Menge $\mathcal{S}_1^{(N)}$ auf. $\mathcal{S}_1^{(J)}$ enthält entsprechend der gewählten Frage alle Zustände, deren Grapheme die rechtsseitigen Kontextgrapheme $/d/$ oder $/t/$ aufweisen. Knoten $\mathcal{S}_1^{(N)}$ enthält alle übrigen Zustände aus Position 1 des Zentralgraphems $/e/$. In dem gezeigten Beispiel enthält $\mathcal{S}_1^{(J)}$ also die Zustände der Position 1 der Trigrapheme $d/e/r$, $d/e/s$ und $t/e/r$. $\mathcal{S}_1^{(N)}$ enthält Zustand 1 aus dem Trigraphem $i/e/s$.
Das Beispiel in Abb. 5.6 gibt zudem einen Einblick, wie sich welcher Kontext zusammen mit der Position der Zustände eines Knotens auf die Entwicklung des entsprechenden Baumes auswirkt.
Als erstes wurde für den Knoten $\mathcal{S}_1$, wie auch für alle übrigen Wurzelknoten, die Frage nach dem linken Kontext gestellt (ist linkes Kontextgraphem ein /d/ oder ein /t/?). Dies ist im übrigen bezeichnend: da im Allgemeinen auch bei anderen Zentralgraphemen zuerst Fragen nach dem linken Kontext gewählt werden (weil diese einen entsprechend großen Likelihood-Gewinn einbringen), kann davon ausgegangen werden, dass das vorausgehende Zeichen einen besonders starken Einfluß auf die Realisierung der Zentralgrapheme ausübt.

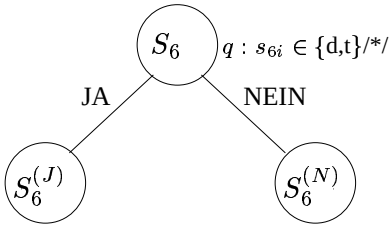


$S_1^{(J)}$ = {d,t}/e/* = {d/e/r, d/e/s, t/e/r}

$S_1^{(N)}$ = {i/e/s}

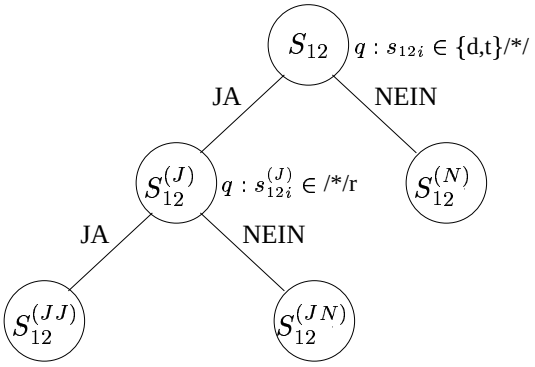
$S_1^{(JJ)}$ = {d/e/r, d/e/s}

$S_1^{(JN)}$ = {t/e/r}



$S_6^{(J)}$ = {d,t}/e/* = {d/e/r, d/e/s, t/e/r}

$S_6^{(N)}$ = i/e/s



$S_{12}^{(J)}$ = {d,t}/e/*

$S_{12}^{(N)}$ = i/e/s

$S_{12}^{(JJ)}$ = {d,t}/e/r = {d/e/r, t/e/r}

$S_{12}^{(JN)}$ = {d/e/s}

Abbildung 5.6: Prinzip des Entscheidungsbaum basierten State-Clusterings

Des weiteren ist zu beobachten, dass die Relevanz kontextbedingter Entscheidungen zur Aufspaltung von Clustern zur Modellmitte (6. Zustand) hin zunächst abnimmt, um zum Modellende (12. Zustand) hin wieder anzusteigen. Dies wird durch die abflachende Tiefe der Bäume zur Modellmitte angezeigt. Eine allzu große Diversifizierung der Zustände ist dort scheinbar nicht sinnvoll. Dieser Effekt ist mit den Kontext-erfassenden Eigenschaften der Merkmalsextraktion zu begründen, da diese Eigenschaft gerade am Modellanfang und -ende zu einer höheren Inkonsistenz der Merkmale führen muß.

Darüber hinaus nimmt der oben erwähnte Einfluß der vorausgehenden Kontext-Grapheme

zum Modellende hin ab. Dies läßt sich daran ablesen, dass mit fortschreitender Position der Zustände in den Modellen zunehmend Fragen ausgewählt werden, die den rechten Kontext betreffen. In Abb. 5.6 ist dies exemplarisch an der Aufspaltung des Knotens $\mathcal{S}_{12}^{(J)}$ zu sehen.

