Untersuchung von Methoden zur flexiblen Schlüsselworterkennung bei Telefonsprache

Diplomarbeit

von

Lutz Vieweg

Institut für Angewandte Physik

der Johann Wolfgang Goethe-Universität

Frankfurt am Main

November 1996

Inhaltsverzeichnis

1. Einführung 3

2. Signalaufbereitung und Merkmalsextraktion 5

3. Struktur des Sprachmodells 8

4. Bewertung und Klassifikation der Merkmalsfolge 10

5. Phonembasierte Schlüsselworterkennung 12

5.1. Phoneme als Füllmodelle 14

5.2. Grammatikfreie Schlüsselworterkennung 16

6. Simulationsprogramm zur Schlüsselworterkennung 22

6.1. Übersicht der Programmstruktur 22

6.2. Optimierung des Rechenaufwands und der Rechenzeit 25

7. Messungen und Ergebnisse 31

7.1. Verwendete Sprachdaten 31

7.2. Sprachmodelle 32

7.3. Experimente zur 1-Schlüsselwort-Erkennung 34

7.4. Experimente zur n-Schlüsselwort-Erkennung 40

8. Zusammenfassung 46

9. Literaturverzeichnis 49

A. Beschreibung der Spracherkennungprogramme 50

Abbildungsverzeichnis 68

Verzeichnis der Tabellen 69

1. Einführung

Der Einsatz von Sprache zur Informationsabfrage bei Datenbanken oder zur Steuerung komplexer Maschinen und Prozesse ist seit einiger Zeit Gegenstand intensiver Forschung und Entwicklung. Dabei hat sich gezeigt, dass mit Spracherkennungssystemen, die auf einer isolierten Sprechweise von Kommandowörtern basieren, eine nur unbefriedigende Leistungsfähigkeit und damit geringe Akzeptanz der Spracherkennungstechnologie erreicht wird. In einer realen Umgebung werden Kommandowörter meist in Höflichkeitsfloskeln eingebettet oder sind mit Versprechern und nichtsprachlichen Äußerungen behaftet. Aktuelle Forschungsanstrengungen konzentrieren sich daher auf die Entwicklung von Verfahren zur sogenannten Schlüsselworterkennung. Schlüsselworterkenner erlauben die Erkennung eines Vokabulars von Kommandowörtern in einem natürlichsprachlichen Kontext. Nicht im Vokabular enthaltene Wörter sowie nichtsprachliche Äußerungen sollen dabei ignoriert werden.

Zwei grundlegende Konzepte zur Schlüsselworterkennung werden derzeit diskutiert. Ein Ansatz beruht auf dem Vergleich von Bewertungen einer Äußerung durch Schlüsselwortmodelle mit vorgegebenen Schwellenwerten. Die Mehrzahl der Arbeiten auf diesem Gebiet basiert hingegen auf dem Ansatz, sowohl Modelle der Schlüsselwörter als auch Modelle für die zu ignorierenden Äußerungen vorzusehen und deren Bewertungen miteinander konkurrieren zu lassen. Modelle zu ignorierender Äußerungen werden häufig als Füllmodelle bezeichnet.

Für den Aufbau der Schlüsselwort- und Füllmodelle existieren zwei unterschiedliche Konzepte: Wird eine wortorientierte Modellierung gewählt, so werden die Füllmodelle anhand möglichst vieler verschiedener Äußerungen trainiert, die keines der Schlüsselwörter enthalten, während die Modelle der Schlüsselwörter an Äußerungen ebendieser trainiert werden. Eine wortorientierte Modellierung schränkt jedoch die Flexibilität des Schlüsselworterkennersystems erheblich ein, denn sowohl die Aktivierung neuer Wörter als auch der Einsatz unter veränderten Rahmenbedingungen bedingt ein Nachtraining des Systems anhand geeigneter Daten, deren Akquirierung sehr kostenaufwendig und zeitintensiv ist.

Bei einer phonemorientierten Modellierung werden die Schlüsselwortmodelle durch ihre phonetische Transkription repräsentiert, die in einem Wortlexikon abgelegt ist. Die Füllmodelle

können durch die uneingeschränkte Abfolge beliebiger Phoneme realisiert werden. Der phonemorientierte Aufbau eines Systems zur Schlüsselworterkennung erlaubt eine hohe Flexibilität, da eine Erweiterung oder Modifikation des Vokabulars einfach durch Änderung des Wortlexikons erreicht werden kann.

Im Rahmen der vorliegenden Arbeit wurden phonemorientierte Ansätze zur Schlüsselworterkennung mit und ohne Füllmodelle implementiert und untersucht. Bei der Implementation der Verfahren wurde eine hohe Effizienz angestrebt, da das Schlüsselworterkennersystem für den Echtzeitbetrieb auf verfügbarer Hardware geeignet sein sollte. Die Leistungsfähigkeit der phonemorientierten Verfahren wurde unter Verwendung realer Telefonsprache optimiert und mit der eines auf Wortmodellen basierenden Ansatzes verglichen.

In den folgenden Abschnitten werden zunächst einige Grundlagen der automatischen Spracherkennung mit den verwendeten Hidden-Markov-Modellen erläutert und darauf aufbauend die beiden untersuchten Verfahren zur Schlüsselworterkennung dargestellt. Anschließend werden die Eigenschaften der konkreten Implementation unter besonderer Berücksichtigung der Laufzeitoptimierungen vorgestellt. Danach werden die verwendeten Datenkorpora und Modelle beschrieben, und die Ergebnisse der Untersuchungen zur Erkennungsleistung beider Schlüsselworterkennungsverfahren bei unterschiedlichen Randbedingungen genannt, wobei die Resultate der phonemorientierten Modelle mit denen der wortorientierten verglichen werden.

2. Signalaufbereitung und Merkmalsextraktion

Jedes automatische Spracherkennungssystem verwendet als Eingangssignal digitalisierte Sprache. Systeme, die in der unmittelbaren Umgebung des Anwenders (z.B. seinem Arbeitsplatzrechner) angesiedelt sind, verwenden nahezu den vollen Frequenzumfangs der menschlichen Stimme, sie digitalisieren daher meist mit Abtastraten um 16kHz. Bei den zur Aufnahme verwendeten Mikrophonen handelt es sich vorzugsweise um solche, die dicht am Mund des Sprechenden lokalisiert sind, um Störgeräusche aus dem Umfeld des Benutzers relativ zum Nutzsignal gering zu halten. Erkennersysteme, die via Telefon eingehende Sprache verarbeiten, können aufgrund der Übertragungsbandbreite gewöhnlicher Telefonstrecken nur den Frequenzbereich von 300Hz bis 3400Hz nutzen. Obwohl es dem Menschen nicht schwer fällt, am Telefon zu kommunizieren, stellt dieser Umstand eine wesentliche Komplikation für automatische Spracherkennungssysteme dar. Anders als der Mensch, der Äußerungen inhaltlich versteht, und dadurch bei der Interpretation des Gehörten viele sinnlose Varianten ausschließen kann, verläßt sich ein automatischer Spracherkenner rein auf die statistische Auswertung akustischer Modelle. Daher ist es von besonderer Tragweite, dass sich durch Beschränkung auf Telefonbandbreite bestimmte Laute kaum noch voneinander unterscheiden lassen: So unterscheiden sich zum Beispiel das scharfe „s“ (/**s**/) und das „f“ (/**f**/) hauptsächlich oberhalb des Telefonbandbereiches.

Andererseits bietet das Telefon die Möglichkeit, von überall auf Informationsdienste zugreifen zu können. Daher wurde im Rahmen der vorliegenden Arbeit stets mit telefonbandbegrenzten Signalen gearbeitet.

Das digitalisierte und Telefonbandbegrenzte Signal wird zunächst durch einen Preemphasefilter $H_p(z)$ mit der Systemfunktion

$$H_p(z) = 1 - 0.95z^{-1} \qquad (1)$$

geleitet, um die Abstrahlcharakteristik der Lippen zu kompensieren. Aus dem Signal werden anschließend alle N' Abtastwerte Blöcke der Länge N gebildet, die um je $N_0 = (N - N')/2$ Abtastwerte mit dem vorangegangenen und nachfolgenden überlappen. Um Blockrandeffekte

bei der nachfolgenden Verarbeitung zu vermeiden, werden die Signalblöcke mit den Koeffizienten des HAMMING-Fensters[1]

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad ; \quad n = 0, 1, \ldots, N-1 \tag{2}$$

gewichtet:

$$s_t'(n) = w(n) \cdot s(tN' - N_0 + n) \quad ; \quad n = 0, 1, \ldots, N-1 \tag{3}$$

Zu jedem Signalblock werden die ersten P+1 Koeffizienten der Kurzzeit-Autokorrelationsfunktion berechnet,

$$R_t(k) = \sum_{i=0}^{N-k-1} s_t'(i) s_t'(i+k) \quad ; \quad k = 0, 1, \ldots, P \tag{4}$$

aus denen anschließend mit Hilfe des DURBIN-Algorithmus[3]

$$\begin{aligned}
&\varepsilon^0 = R_t(0) \\
&\text{für } i = 1, 2, \ldots, P \\
&\qquad k_i = \frac{R_t(i) - \sum_{m=1}^{i-1} a_m^{i-1} R_t(i-m)}{\varepsilon^{i-1}} \\
&\qquad a_i^i = k_i \\
&\qquad a_j^i = a_j^{i-1} - k_i a_{i-j}^{i-1} \quad ; \quad j = 1, 2, \ldots, i-1 \\
&\qquad \varepsilon^i = (1 - k_i^2)\varepsilon^{i-1}
\end{aligned} \tag{5}$$

Koeffizienten a_i^P für einen linearen Prädiktor der Ordnung P

$$\hat{s}_P(n) = \sum_{i=1}^{P} a_i^P s(n-i) \tag{6}$$

berechnet werden können, die hinsichtlich einer Minimierung der mittleren Prädiktionsfehlerleistung

$$\varepsilon^P = E\left\{\left(\hat{s}_P(n) - s(n)\right)^2\right\} \tag{7}$$

optimal sind.

Aus den Prädiktorkoeffizienten a_i lassen sich gemäß

$$c_i = -a_i - \frac{1}{i}\sum_{j=1}^{i-1}(i-j)c_{i-j}\,a_i \;\;;\;\; i = 1, 2, \ldots, P \tag{8}$$

die Cepstralkoeffizienten c_i des mit einem autoregressiven Modell approximierten Sprachsegments errechnen. Die Cepstralkoeffizienten sind geeignete Merkmale für die Spracherkennung, da sie die spektralen Anteile des Sprachsignals im Zeitbereich widerspiegeln[4].

Aus der logarithmierten und gemäß

$$e_t = \max\left(-1.0,\, 1.0 - 0.1\left(\max_t(\ln R_t(0)) - \ln R_t(0)\right)\right) \tag{9}$$

über die gesamte Äußerung normierten Signalenergie, den Cepstralkoeffizienten erster bis P-ter Ordnung sowie der zeitlichen Ableitung dieser Werte wird für jeden Sprachsignalblock ein Merkmalsvektor

$$\underline{o}_t = \{e_t, c_1, c_2, \ldots, c_P, \dot{e}_t, \dot{c}_1, \dot{c}_2, \ldots, \dot{c}_P\} \tag{10}$$

zusammengesetzt. Die Berechnung der Ableitung einer Merkmalskomponente x erfolgt gemäß[1]

$$\dot{x}_t := \frac{x_{t+1} - x_{t-1} + 2(x_{t+2} - x_{t-2})}{10} \tag{11}$$

1. Der Grund für die Wahl dieser etwas willkürlich erscheinenden Form der Energienormierung und zeitlichen Ableitung der Koeffizienten liegt in der gewünschten lese-kompatibilität mit Modellen, die durch HTK trainiert wurden. So ist z.B. die Division durch einen konstanten Faktor in (11) völlig entbehrlich. Siehe auch [9]

3. Struktur des Sprachmodells

Ein bewährtes Verfahren zur Modellierung der äußert variablen menschlichen Sprache stellt die Verwendung sogenannter „Hidden-MARKOV-Modelle“ (HMM) dar. Bei diesem zweistufigen stochastischen Prozeß geht man davon aus, dass sich der (fiktive) Sender des Sprachsignals zu jedem Zeitpunkt t in einem von N diskreten Zuständen $S_1, S_2, ..., S_N$ befindet, was zur Beobachtung des Merkmalsvektors $\underline{o}_t$ durch den Empfänger mit einer durch $b_j(\underline{o}_t)$ bewerteten Wahrscheinlichkeit führt. Die Wahrscheinlichkeit, dass der Sender vom Zustand S_i zum Zeitpunkt t in den Zustand S_j bei $t+1$ wechselt, ist durch die Übergangswahrscheinlichkeit a_{ij} gegeben.

Für die Spracherkennung haben sich HMM bewährt, deren Übergangswahrscheinlichkeiten a_{ij} nur dann ungleich Null sind, wenn $j=i$ oder $j=i+1$, sogenannte „strikte links-rechts“ Modelle.

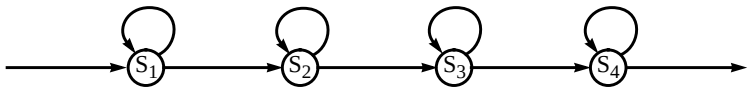

Abbildung 1: Übergangsstruktur eines strikten links-rechts HMM

Die vom Sender eingenommene Zustandsfolge

$$Q = \{q_1, q_2, \ldots, q_T\}, q_i \in \{S_1, S_2, \ldots, S_n\} \qquad (12)$$

bleibt dem Empfänger verborgen, sie ist eine Markovkette erster Ordnung, wenn die Bewertung b_j nur vom aktuell angenommenen Zustand abhängt.