5.4.1 Einbeziehung typographischer Eigenschaften

Bei sämtlichen Betrachtungen in dem vorangegangenen Abschnitt wurde davon ausgegangen, dass die für das Clustering notwendigen Fragen bekannt sind. Da diese Fragen und die damit verbundenen Graphemgruppen jedoch möglichst unterschiedliche Kontexteinflüsse berücksichtigen sollen, ist deren Formulierung keineswegs trivial. Während aus der Linguistik solche Kontexteinflüsse für phonetische Realisierungen der gesprochenen Sprache bekannt sind und dieses Wissen auch für den Aufbau kontextabhängiger Phonem-Modelle eingesetzt wird, gibt es in der Literatur bisher wenig Hinweise über Kontexteinflüsse von Sub-Worteinheiten auf handgeschriebene Sprache.
Einen wesentlichen Einfluß haben sicherlich Großbuchstaben auf die Ausführung nachfolgender Zeichen. Die im Abschnitt B.2 gezeigten Beispiele lassen einen solchen Zusammenhang vermuten, da gerade Großbuchstaben - auch bei einer ansonsten eher kursiven Schreibweise - tendenziell eher in Druckschrift ausgeführt werden. Zwei mögliche Graphemklassen aus denen sich Fragen nach dem Kontext ableiten lassen sind demnach Groß- und Kleinbuchstaben.
Eine weitere Unterteilung bietet sich bei den Kleinbuchstaben an. Hier ist es z. B. denkbar, dass die Zeichenhöhe und die vertikale Position - ähnlich wie bei den Großbuchstaben - zu Graphemvariationen führt. Dementsprechend ließen sich die Klassen der *in-line*-Zeichen (Zeichen ohne Ober- oder Unterlängen), der Oberlängen-behafteten Zeichen, und der Buchstaben mit Unterlängen definieren.
Im Sinne maximaler Benutzbarkeit wurde schon bei der Datenakquisition auf die Definition bestimmter Regeln für die Schriftgenerierung verzichtet. Bedingt dadurch ist es in den Datenbasen häufig zu beobachten, dass zuerst eine Sequenz von Buchstabenkörpern geschrieben wird, und erst einige Zeichen später sog. diakritische Zeichen gesetzt werden. Dazu gehören die Punkte über einem ’i’ oder einem ’j’, sowie Punkte über Umlauten und t-Striche. Ein Einfluß diakritischer Zeichen auf Umgebungsbuchstaben ist daher ebenfalls naheliegend.
Diese Überlegungen führen zusammengefasst zu den folgenden Graphemklassen, aus denen sich kontextbezogene Fragen ableiten lassen:

Klasse 1: A, B, C, . . . Z, Ä, Ö, Ü

Klasse 2: a, c, e, m, n, o, r, s, u, v, w, x, z

Klasse 3: b, d, h, k, l, ß

Klasse 4: f, g, p, q, y

Klasse 5: i, j, t, ä, ö, ü

Beispielsweise kann eine aus diesen Klassen abgeleitete Frage demzufolge lauten: 'Ist der rechte Kontext ein Graphem der Klasse 1'?
Im Vorgriff auf die später dargestellten Ergebnisse ist anzumerken, dass dieser Ansatz eine eher mäßige Erkennungsgenauigkeit liefert. Von diesem Standpunkt aus wäre darum vergleichsweise eher der datengetriebene Ansatz zu favorisieren. Jedoch kann andererseits auch nicht garantiert werden, dass mit den oben dargestellten Kontextklassen eine optimale Grundlage für das Entscheidungsbaum-basierte Clustering gefunden wurde. Vielmehr ist anzunehmen, dass diese relativ einfache Unterteilung die Kontexteinflüsse noch nicht in vollem Maße berücksichtigt.
Im folgenden sollen daher verstärkt selbstorganisierende Verfahren entwickelt werden, die es ermöglichen, auf Basis der vorhandenen Daten das fehlende graphonomische Wissen zu generieren. Diese Verfahren werden in den nachfolgenden Abschnitten näher beschrieben.

5.4.2 Clustering Markov'scher Quellen

Grundlegende Idee dieses Ansatzes ist es, gewisse Ähnlichkeiten unter Graphemen zu finden, die anhand des Schriftbildes nicht ohne weiteres ersichtlich sind, sich aber in der gewählten Merkmalsrepräsentation wiederfinden. Naheliegend wäre es also, die Merkmalssequenzen verschiedener Grapheme in der Trainingsdatenbasis zu vergleichen. Ein direkter Vergleich der Merkmalssequenzen brächte jedoch die folgenden zwei wesentlichen Probleme mit sich:

- Zum einen ist zu beachten, dass es relativ schwierig ist, einzelne Merkmalssequenzen, die zum Vergleich herangezogen werden könnten, als repräsentativ zu bezeichnen. Die stark variierenden Realisierungsformen verhindern dies.

- Zum anderen erschweren aber auch die äußerst uneinheitlichen Merkmalssequenz-Längen einen direkten Vergleich durch herkömmliche Abstandsmaße.

Eine interessante Alternative bietet sich allerdings mit einem indirekten Vergleich der Merkmalssequenzen über die Monograph-HMM, da diese - in statistisch abstrahierter Form - ein gemitteltes aber dennoch relativ detailiertes Abbild der Trainingsdaten darstellen. Hierzu werden die HMM zu Markov'schen Quellen umfunktioniert [Kos98d]. Abb. 5.7 zeigt dazu eine Möglichkeit wie sich HMM als Markov'sche Quellen nutzen lassen. Werden die Tran-