Die Observablenwahrscheinlichkeitsbewertung der kontinuierlichen HMM (KHMM) basiert auf n-dimensionalen Wahrscheinlichkeitsdichtefunktionen, die den Raum der Merkmalsvektoren ausfüllen. Dabei ist jedem Zustand S_j eine Anzahl M von Warscheinlichkeitsdichtefunktionen $V_j^m(\underline{o}_t)$ und Gewichten c_j^m zugeordnet, aus deren Überlagerung sich die Bewertung berechnet:

$$b_j(\underline{o}_t) = \sum_{m=1}^{M} c_j^m V_j^m(\underline{o}_t) \qquad (13)$$

Als Wahrscheinlichkeitsdichtefunktionen werden Gaußdichten verwendet, die durch einen Mittelwertsvektor $\underline{\mu}$ und eine Kovarianzmatrix $\underline{\underline{\Sigma}}$ charakterisiert sind:

$$V_j^m(\underline{o}_t) = \frac{1}{\sqrt{(2\pi)^n \cdot \det(\underline{\underline{\Sigma}}_j^m)}} \cdot e^{-\frac{1}{2}(\underline{o}_t - \underline{\mu}_j^m)^T (\underline{\underline{\Sigma}}_j^m)^{-1} (\underline{o}_t - \underline{\mu}_j^m)} \tag{14}$$

Zur vollständigen Beschreibung eines HMM λ werden schließlich noch die Wahrscheinlichkeiten π_i benötigt, dass der Zustand S_i der zu Beginn vom Sender eingenommene ist. Bei gegebenen N und M sind die Parameter eines KHMM

$$\lambda = (\pi_i, a_{ij}, b_i) = (\pi_i, a_{ij}, c_j^m, \mu_j^m, \underline{\underline{\Sigma}}_j^m) \tag{15}$$

Sie können an bekanntem Sprachmaterial mit Verfahren wie dem BAUM-WELCH-Algorithmus[5] trainiert werden.

4. Bewertung und Klassifikation der Merkmalsfolge

Liegen eine Äußerung $O = \{o_1, o_2, \ldots o_T\}$ sowie Modelle λ_w für jedes mögliche Wort w vor, so wurde mutmaßlich jenes Wort W' geäußert, für das die Bewertung

$$c_w(Q, O) = \pi^w_{q_1} b^w_{q_1}(\underline{o}_1) a^w_{q_1 q_2} \cdot b^w_{q_2}(\underline{o}_2) a^w_{q_2 q_3} \cdot \ldots \cdot b^w_{q_T}(\underline{o}_T) \tag{16}$$

summiert über alle möglichen Zustandsfolgen $\{Q\}$ maximal ist:

$$W' = \underset{w}{\operatorname{argmax}} \left(\sum_{\{Q\}} c_w(Q, O) \right) \tag{17}$$

Die Berechnung der Summe der Bewertungen aller möglichen Zustandsfolgen ist durchaus möglich[7], es hat sich jedoch gezeigt, dass es die Erkennungsrate nicht nennenswert vermindert, wenn statt dieser Summe nur die Bewertung

$$C_w(O) = c_w(Q_{opt}(w, O), O) \tag{18}$$

der für das jeweilige Modell optimalen Zustandsfolge

$$Q_{opt}(w, O) = \underset{Q}{\operatorname{argmax}}(c_w(Q, O)) \tag{19}$$

ausgewertet wird, also auf das Wort W mit

$$W = \underset{w}{\operatorname{argmax}}(C_w(O)) \tag{20}$$

erkannt wird. Die Bewertung $C_w(O)$ läßt sich durch den VITERBI-Algorithmus[6] berechnen: Nach einer Initialisierung für den ersten Zeitpunkt

$$\delta^w_j(1) = \pi^w_j b^w_j(\underline{o}_t) \quad ; \quad j = 1, 2, \ldots, N_w \tag{21}$$

wird zu jedem Zeitpunkt t und jeden Zustand S_j nur die maximal erreichbare Zwischenbewertung

$$\delta^w_j(t) = \max_i (\delta^w_i(t-1) \cdot a^w_{ij})\, b^w_j(\underline{o}_t) \quad ; \quad j = 1, 2, \ldots, N_w \tag{22}$$

berechnet.

Die wortabhängige Bewertung $C_w(O)$ der gesamten Äußerung liegt dann zum letzten Zeitpunkt als

$$C_w(O) = \delta^w_{N_w}(T) \tag{23}$$

beim Endzustand der Wörter vor.

Möchte man außer $C_w(O)$ auch Q_{opt} ermitteln, so muß zu jedem Zeitpunkt und jeden Zustand auch der optimale Vorgängerzustand gespeichert werden; für die Erkennung ist diese Zustandsfolge innerhalb von Wortmodellen jedoch irrelevant.

5. Phonembasierte Schlüsselworterkennung

Für die Schlüsselworterkennung ist es nicht ausreichend berechnen zu können, welchem einer Reihe von Wortmodellen λ_w eine Äußerung *O* am besten entspricht. Vielmehr muß ein Schlüsselworterkenner prüfen, ob und wo sich innerhalb *O* Abschnitte befinden, die einem der λ_w so gut entsprechen, dass dies auf eine Äußerung des Wortes schließen läßt.

Man kann allgemein zwischen zwei Unterarten der Schlüsselworterkennung unterscheiden: Bei dialoggeführten Systemen wird vom Benutzer oft erwartet, dass seine Antwort auf Fragen des Systems genau ein Schlüsselwort enthält, im folgenden wird dieses Szenario als „1-Schlüsselwort-Erkennung“ bezeichnet. Ein solches System kann anhand des Verhältnisses von korrekt erkannten zu den insgesamt in der Äußerung enthaltenen Schlüsselwörtern bewertet werden, diese Größe wird als Erkennungsrate bezeichnet.

Andere Systeme, insbesondere solche die nur „mithören“, können keine einschränkenden Annahmen machen, und müssen davon ausgehen, dass sich in jeder Äußerung Null bis n Schlüsselwörter befinden. Dieses Szenario wird im folgenden „n-Schlüsselwort-Erkennung“ genannt. Zur Bewertung eines n-Schlüsselwort-Erkenners reicht die Erkennungsrate nicht aus, da mehr oder weniger Worthypothesen geliefert werden können, als tatsächlich Schlüsselwörter in der Äußerung enthalten waren. Daher werden solche Systeme zusätzlich durch die Zahl der Fehlalarme bewertet: Ein Fehlalarm liegt vor, wenn der Erkenner ein Schlüsselwort an einer Stelle vermutet, an der dieses nicht geäußert wurde. Meist wird die Zahl der Fehlalarme mit der Gesamtlänge der Äußerungen und der Anzahl der im Vokabular enthaltenen Schlüsselwörter ins Verhältnis gesetzt, man spricht dann von Fehlalarmen pro Schlüsselwort (englisch: keyword) und Stunde: $\frac{\mathrm{FA}}{\mathrm{KW} \cdot \mathrm{h}}$

Spracherkennungssysteme sollen Wort- oder Satzhypothesen liefern; die kleinste im Resultat interessierende Einheit ist in der Regel das Wort. Es liegt daher zunächst nahe, die zur automatischen Spracherkennung benötigten Modelle auf Wortbasis zu trainieren. Dabei treten jedoch eine Reihe von Problemen auf:

- Zum Training sprecherunabhängiger Erkennersysteme wird eine große Anzahl Äußerungen verschiedener Personen benötigt. Es sind daher speziell für jedes zu

erkennende Wort umfangreiche Sprachproben aufzunehmen - ein langwieriger und teurer Vorgang.

- Schon bei mäßig großen Wortschätzen wird die Zahl der Modellparameter enorm groß, daher steigt der Speicherplatzbedarf und die benötigte Rechenzeit bei der Erkennung, was die Realisierung teuer wenn nicht gar technisch unmöglich macht.

- Eine Veränderung bzw. Erweiterung des Wortschatzes durch den Anwender ist praktisch nicht möglich, da er weder die benötigten Trainingsdaten, noch die Möglichkeit hat, die aufwendigen Modellberechnungen durchzuführen.