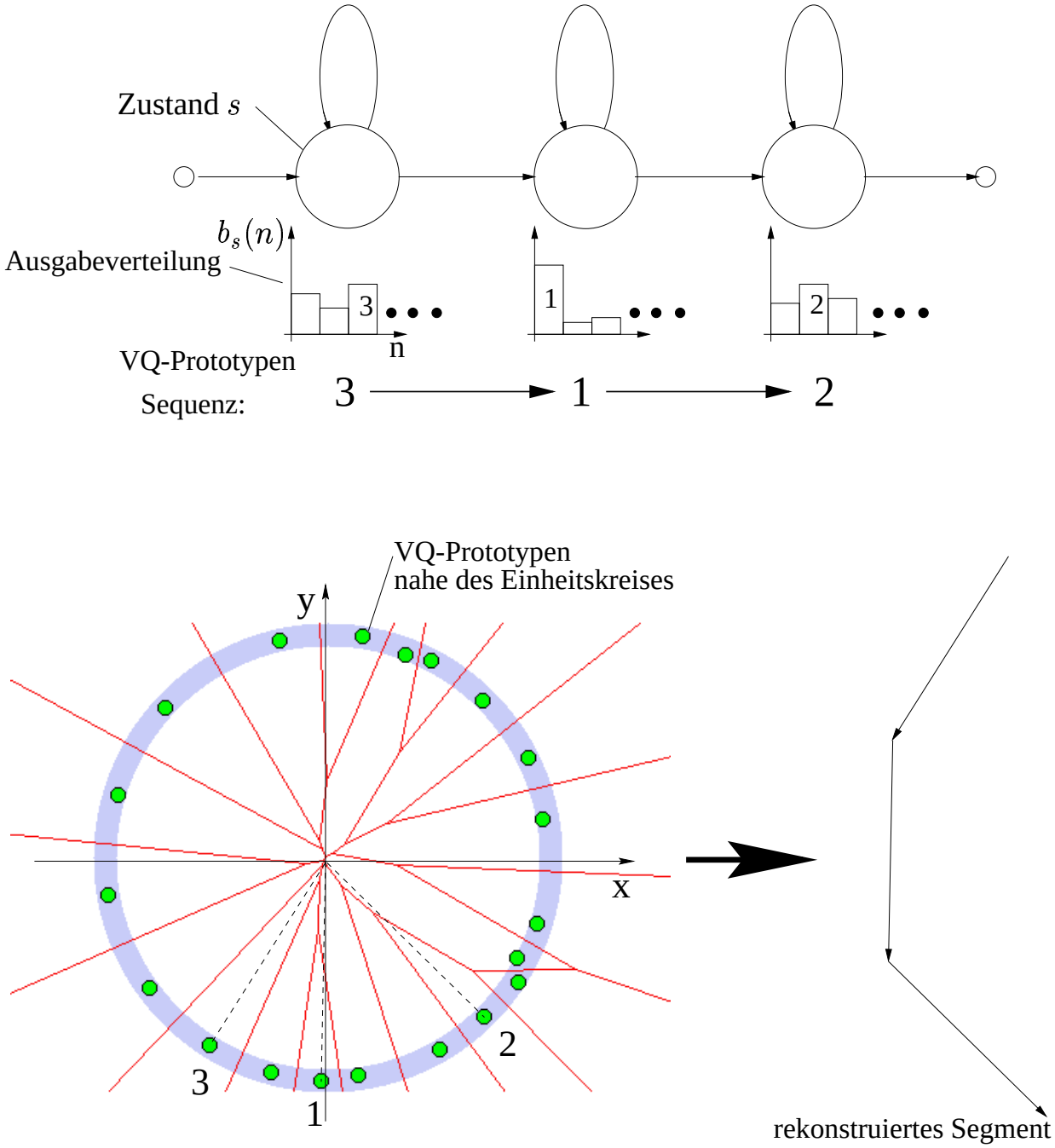


Abbildung 5.7: Clustering Markov'scher Quellen

sitionswahrscheinlichkeiten der HMM und die damit verbundenen Verweilzeiten in den einzelnen Zuständen vernachlässigt, lassen sich für jedes Graphem typische, längennormierte Trajektorien ermitteln, die anschließend problemlos einem selbstorganisierenden Minimum-Abstands-Klassifikator zugeführt werden können.

Zur Rekonstruktion der Stifttrajektorie anhand der als Quelle genutzten HMM bietet sich insbesondere die als Merkmal genutzte Orientierung der Tangente der Stifttrajektorie (α-Merkmalsstrom) an. Dies ist dadurch zu erklären, dass die Stifttrajektorie zunächst räumlich unterabgetastet wird. Die Abtastvektoren interpolieren dabei quasi-äquidistante Abtastpunk-

te. Im Umkehrschluß kann also die Trajektorie lediglich anhand der Winkelsequenz der Abtastvektoren angenähert werden, da ja der Abstand aufeinander folgender Abtastpunkte als Konstante bekannt ist. Da jede dieser Orientierungen wiederum in je einen $\cos$- und $\sin$-Wert kodiert wird (um die Unstetigkeit bei 2π zu beheben), sind die Merkmale demzufolge auf dem Einheitskreis verteilt. Ein weiterer Vorteil der Nutzung der α-Merkmale ist, dass die nur zweidimensionalen Eingangsvektoren durch ein Codebuch der Größe 30 quantisiert werden. Der entstehende nachteilige Quantisierungsfehler hält sich damit in Grenzen, was sich letztendlich positiv auf das Rekonstruktionsergebnis auswirkt.
Um die Rekonstruktion der Trajektorie anhand der Abb. 5.7 zu verdeutlichen, sei zunächst angenommen, dass die HMM aus drei Zuständen bestehen. Anhand der Maximumsuche

$$n_s^* = \arg\max_n b_s(n) \tag{5.28}$$

wird das Codewort n_s^* mit der höchsten Wahrscheinlichkeit aus der Ausgabeverteilung $b_s(n)$ für jeden Zustand s innerhalb eines HMM gesucht. Aus den oben dargelegten Gründen wird diese Suche ausschließlich auf den Ausgabeverteilungen der α-Merkmale durchgeführt. Die Wahrscheinlichkeit für ein Codewort n ist natürlich ein direktes Maß für die Häufigkeit des korrespondierenden n-ten Codebuchvektors (im betreffenden Zustand). Darauf basierend läßt sich nun eine Sequenz von VQ-Prototypen für jedes HMM bestimmen. Diese synthetisierte VQ-Sequenz läßt sich in Verbindung mit dem Codebuch des α-Merkmals nutzen um die Stifttrajektorie des modellierten Graphems zu rekonstruieren. Die Codewörter stellen die (numerierten) Cluster des partitionierten Merkmalsraumes dar. Im Falle des α-Merkmals besteht jeder Codebuchvektor aus zwei Komponenten, die den $\sin$- und $\cos$-Werten der Abtastvektororientierungen in dem Einheitskreis entsprechen. Diese Codebuchvektor-Komponenten, multipliziert mit der konstanten Abtastvektorlänge ergeben die Beiträge in x- und y-Richtung zu dem interpolierten Trajektorienstück. Die Trajektorienstücke lassen sich dann zustandsweise zu einem synthetischen Graphem zusammenfügen indem die Beiträge in x- und y-Richtung inkrementell an das Ende des jeweils vorherigen Vektors angefügt werden.
Abb. 5.8 zeigt einige der auf diese Weise rekonstruierten Grapheme. Jede dieser rekonstruierten Grapheme repräsentiert eine charakteristische Kurvenform. Die uniforme Länge erlaubt es schließlich Standard-Clustering Methoden, wie z. B. den k-means Algorithmus anzuwenden, um durch unüberwachtes Lernen Graphemklassen zu identifizieren.
Das hier zugrundeliegende Prinzip zur Bestimmung von Kontextklassen beruht auf der Annahme, dass Nachbargrapheme, die auf Grund bestimmter gemeinsamer Eigenschaften als ähnlich gelten, auch ähnliche Auswirkungen auf ein Zentralgraphem zeigen. Ist dies der Fall, lassen sich aus den ermittelten Graphemklassen natürlich Fragen für das Baum-basierte Clustering ableiten, indem danach gefragt wird, ob der Kontext zu einer bestimmten Graphemgruppe gehört.

Die folgende Tabelle zeigt das Ergebnis des Clusterings.

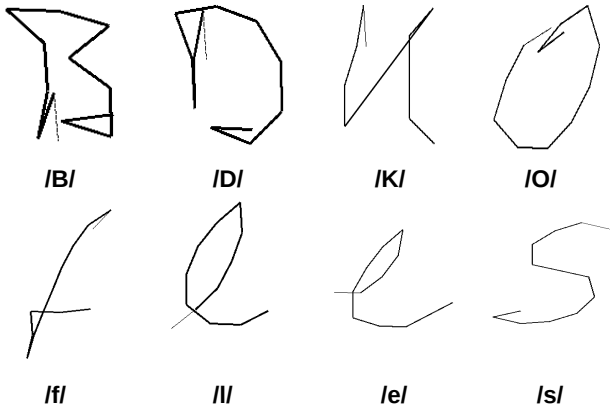


Abbildung 5.8: Beispiele rekonstruierter Grahpeme

Klasse 1: A, D, J, M, Y

Klasse 2: B, C, F, G, H, I, K, L, P, R, S, U, Z

Klasse 3: E, N, O, Q, T, V, W, X, Ä, Ö, Ü

Klasse 4: a, d, j, k, n, o, p, q, r, s, t, v, w, x, ö, ü, ß

Klasse 5: b, c, f, h, i, l, u, ä

Klasse 6: e, g, m, y, z

Hierbei wurden die repräsentativen Sequenzen mittels des k-means Algorithmus zu 2x3 Gruppen eingeordnet, wobei Groß- und Kleinbuchstaben jeweils getrennt gruppiert wurden. Verglichen mit dem in Abschnitt 5.4.1 vorgestellten heuristischen Ansatz stehen durch den selbstorganisierenden Ansatz insgesamt sechs anstatt der fünf zuvor definierten Klassen zur Verfügung. Auch die Gruppenaufteilung scheint eine komplett andere Struktur hervorgebracht zu haben als bei der heuristischen Aufteilung. Ein Teil der Ergebnisse des Clusterings scheint aus graphonomischer Sicht einleuchtend zu sein. Dies ist z. B. der Fall bei den Graphemen /C/ und /G/ der Klasse 2, deren Trajektorien über weite Strecken durchaus Ähnlichkeiten zugeschrieben werden könnten. Gleiches ließe sich beispielsweise auch für das /O/ und /Q/ als eine Teilklasse, sowie für das /N/, das /V/ und das /W/ als eine weitere Teilgruppe aus Klasse 3 postulieren.
Selbst unter der Berücksichtigung der Bildung von Teilgruppen scheinen andere Klassenzuordnungen wiederum weniger offensichtlich. Jedoch ist hier nochmals zu unterstreichen, dass die Gruppierungen auf der Basis der beobachteten Merkmale durchgeführt wurden, und dies nicht zwangsweise mit dem Schriftbild gleichzusetzen ist, das sich dem menschlichen Betrachter offenbart.

Zusammenfassend läßt sich feststellen, dass die Ähnlichkeiten aus einem Teil des Merkmalsraumes abgeleitet werden konnten, der auch bei der allgemeinen Modellbildung kontextabhängiger wie kontextunabhängiger HMM relevant ist und darum gegenüber einem heuristischen Ansatz gewisse Vorteile bieten sollte.