Möchte man diese Probleme vermeiden, so können Modelle für Wortuntereinheiten verwendet werden, die in der Anwendung zu kompletten Wörtern kombiniert werden. Solche Modelle lassen sich unabhängig vom benötigten Wortschatz an existierenden Sprachdatensammlungen trainieren. Die Zahl der Modellparameter hängt im wesentlichen[1] nur noch von der Anzahl verschiedener Wortuntereinheiten ab. Eine Veränderung bzw. Erweiterung des Wortschatzes wird ohne Neuerfassung von Trainingsdaten möglich, wenn die Menge der Wortuntereinheiten eine vollständige Lautbasis der Sprache darstellt.

Ein prinzipieller Nachteil aus Wortuntereinheiten zusammengesetzter Modelle ist es, dass die Adaption des Modells an die tatsächlichen Realisationen des Wortes durch die Sprecher niemals so exakt sein kann, wie es bei einem Ganzwortmodell möglich ist.

Im Rahmen dieser Arbeit wurden Schlüsselworterkenner implementiert und experimentell untersucht, die als Wortuntereinheiten kontextunabhängige Phoneme verwenden, ferner wurden auch Ganzwortmodelle der Schlüsselwörter anhand bekannter Äußerungen trainiert und zu Vergleichszwecken getestet. Es wurden zwei Verfahren zur Schlüsselworterkennung erprobt: Eines verwendet einzelne Phoneme als Füllmodelle (PAF) für die unbekannten Wörter in einer Grammatik, das andere arbeitet als grammatikfreie Schlüsselworterkennung (GFSE) mit unabhängig voneinander und unabhängig vom Kontext ausgewerteten Wortmodellen.

1. Selbstverständlich muß auch die Information gespeichert werden, aus welchen Wortuntereinheiten die Wörter des Wortschatzes bestehen, jedoch ist die dafür benötigte Zahl Parameter sehr klein im Vergleich zur Anzahl Parameter für ein komplettes Wort- oder Wortuntereinheitenmodell.

5.1. Phoneme als Füllmodelle

Da alle Wörter, also auch die unbekannten nicht-Schlüsselwörter, aus Phonemen $p_i \in P$ bestehen, kann man eine Erkennung der gesamten Äußerung versuchen, wobei eine Grammatik vorgegeben wird, die außer den Schlüsselwörtern $z_i = \{p_{z_i 1}, \ldots, p_{z_i N_{z_i}}\}$ auch alle einzelnen Phoneme p_i als „Wörter“ enthält. Unter Grammatik versteht man in diesem Zusammenhang die durch Definition von Wortübergangswahrscheinlichkeiten gebildete Struktur möglicher Wortfolgen.

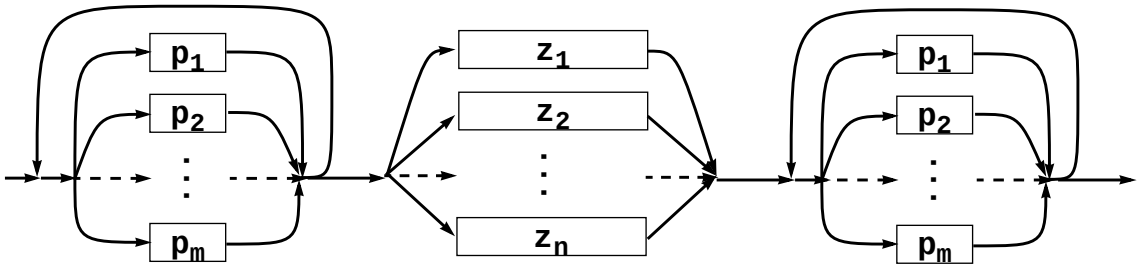

Abbildung 2: PAF Grammatik für die 1-Schlüsselwort-Erkennung

Je nach Anwendungsfall erzwingt die Grammatik exakt ein Schlüsselwort pro Äußerung (Abbildung 2), oder die Zahl der zu erkennenden Schlüsselwörter wird offen gelassen (Abbildung 3).

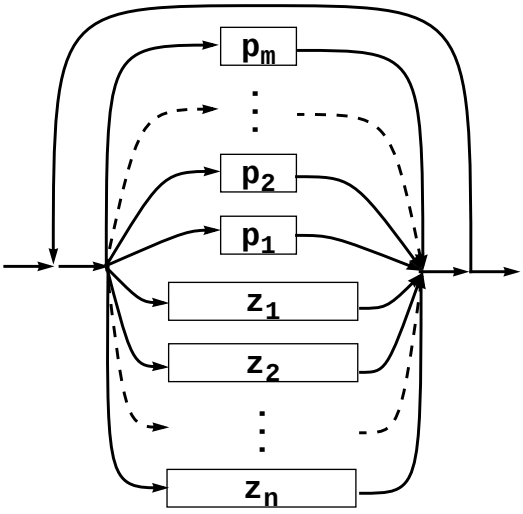

Abbildung 3: PAF Grammatik für die n-Schlüsselwort-Erkennung

Die PAF Methode läßt sich auf die Erkennung kontinuierlicher Sprache zurückführen, die wie die Einzelworterkennung mit dem VITERBI-Algorithmus erfolgen kann, wobei die Grammatik die Endzustände $V_{w_i} = S_{N_w w_i}$ aller λ_w mit den Anfangszuständen $U_{w_j} = S_{1 w_j}$ aller λ_w durch Übergangswahrscheinlichkeiten $a_{V_{w_i} U_{w_j}}$ verbindet. Um am Ende der Erkennung rekonstruieren zu können, welche λ_w innerhalb von Q_{opt} lagen, wird während des VITERBI-Algorithmus zu jedem gemäß (22) berechneten $\delta_j^w(t)$ ein Zeitpunkt

$$\gamma_j^w(t) = \begin{cases} t \text{ für } j = 1 \\ \gamma_x^w(t-1) \text{ für } j \neq 1 \end{cases} \qquad (24)$$

$$\text{mit } x = \underset{i}{\operatorname{argmax}} \left(\delta_i^w(t-1) \cdot a_{ij}^w\right) b_j^w(\underline{o}_t) \quad ; \quad i = 1, 2, \ldots, N_w$$

gespeichert, der den Eintritt in das aktuelle Wortmodell markiert.

$\delta_j^w(t)$ und $\gamma_j^w(t)$ werden innerhalb der Wortmodelle nur für Berechnungen zum Zeitpunkt t+1 benötigt, und können daher bereits bei t+2 wieder verworfen werden. Die den Wortmodellendzuständen V_{w_j} zugeordneten δ und γ werden jedoch aufbewahrt:

$$\begin{aligned} \eta_{w_i}(t) &= \gamma_{N_{w_i}}^{w_i}(t) \\ \theta_{w_i}(t) &= \delta_{N_{w_i}}^{w_i}(t) \end{aligned} \qquad (25)$$

Aus diesen Daten läßt gemäß folgendem Verfahren eine Ergebnisliste E erstellen, die nacheinander alle Wörter enthält, deren Modelle in Q_{opt} durchlaufen wurden[1]:

$$\begin{aligned} &y = T \\ &x = \underset{i}{\operatorname{argmax}}(\theta_{w_i}(y)) \\ &E = \{w_x\} \\ &\text{solange } y \neq 1 \\ &\qquad y = \eta_{w_x}(y) - 1 \\ &\qquad z = \underset{i}{\operatorname{argmax}} \left(\theta_{w_i}(y) \cdot a_{V_{w_i} U_{w_x}}\right) \\ &\qquad E = w_z \oplus E \\ &\qquad x = z \end{aligned} \qquad (26)$$

1. Man könnte auf die Speicherung der $\theta_{w_i}(t)$ verzichten, und stattdessen direkt den in (26) mit z bezeichneten optimalen Vorgänger eines Wortes speichern, der ja schon während des VITERBI-Algorithmus ermittelt wird. Die Maximum-Bildung ist in Relation zur Gesamtrechenzeit jedoch sehr unaufwendig, und man hält sich durch die Speicherung der $\theta_{w_i}(t)$ die Möglichkeit offen, ggf. nicht nur das optimale Ergebnis E, sondern noch weitere („n-beste“) Hypothesen generieren und nachbewerten zu können.