5.4.3 Tree-Sweeping

Ein weiterer untersuchter Ansatz zur Generierung von Kontextklassen ist ein zweistufiger, ebenfalls selbstorganisierender Prozeß, der wiederum auf dem Entscheidungsbaum-basierten Clustering beruht. In einem ersten Clustering-Durchlauf werden auch hier die Graphemklassen zusammengefasst, die in einer zweiten Stufe dann als Grundlage für das eigentliche Clustering der HMM-Zustände verwendet werden (Abb. 5.9). In der ersten Stufe wird zunächst ein Baum-basiertes Clustering auf die vorhandenen Monographeme angewendet. In der zweiten Stufe werden anhand der generierten Fragen die Zustände der expandierten n-Grapheme verknüpft.
Wesentlicher Unterschied zu dem im vorherigen Abschnitt beschriebenen Verfahren der Markov'schen Quellen ist, dass die Zusammenfassung der Monographeme zu Kontextklassen entsprechend einer Maximum-Likelihood Optimierung durchgeführt wird. Damit ist es möglich über den Vergleich der Kurveneigenschaften hinaus zu gehen und die Betrachtungen auf sämtliche zur Verfügung stehenden Merkmale auszuweiten.
Die Ausgangssituation in der ersten Stufe ist, dass zunächst alle Zustände der vorhandenen Monographeme in einem Wurzelknoten akkumuliert werden. Im Unterschied zu dem allgemeinen Baum-basierten Verfahren werden die Entscheidungsbäume nicht getrennt zustandsweise expandiert. Ausgehend von dem Wurzelknoten wird das Splitting so lange vorangetrieben, wie ein bestimmter Wert für $\hat{\Delta L}$ überschritten wird. Die Aufspaltung wird dabei anhand primitiver Fragen vorgenommen, welche die Zustände der enthaltenen Monograph-

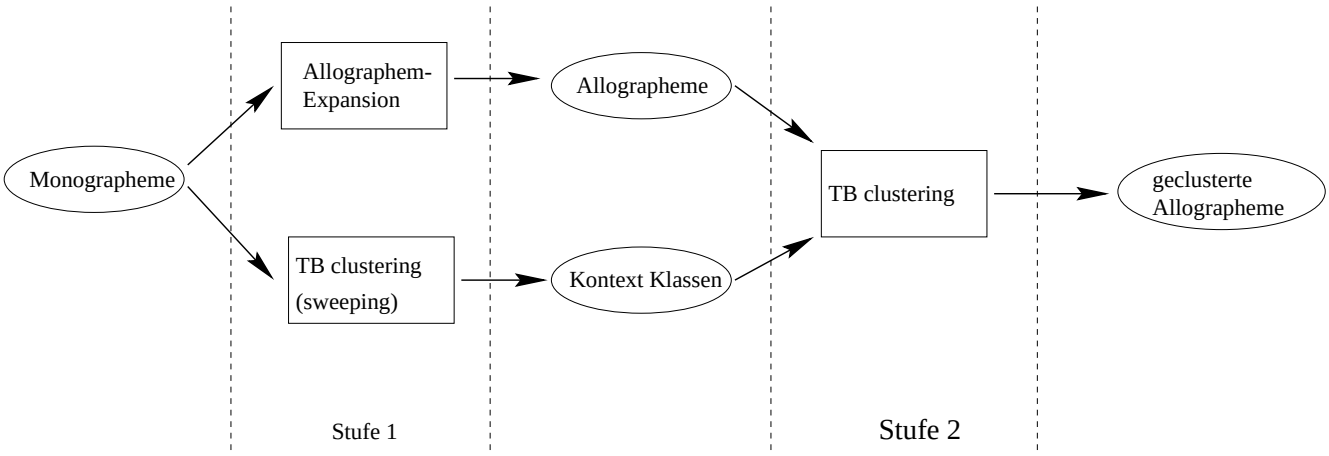


Abbildung 5.9: Übersicht zum Entscheidungsbaum basierten Clustering

eme betreffen. Z. B. ist eine solche potenzielle Frage, ob ein Zustand zum Monographem /E/ gehört ($q : s_{ij} \in /E/$). Betrachtungen des Kontextes bleiben in dieser Stufe zunächst außen vor. Die Auswahlmechanismen der Fragen sind identisch mit denen des allgemeinen Baum-basierten Clusterings.

Die sukzessive Anwendung solcher Fragen bringt die im oberen Teil in Abb. 5.10 gezeigte Baum-Struktur hervor, in der an jedem Knoten die Zustände des nachgefragten Monographems abgespalten werden (wenn ein entsprechender Likelihood-Gewinn gegeben ist). Diese aus den Zuständen eines Monographems bestehenden Ausgangsknoten seien als primitive Knoten bezeichnet. Nach vollständiger Aufspaltung, d. h., wenn der durch $\hat{\Delta L}$ spezifizier-