Bei der PAF Methode werden die im Ergebnis enthaltenen einzelnen Phoneme ignoriert, es bleiben somit nur die erkannten Schlüsselwörter übrig.

Bei Verwendung der in Abbildung 3 gezeigten Grammatik kann die Bewertung eines Schlüsselwortmodelles z_i nicht höher sein als die der entsprechenden Abfolge $\{ p_{z_i 1}, \ldots, p_{z_i N_{z_i}} \}$ freier Phonemmodelle. Aus diesem Grund wird die Zwischenbewertung nach (22) um einen Faktor f_w erweitert, der eine Bevorzugung der Schlüsselwortmodelle ermöglicht:

$$\delta_j^w(t) = \max_i \left(\delta_i^w(t-1) \cdot a_{ij}^w \right) b_j^w(\underline{o}_t) f_w \quad ; \quad j = 1, 2, \ldots, N_w \qquad (27)$$

Für die n-Schlüsselwort-Erkennung wählt man $f_{w=\{p_i\}} = 1$ und $f_{w=z_i} > 1$ so, dass möglichst viele Schlüsselwörter unter Einhaltung vorgegebener maximaler Fehlalarmraten erkannt werden. Ein niedriger Wert für f_w führt zu einer geringen Fehlalarmrate, aber auch zu einer geringen Erkennungsrate, umgekehrt geht die Fehlalarmrate für $f_w \to \infty$ ebenfalls gegen unendlich, während die Erkennungsrate auf 100% steigt.

Auch bei der 1-Schlüsselwort-Erkennung unter Verwendung der in Abbildung 2 gezeigten Grammatik müssen für optimale Ergebnisse die Schlüsselwörter bevorzugt werden, da sie nicht immer gemäß der Standardtranskription gesprochen werden, und besser passende Folgen einzelner Phoneme höher bewertet werden können. Da dieses Problem mit zunehmender Länge der Schlüsselwörter wahrscheinlicher auftritt, führte ein Verzicht auf den Faktor f_w bei der 1-Schlüsselwort-Erkennung zu einer unerwünschten Bevorzugung kurzer Schlüsselwörter. Darüberhinaus eignen sich die f_w auch zu einem Ausgleich der im Mittel unterschiedlich hohen Bewertungen bestimmter Phonemmodelle. Die f_w werden für die 1-Schlüsselwort-Erkennung nur hinsichtlich einer maximalen Erkennungsrate gewählt, und hängen stark voneinander ab. Wird f_w für ein Wort erhöht, so sinkt die Erkennungswahrscheinlichkeit aller anderen Wörter. Unabhängig davon, wo innerhalb der Äußerung sich zwei Schlüsselwörter befinden, treten ihre Bewertungen in Konkurrenz zueinander, da die Grammatik nur ein Schlüsselwort pro Äußerung zuläßt.

5.2. Grammatikfreie Schlüsselworterkennung

Der PAF Methode zur Schlüsselworterkennung mangelt es an einem analytischen Ansatz, geeignete f_w für die Bevorzugung der Schlüsselwörter auf Basis der phonetischen Transkriptionen berechnen zu können. In [8] wird ein anderes Verfahren zur Schlüsselworterkennung

beschrieben, das dieses Problem zumindest für den Fall der n-Schlüsselwort-Erkennung lösen soll.

Auch bei diesem Verfahren werden aus den Phonemmodellen zunächst Modelle der Schlüsselwörter zusammengefügt. Diese werden dann allerdings nicht in den Kontext einer Grammatik gestellt, wie es für eine Bewertung mit dem VITERBI-Algorithmus notwendig wäre. Stattdessen werden alle Schlüsselwortmodelle unabhängig voneinander mit einem modifizierten Verfahren bewertet. Ähnlich (22) wird hierbei jedem Zustand S_j zu jedem Zeitpunkt t eine Bewertung

$$\begin{aligned} \delta_j^w(t) &= \left(1-\frac{1}{L_x}\right)\delta_x^w(t-1) + \frac{1}{L_x}\left(\ln(a_{xj}^w) + B_j^w(\underline{o}_t)\right) \quad ; \quad j = 1, 2, \ldots, N_w \\ \text{mit } x &= \underset{i}{\operatorname{argmax}}\left(\left(1-\frac{1}{L_i}\right)\delta_i^w(t-1) + \frac{1}{L_i}\left(\ln(a_{ij}^w) + B_j^w(\underline{o}_t)\right)\right) \end{aligned} \tag{28}$$

sowie eine Länge

$$L_j^w(t) = L_x^w(t-1) + 1 \tag{29}$$

zugeordnet, wobei $B_j^w(\underline{o}_t)$ eine über alle Zustände aller Phonemmodelle zum aktuellen Zeitpunkt normierte und logarithmierte Variante der Bewertungszahl $b_j^w(\underline{o}_t)$ ist:

$$B_j^w(\underline{o}_t) = \ln(b_j^w(\underline{o}_t)) - \max_{i,k}\left(\ln(b_k^{p_i}(\underline{o}_t))\right) \tag{30}$$

Zu jedem Zeitpunkt t soll an jedem Anfangszustand U_w ein neuer Pfad mit der Länge L=1 und Übergangswahrscheinlichkeit a=1 beginnen können, so dass für die Bewertungen der Anfangszustände in Ergänzung zu (28) und (29) gilt:

$$\delta_1^w(t) = \max\left(\left(1-\frac{1}{L_x}\right)\delta_x^w(t-1) + \frac{1}{L_x}\left(\ln(a_{xj}^w) + B_1^w(\underline{o}_t)\right), B_1^w(\underline{o}_t)\right) \tag{31}$$

und

$$L_1^w(t) = \begin{cases} 1 \text{ falls } B_1^w(\underline{o}_t) > \left(1-\frac{1}{L_x^w}\right)\delta_x^w(t-1) + \frac{1}{L_x^w}\left(\ln(a_{xj}^w) + B_1^w(\underline{o}_t)\right) \\ L_x^w(t-1) + 1 \text{ sonst} \end{cases} \tag{32}$$

Durch die Normierung über die Pfadlänge wird sichergestellt, dass die Bewertungen unterschiedlich langer Pfade stets vergleichbar bleiben.

Das dem Endzustand zugeordnete $\delta^w_{N_w}(t)$ eines Wortmodells *w* liefert zu jedem Zeitpunkt *t* die Bewertung

$$C(O^w_t) = \delta^w_{N_w}(t) \tag{33}$$

einer bis zu diesem Zeitpunkt geäußerten Merkmalsvektorenfolge O^w_t, deren Länge

$$L(O^w_t) = L^w_{N_w}(t) \tag{34}$$

durch die optimale Übereinstimmung mit dem Wortmodell determiniert ist, da durch Gleichung (28) sichergestellt wird, dass alle Merkmalsvektorenfolgen $\tilde{O}^w_t$, die zu einem anderen Startzeitpunkt als

$$\tau(O^w_t) = t - L(O^w_t) \tag{35}$$

begannen, eine geringere Bewertung erhalten hätten.

Die n-Schlüsselwort-Erkennung funktioniert nun wie folgt: Vor dem Erkennungsvorgang wird eine leere Ergebnisliste *E* angelegt. Zu jedem Zeitpunkt *t* wird für jedes Wort *w* überprüft, ob $C_w(O^w_t)$ größer einem konstanten Schwellenwert G_w ist. Ist dies der Fall, dann wird überprüft, ob *E* bereits eine Äußerung O^w_y enthält, die wie O^w_t zum Zeitpunkt $\tau(O^w_t)$ begann. Ist auch dies gegeben, und $C_w(O^w_t)$ ist größer als die gespeicherte Bewertung $C_w(O^w_y)$, so wird O^w_y in der Ergebnisliste durch O^w_t ersetzt. Enthielt *E* keine zum gleichen Zeitpunkt wie O^w_t begonnene Äußerung von *w*, so wird O^w_t der Ergebnisliste *E* hinzugefügt. Auf diese Art und Weise wird sichergestellt, dass *E* am Ende des Erkennungsvorgangs nicht mehrere Äußerungen eines Wortes enthält, die zum gleichen Zeitpunkt beginnen, was sonst aufgrund des typischen zeitlichen Verlaufs der Bewertungen, wie er in Abbildung 4 dargestellt ist, häufig der Fall wäre: In der Umgebung des Zeitpunktes maximaler Bewertung eines Schlüsselwortes befinden sich in aller Regel noch weitere Zeitpunkte, zu denen die Bewertung über G_w lag, wobei auch lokale Minima und Maxima über dem Schwellenwert beobachtet werden. Es ist nicht notwendig, $C_w(O^w_t)$ mit solchen Bewertungen des gleichen Schlüsselwortes zu vergleichen, die zu einem Zeitpunkt ungleich $\tau(O^w_t)$ begannen, da sich die postulierten Wortanfänge während der Erkennung nur selten ändern, wie aus den unteren Kurven in Abbildung 4 ersichtlich ist.

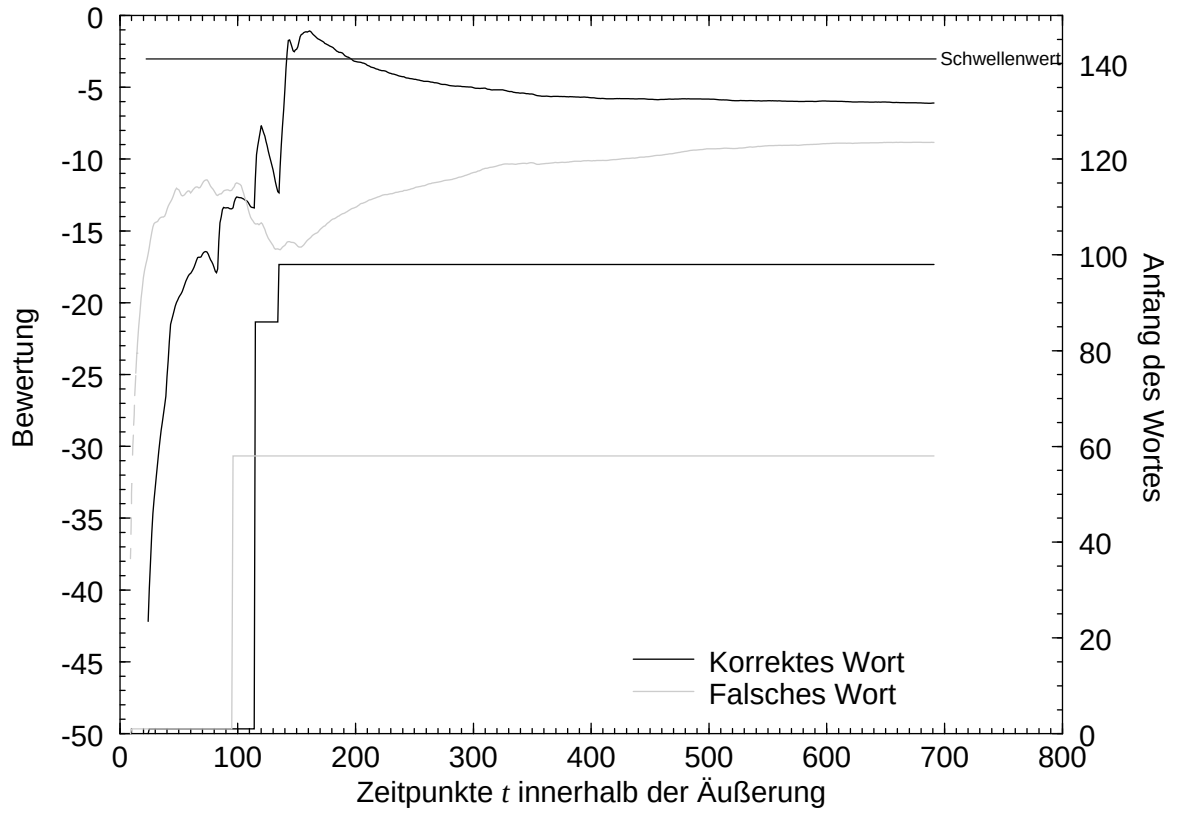

Abbildung 4: Exemplarischer Verlauf der Wortbewertungen $C_w(O_t^w)$ eines in der Äußerung enthaltenen sowie eines nicht enthaltenen Wortes. Die unteren Kurven geben zu jedem Zeitpunkt den Wortbeginn $\tau(O_t^w)$ an. Die Äußerung endet mit einer längeren Pause, daher die Annäherung der Bewertungskurven

Für die 1-Schlüsselwort-Erkennung wird das Wort *W* mit dem absoluten Maximum der Bewertungen aller Wörter *w* zu allen Zeitpunkten *t* gesucht:

$$W = \underset{w}{\operatorname{argmax}}\left(\max_t\left(C_w(O_t^w)\right)\right) \tag{36}$$

Da die Schlüsselwörter im Mittel jedoch recht unterschiedlich hoch bewertet werden, bedarf es noch eines konstanten, wortabhängigen Korrekturterms K_w:

$$W = \underset{w}{\operatorname{argmax}}\left(\max_t\left(C_w(O_t^w)\right) + K_w\right) \tag{37}$$

Die K_w hängen wie die f_w bei der 1-Schlüsselwort-Erkennung nach der PAF Methode stark voneinander ab, während die G_w für die n-Schlüsselworterkennung untereinander völlig unabhängig sind.

In [8] findet sich folgender Vorschlag zur Berechnung der G_w auf Basis der phonetischen Transskriptionen der zu erkennenden Wörter: Die Bewertung X_{p_i} eines gesprochenen Phonems p_i ist als Zufallsvariable aufzufassen, da sie je nach Aussprache stark schwankt. Die Bewertung X_w eines Wortes $w = \{p_1, p_2, \ldots, p_n\}$ kann als Linearkombination

$$X_w = l_{p_1} X_{p_1} + l_{p_2} X_{p_2} + \ldots + l_{p_n} X_{p_n} \tag{38}$$

der Phonembewertungen angesehen werden, wobei der Einfluß der Phonemlängen L_{p_i} durch Gewichtung mit

$$l_{p_i} = \frac{L_{p_i}}{L_{p_1} + L_{p_2} + \ldots + L_{p_n}} \tag{39}$$

eliminiert werden kann.