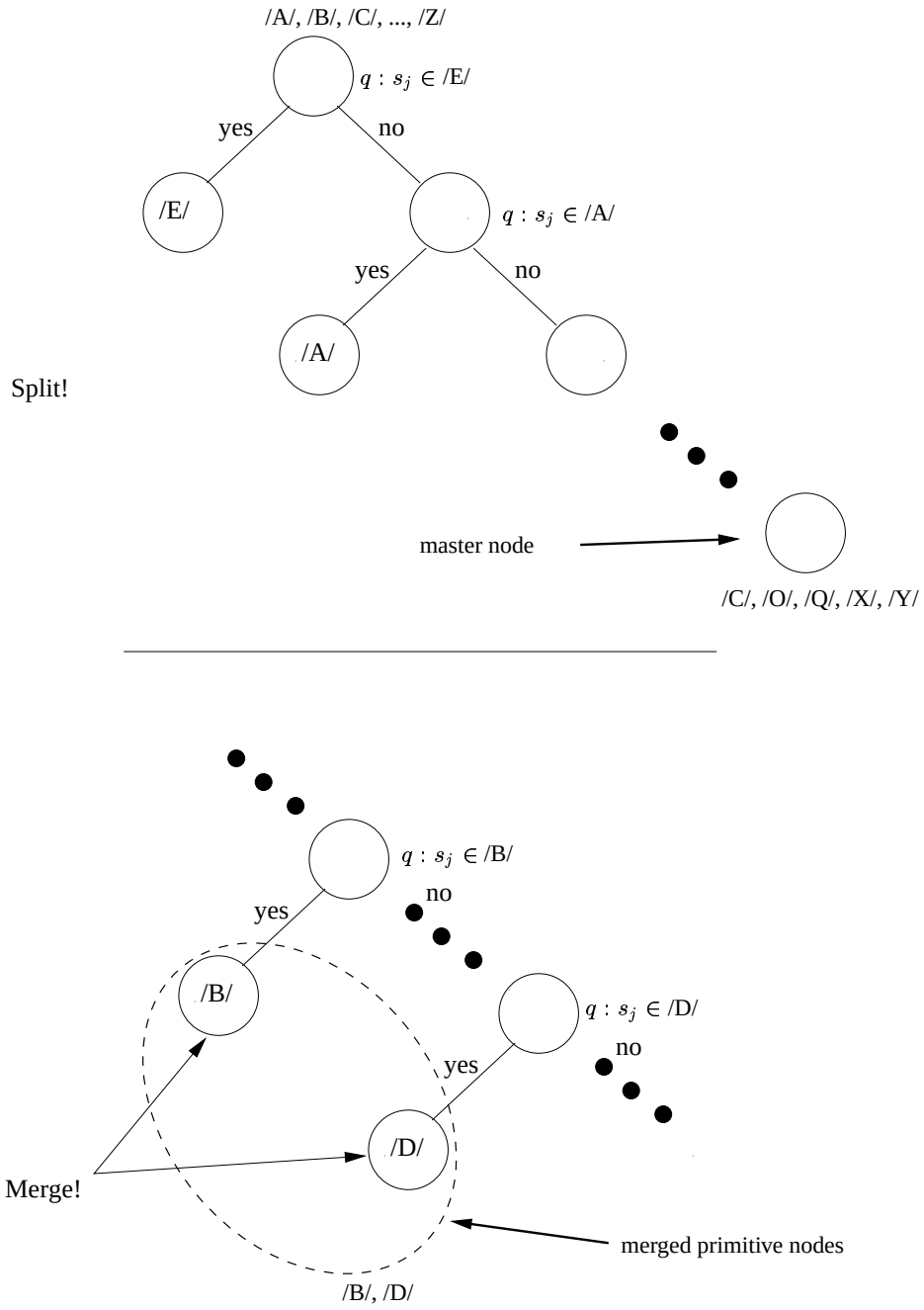


Abbildung 5.10: Split-and-Merge von Zustands-Clustern

te minimal geforderte Likelihood-Gewinn durch weitere Abspaltungen nicht mehr erreicht werden kann, verbleibt ein nicht-primitiver Ausgangsknoten auf unterster Baum-Ebene. Dies sei der sog. Master-Knoten.
Um zusätzliche nicht-primitive Cluster mit einem Baum zu generieren, wird wie im unteren Teil in Abb. 5.10 gezeigt, in einem *Merging*-Durchlauf versucht die primitiven Cluster weiter zusammenzufassen. Für jede mögliche Cluster-Paarung wird geprüft, ob deren Verknüpfung einen Likelihood-Verlust unterhalb der spezifizierten Schranke $\hat{\Delta L}$ mit sich bringt. Ist dies der Fall, erfolgt die Verknüpfung der beteiligten Cluster. Die Verknüpfung der Zustände in den nicht-primitiven Knoten des entstandenen Baumes zu einem allgemeinen, Graphemklassen-übergreifenden HMM würde auf diese Weise zu einem vergleichsweise geringen Likelihood-Verlust führen.
Die ausgebildete Struktur des Baumes - insbesondere dessen Tiefe - hängt unter anderem entscheidend von dem gewählten Parameter $\hat{\Delta L}$ ab. Durch das sog. *Sweeping* des $\hat{\Delta L}$-Parameters innerhalb eines bestimmten Bereiches ist es möglich eine Vielzahl verschiedener Baumstrukturen, bzw. Graphemkombinationen zu erzeugen. Aus allen erzeugten Baum-Strukturen zwischen den Extremen der völligen Verknüpfung aller Modelle und der totalen Aufspaltung in jeweils einzelne HMM werden alle auftretenden nicht-primitiven Cluster gespeichert. Diese Sammlung nicht-primitiver Graphemklassen läßt sich anschließend wiederum als Grundlage für Kontext-bezogene Fragen verwenden.
In dieser Weise wird für Groß- und Kleinbuchstaben getrennt verfahren. Ein Teil der so erzeugten Graphemklassen ist in Abb. 5.11 gezeigt. Bei den dargestellten Graphemgruppen handelt es sich um die entstandenen Master-Knoten. Die Pfeile in der Abbildung deuten die Abhängigkeit der Zusammensetzung eines Master-Knotens von dem Parameter $\hat{\Delta L}$ an. Abb. 5.11 verdeutlicht darüber hinaus, dass Master-Knoten, die unter einem kleineren $\hat{\Delta L}$ entstanden sind, stets eine Teilmenge größerer Master-Knoten sind.

5.5 Ergebnisse

Zur Evaluierung kontextbezogener Modellierungstechniken sind umfangreiche Datenbasen erforderlich. Es ist zu bedenken, dass die kontextbedingten Effekte in möglichst vielfältigen Kombinationen mehrfach bei einem Schreiber zu beobachten sein sollten. Da bei der schreiberabhängigen Datenbasis die Wortanzahl pro Schreiber im Trainings-Set um den Faktor 10 größer ist, als bei der schreiberunabhängigen Datenbasis, werden die Experimente zunächst an dem schreiberabhängigen System durchgeführt. Von jedem der drei Schreiber wurden zu Trainingszwecken ca. 2000 Wörter gesammelt. Es besteht damit die Hoffnung, dass beobachtbare Kontexteinflüsse in gewissem Maße als repräsentativ gelten.
Tab. 5.2 zeigt die erzielten Ergebnisse. Zu Vergleichszwecken ist in Tab. 5.2 in Zeile 1