Geht man davon aus, dass die Bewertungen X_{p_i} normalverteilt sind, läßt sich für jedes Phonem eine Wahrscheinlichkeitsdichtefunktion durch einen Mittelwert μ_p und eine Varianz σ_p der Form

$$N_p(\mu_p, \sigma_p) = \frac{1}{\sigma_p \sqrt{2\pi}} e^{\frac{(x-\mu_p)^2}{-2\sigma_p^2}} \tag{40}$$

angeben. Die entsprechende Wahrscheinlichkeitsdichtefunktion für die Verteilung der Bewertung des ganzen Wortes *w* lautet unter der vereinfachenden Annahme, dass die Phonembewertungen unabhängig voneinander sind:

$$N_w(\mu_w, \sigma_w) = \frac{1}{\sigma_w \sqrt{2\pi}} e^{\frac{(x-\mu_w)^2}{-2\sigma_w^2}} \tag{41}$$

mit

$$\mu_w = l_1\mu_1 + l_2\mu_2 + \ldots + l_n\mu_n \tag{42}$$

und

$$\sigma_w = \sqrt{(l_1\sigma_1)^2 + (l_2\sigma_2)^2 + \ldots + (l_n\sigma_n)^2} \tag{43}$$

Gibt man den Anteil *h* aller Äußerungen vor, der erkannt werden soll, so läßt sich mit Kenntnis von μ_w und σ_w ein Schwellenwert G_w berechnen, der zur vorgewählten Erkennungsrate führen sollte. Der Anteil *h* entspricht der Wahrscheinlichkeit $P(X_w > G_w)$, die gemäß

$$h = \int_{G_w}^{\infty} N_w(\mu_w, \sigma_w)dx = \frac{1 - \mathrm{erf}\left(\frac{G_w - \mu_w}{\sqrt{2}\sigma_w}\right)}{2} \tag{44}$$

berechnet werden kann. Aus (44) ergibt sich der zugehörige Schwellenwert

$$G_w = \mu_w + \sqrt{2}\sigma_w \mathrm{erf}^{-1}(1 - 2h) \tag{45}$$

Es wäre wünschenswert, in ähnlicher Weise den Schwellenwert auf Basis einer vorgegebenen Fehlalarmquote berechnen zu können, indem man einfach die Wahrscheinlichkeitsverteilungen für den Fall „Phonem wurde nicht gesprochen“ miteinander kombiniert. Dies ist aber nicht möglich: Die oben gemachte Annahme der Unabhängigkeit der Bewertungen aufeinanderfolgender Phoneme ist für den Fall „Phonem wurde nicht gesprochen“ unzutreffend. Nehmen wir an, in einem Schlüsselwort folgten p_1 und p_2 aufeinander. Es ist durchaus nicht unwahrscheinlich, dass ein Phonemmodell p_3 existiert, das zwar für einzeln oder in anderem Kontext gesprochene p_1 und p_2 keine hohe Bewertung erhält, wohl aber dann, wenn dem p_3 Modell die letzten Merkmalsvektoren einer Äußerung von p_1 und die darauffolgenden ersten Merkmalsvektoren einer Äußerung von p_2 zugeordnet werden. Außerdem existiert keine „mittlere Länge eines nicht gesprochenen Phonems“, die zur Bestimmung der Gewichtungskoeffizienten l_{p_i} notwendig wäre.

6. Simulationsprogramm zur Schlüsselworterkennung

Ein wesentlicher Teil der vorliegenden Arbeit lag in der konkreten Realisation eines Schlüsselworterkennungssystems als Programm für Unix-Workstations. Nur unter realistischen Bedingungen erweist sich z.B., ob die an gespeicherten Sprachdaten gemessenen Ergebnisse auf den Anwendungsfall übertragbar sind, ob der Rechenzeit- und Speicherplatzbedarf durch existierende Hardware befriedigt werden kann und ob unerwartete Störeinflüsse den Betrieb beeinträchtigen.

6.1. Übersicht der Programmstruktur

Mit Ausnahme der graphischen Benutzeroberfläche wurden alle Programmteile objektorientiert in C++ implementiert. Das System läuft unter diversen Unix-Derivaten (AIX, HP-UX, SunOS, Solaris, Linux) und kann unter diesen Betriebssystemen verfügbare Audio-Hardware zur Spracheingabe nutzen. Das System besteht im wesentlichen aus zwei Teilen: Ein via Kommandozeile zu startendes Programm namens „**EFSE**“[1], das zunächst - durch zahlreiche, im Anhang erläuterte Optionen gesteuert - ein Sprachmodell einliest. Das Sprachmodell kann dabei in einer bearbeitbaren Textrepräsentation oder, zur Beschleunigung des Ladevorgangs, als binär oder architekturunabhängig textuell kodierte Datei vorliegen. **EFSE** kann das gelesene Modell anschließend optional in einem der beiden letztgenannten Formate für spätere Benutzungen zurückschreiben. Sobald das Sprachmodell im Hauptspeicher des Rechners etabliert wurde, kann **EFSE** Äußerungen auf eine von verschiedenen Arten entgegennehmen:

- Dateien mit vorberechneten Merkmalsvektoren
- Dateien mit digitalisierten Sprachsignalen, diese können optional einer digitalen Filterung, z.B. zur Anpassung an Telefonkanalcharakteristika, unterzogen werden.
- Über betriebssystemeigene Kommunikationskanäle (Pipes) eingehende Datenpakete mit digitalisierten Sprachsignalen
- Via PVM[2] eingehende Datenpakete mit digitalisierten Sprachsignalen

1. „**E**xperimentierplattform für **F**lexible **S**chlüsselwort**E**rkenner“
2. „PVM“ steht für „Parallel Virtual Machine“, eine frei erhältliche Bibliothek von Funktionen, die eine einheitliche Schnittstelle zur Rechner- und Architekturübergreifenden Interprozeßkommunikation zur Verfügung stellt.

- Direkt über Mikrophon aufgenommene Sprachsignale

Jede Äußerung wird einer dem geladenem Sprachmodell entsprechenden Funktionen zur Erkennung übergeben, die daraus resultierenden Ergebnisse können

- in textueller Form ausgegeben
- via Pipe oder PVM an den Absender der Sprachsignaldaten zurückgegeben
- zur späteren Auswertung in einem „HTK Labelfile[1]“-ähnlichen Format gespeichert

werden. **EFSE** ist nicht auf Schlüsselworterkennung beschränkt, es kann auch zur Erkennung von kontinuierlicher Sprache oder Einzelwortäußerungen eingesetzt werden.

Das zweite ausführbare Programm des Systems namens „**ISEO**“[2] erzeugt mehrere, miteinander kommunizierende Unterprozesse, die im Zusammenspiel

- eine mit Tcl/Tk[3] realisierte graphische Benutzeroberfläche
- eine permanente Aufnahme und Filterung von Sprachsignalen durch vorhandene Audio-Hardware mit anschließender Äußerungsdetektion
- eine - auch auf mehrere Rechner verteilte - Erkennung der Äußerungen durch einen oder mehrere **EFSE** Prozesse

zur Verfügung stellen. Die Ergebnisse der Erkenner werden an das steuernde Tcl/Tk Programm geliefert, das diese ausgeben oder in beliebiger Art darauf reagieren kann.