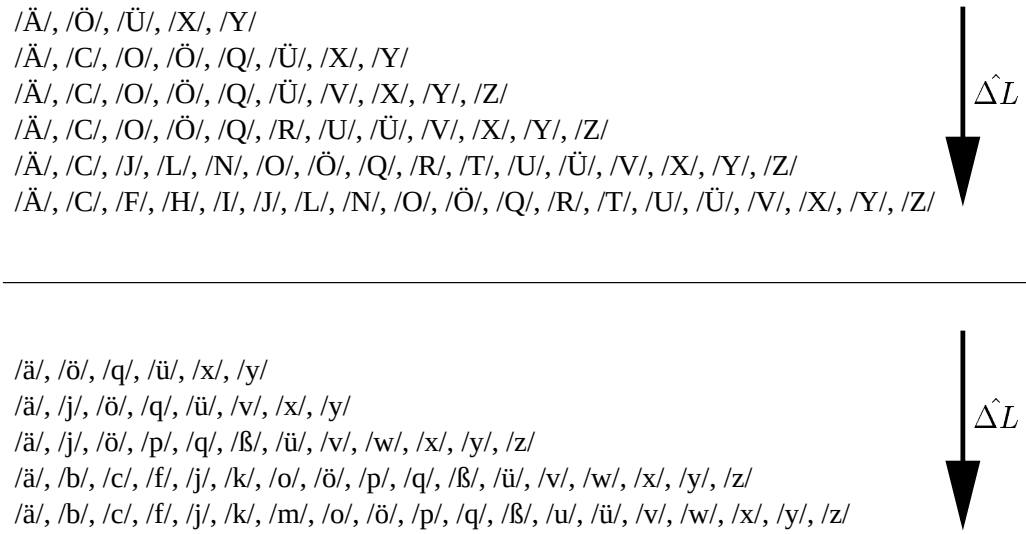


Abbildung 5.11: Entwicklung der Master-Knoten durch Tree-Sweeping

das Ergebnis des diskreten Basissystems, mit kontextunabhängigen Monographemen aufgeführt. Die Spalten 2 bis 4 geben die Erkennungsraten für die einzelnen Schreiber wieder. Die 5. Spalte zeigt die durchschnittliche Erkennungsrate mit der relativen Fehlerreduktion in Klammern, bezogen auf das Monographemsystem in Zeile 1. Spalte 6 gibt Auskunft über die durchschnittliche Anzahl von Modellen *nach* dem Clustering. Die Modellanzahl in Spalte 6 ist mit den entsprechenden Werten aus Tab. 5.1 zu vergleichen, in der die Modellanzahl in Abhängigkeit des verwendeten Lexikons und des berücksichtigten Kontextes aufgelistet ist. Die Ergebnisse in Tab. 5.2 sind mit dem 30k-Lexikon erzielt worden. Die Zeilen 2-6 zeigen die Ergebnisse unter Verwendung von Trigraphemen. Die Anzahl der Trigrapheme vor dem Clustering beträgt demnach 8900. Die 2. Zeile in Tab. 5.2 gibt die erzielten Ergebnisse für die selektiv hinzugefügten Trigrapheme wieder (Abschnitt 5.2.1). Die Ergebnisse mit kontextabhängigen Modellen geben die maximalen Erkennungsraten, die mit optimaler Einstellung der jeweiligen Parameter erzielt werden konnten, wieder. Mit dem datengetriebenen Clustering der Trigraphem-Zustände (Abschnitt 5.3) konnte die durchschnittliche Er-

	ank	jmr	vdm	ø	Anzahl HMMs
Monographeme	93.0 %	77.9 %	83.4 %	84.8 %	80
selektives Verfahren	93.6 %	79.1 %	89.8 %	87.5 (17.8) %	138
datengetrieben	96.2 %	82.9 %	94.7 %	91.3 (42.8) %	96
Baum, heuristisch	95.7 %	90.9 %	93.6 %	93.4 (56.6) %	416
Baum (WI), Markov-Quelle	96.2 %	92.0 %	93.6 %	93.9 (59.9) %	403
Baum (WD), Markov-Quelle	96.2 %	91.4 %	94.6 %	**94.1** (61.1) %	434

Tabelle 5.2: Erkennungsergebnisse - Vergleich verschiedener Reduktionsansätze, schreiberabhängig, 30K Lexikon

kennungsrate auf 91,3 % verbessert werden. Die drei letzten Zeilen in Tab. 5.2 zeigen die Ergebnisse mit Entscheidungsbaum-basiertem Clustering. An der Anzahl verbleibender Modelle ist bei dem Entscheidungsbaum-basierten Clustering ein interessanter Effekt abzulesen. Die größere Anzahl läßt darauf schließen, dass die kontextabhängigen Modelle deutlich detailierter ausgeführt werden, als bei den übrigen Verfahren, was schließlich zu einer vergleichsweise hohen Erkennungsrate führt.
Zunächst ist in Zeile 4 das Ergebnis wiedergegeben, welches mit heuristisch angesetzten Fragen (Abschnitt 5.4.1) zu erreichen war. Die Ergebnisse in Zeilen 5 und 6 beziehen sich auf die Generierung von Fragen durch Markov'sche Quellen (Abschnitt 5.4.2). Interessant sind die geringen Differenzen zwischen der letzten und der vorletzten Zeile. Für die Ergebnisse der letzten Zeile wurden für jeden Schreiber individuelle Fragen, bzw. Bäume für das Clustering generiert. Die Generierung der Fragen wurde dabei mit den schreiberabhängigen Monographemen durchgeführt. Die vorletzte Zeile beschreibt die Ergebnisse, die mit einer allgemeinen Menge von Fragen erzielt wurden. Dieser allgemeine Satz von Fragen wurde wiederum aus einem schreiberunabhängigen System generiert und einheitlich für jeden Schreiber verwendet. Dieser Vergleich legt den Schluß nahe, dass es selbst für schreiberabhängige Systeme möglich ist, eine allgemeingültige Menge von Fragen zu entwickeln, ohne dafür nennenswerte Verluste an Genauigkeit hinnehmen zu müssen.
Die selbstorganisierende Generierung von Fragen zeigt gegenüber den übrigen Verfahren die höchste Korrektheit. Es zeigt sich darüber hinaus, dass über die zur Verfügung gestellten Fragen ein gewisses Optimierungspotenzial gegeben ist. Um schließlich dass optimale Verfahren zu finden, soll ein abschließender Vergleich der Markov'schen Quellen und des Tree-Sweepings anhand des 200k-Lexikons durchgeführt werden. Wie Tab. 5.1 zeigt, stellt eine solche Lexikongröße aufgrund der sehr stark steigenden Modellanzahl eine beträchtliche Herausforderung an kontextorientierte Modellierungsformen.
Bei einer derartigen Größe des Lexikons stellt sich zudem die Frage, ob die generelle Verwendung von Trigraphemen in jedem Fall die beste Lösung darstellt. Die Ergebnisse in Tab.

		ank	jmr	vdm	ø	Anzahl HMMs
Bigrapheme links	Markov'sche Quelle	94.62	84.49	89.84	89.65	284
	Tree-Sweeping	95.70	83.42	89.84	89.65	351
Bigrapheme rechts	Markov'sche Quelle	93.55	83.42	89.84	88.93	341
	Tree-Sweeping	93.55	82.89	88.77	88.40	292
Trigrapheme	Markov'sche Quelle	95.16	84.49	89.84	89.83	603
	Tree-Sweeping	94.62	86.63	90.37	90.54	633

Tabelle 5.3: Vergleich verschiedener Kontextbereiche, schreiberabhängig, 200k Lexikon

5.3 sollen diese Fragen beantworten. Die erste Spalte in Tab. 5.3 gibt Auskunft über den berücksichtigten Kontext. Die zweite Spalte bezeichnet das Verfahren zur Generierung von Fragen. Sowohl mit den Markov'schen Quellen, wie auch mit dem Tree-Sweeping wurde je ein generell angewendeter Satz von Fragen für alle drei Schreiber aus dem schreiberunabhängigen System generiert. Bei einem Vergleich zwischen links-abhängigen Bigraphemen und rechts-abhängigen, zeigt sich eine höhere Korrektheit bei der Berücksichtigung des linken Nachbargraphems. Dies ist in sofern einsehbar, da der aktuell geschriebene Buchstabe offensichtlich stärker von dem vorausgehenden Nachbarn beeinflußt wird, als von seinem Nachfolger. Bei der Verwendung links-abhängiger Bigrapheme läßt sich zunächst kein Einfluß auf die verwendeten Fragen feststellen. Markov'sche Quellen zeigen hingegen einen Vorteil bei rechts-abhängigen Bigraphemen. Die höchsten Erkennungsraten liefert schließlich die Verwendung von Trigraphemen, wobei sich hier insbesondere die aus dem Tree-Sweeping hervorgegangenen Fragen als geeignet erweisen. Die höhere Erkennungsrate bei Trigraphemen läßt sich wiederum damit begründen, dass Trigrapheme die Vorteile von rechts- und links-abhängigen Bigraphemen vereinen. Wenn auch links-Bigrapheme dem dominierenden Einfluß vorausgehender Buchstaben gerecht werden, können diese weniger gut die Abhängigkeiten bei Großbuchstaben erfassen, da diese in der Regel lediglich einen rechten Kontext aufweisen.
Mit dem hier beschriebenen Verfahren des Tree-Sweepings läßt sich bei optimiertem $\hat{\Delta L}$ die Erkennungsrate des schreiber*un*abhängigen Systems mit einem Wortschatz von 2000 Wörtern von 90,6 % auf 93,2 % verbessern.

5.6 Kapitelzusammenfassung

Mit der Einführung kontextabhängiger Modelle eröffnet sich die Möglichkeit die Vielfältigkeit der Handschrift auf komplexere Modellstrukturen abzubilden, als es mit kontextunabhängigen Modellen möglich wäre. Durch entsprechende Verknüpfung von Modellparametern wird auf selbstorganisierende Weise eine optimierte individuelle Modellstruktur gefunden. Mit der Parameterverknüpfung wird gleichzeitig die Parameteranzahl auf ein sinnvolles Maß reduziert, sodass die Trainierbarkeit der Modelle - auch bei einer begrenzten Anzahl von Trainingsbeispielen - gegeben ist.
Insbesondere bei der Verwendung sehr großer Lexika spielt die Parameterverknüpfung eine entscheidende Rolle. Die meisten potenziell möglichen Graphemkombinationen sind bei sehr großem Vokabular nicht in der Trainingsmenge vorhanden. Das Entscheidungsbaumbasierte Clustering bietet dabei gegenüber dem datengetriebenen Clustering besondere Vorteile, da es durch die entstehende Baumstruktur auch 'ungesehene' Graphemkombinationen synthetisiert.
Für die Entwicklung der Baumstruktur ist wiederum die Bildung von Graphemklassen erfor-

derlich, die unterschiedliche Kontexteinflüsse widerspiegeln. Diese Graphemklassen wurden anhand verschiedener Verfahren generiert. Ein abschließender Vergleich der Verfahren hat trotz der relativ kleinen Datenbasen die Überlegenheit kontextueller Modellierung gezeigt.