Ein für den praktischen Einsatz brauchbares Spracherkennersystem sollte die Fähigkeit zur Echtzeitverarbeitung des Sprachsignals aufweisen. Bereits erwähnt wurde die Notwendigkeit einer permanenten Aufmerksamkeit des Systems. Es ist einem Benutzer nicht zuzumuten, nur auf Anforderung sprechen zu können oder selbst auf nichtsprachlichem Wege den Beginn seiner Äußerung anzuzeigen. Da ein Benutzer meist nicht wissen kann, ob das System gerade eine Erkennung durchführt, mußte gewährleistet werden, das Äußerungen auch dann nicht ver-

1. „HTK“ steht für „Hidden Markov Toolkit“, ein Programmsystem von Steve Young, das Training und Erprobung von Hidden-Markov Modellen erlaubt. Siehe [9]
2. „**I**nteraktive **S**prach**E**rkenner **O**berfläche“
3. „Tcl“ ist eine frei verfügbare, von Dr. John Ousterhout entwickelte Interpretersprache, „Tk“ eine ebenfalls frei verfügbare Befehlsbibliothek hierzu, die die plattformunabhängige Erstellung graphischer Benutzeroberflächen erlaubt.

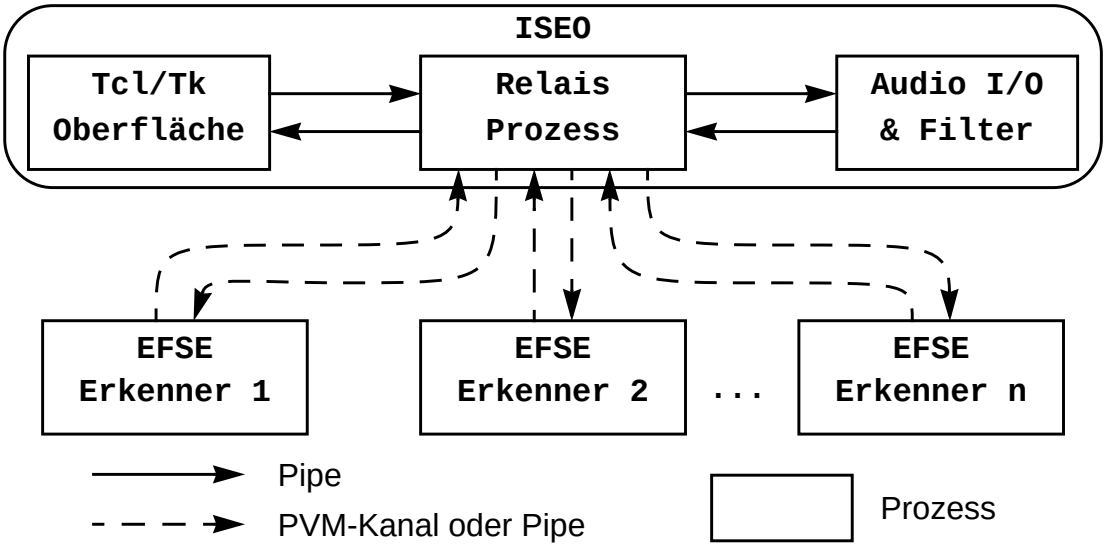

Abbildung 5: Kommunikationsstruktur des Erkennersystems **ISEO**

loren gehen. Dies wurde durch die in Abbildung 5 illustrierte arbeitsteilige Verwendung mehrerer, parallel laufender Prozesse erreicht. Der zentrale Relais-Prozess nimmt Kommandos von einem Prozess entgegen, der sich um die graphische Benutzeroberfläche kümmert, und gibt Rückmeldungen sowie Resultate der Erkennerprozesse an diesen zurück. Der Relais-Prozess erhält regelmäßig Sprachsignalpakete von einem weiteren Prozess, der sich allein um die Aufnahme und Filterung dieser Daten kümmert. Die Sprachsignalpakete werden an einen oder mehrere externe **EFSE** Prozesse weitergeleitet, die ihrerseits Resultate zurückschicken. Die Möglichkeit, mehrere Spracherkenner parallel arbeiten zu lassen, wurde in erster Linie geschaffen, um einen direkten Vergleich der Ergebnisse verschiedener Sprachmodelle zu erlauben. Darüberhinaus könnte diese Möglichkeit aber auch eingesetzt werden, um nicht echtzeitfähige Spracherkenner abwechselnd auf mehreren Rechnern Äußerungen bearbeiten zu lassen, um die Antwortzeit so gering wie möglich zu halten.

Die Verwendung von Phonemmodellen zur Schlüsselworterkennung ermöglicht die Veränderung des Wortschatzes zur Laufzeit. **ISEO** erlaubt dies auf zweierlei Arten: Entweder kann der Benutzer ein Wort sprechen, und eine daraufhin vom Erkenner automatisch erzeugte Phonemfolge als neues Wort eintragen, oder der Benutzer gibt manuell die Transkription eines neuen Schlüsselwortes vor. Zur automatischen Erzeugung einer Phonemfolge arbeitet das System als kontinuierlicher Spracherkenner, die hierzu verwendete Grammatik enthält alle Phoneme als einzelne Wörter.

6.2. Optimierung des Rechenaufwands und der Rechenzeit

Eine wichtige Voraussetzung für die Akzeptanz eines Schlüsselworterkennungssystems ist eine geringe Reaktionszeit. Ein Benutzer geht schon nach einigen wenigen Sekunden ohne Reaktion davon aus, dass er nicht verstanden wurde. Zwar kann man versuchen, durch optische oder akustische Rückmeldung zu signalisieren, dass der Spracherkenner die Eingabe bearbeitet, doch ist dies für den Benutzer ungewohnt und wenig komfortabel.

Es gibt zwei Möglichkeiten, die Reaktionszeit des Systems gering zu halten: Beginnt man erst nach dem Ende einer Äußerung des Benutzers mit der Erkennung, so muß dieser Vorgang sehr schnell ablaufen, und die maximale Äußerungslänge muß so limitiert sein, dass die Gesamtrechenzeit zur Erkennung einer Äußerung unter der maximal tolerablen Antwortzeit des Systems bleibt. Kann man hingegen bereits während der Äußerung mit dem Erkennungsvorgang beginnen, so braucht man auf jeden Fall nur die Echtzeitbedingung einzuhalten, um beliebig lange Äußerungen nach Schlüsselwörtern durchsuchen zu können.

Durch permanente statistische Auswertung des Signalenergieverlaufs ist **ISEO** in der Lage, den ersten Sprachsignalblock bereits 120ms nach Äußerungsbeginn an die externen Spracherkennerprozesse zu schicken. Dabei adaptiert sich der Äußerungsdetektor selbsttätig an den Pegel des Umgebungsgeräusches. **EFSE** kann das Sprachsignal blockweise in Merkmalsvektoren umwandeln und verarbeiten[1], nähere Erläuterungen hierzu finden sich im Anhang.

Optimierungen sind in erster Linie an zeitkritischen Stellen des Programms anzubringen. Die Analyse durch einen Profiler ergibt für die Erkennung von Sprachsignaldateien durch **EFSE** unter Verwendung der Phonemmodelle bei der PAF Methode und 9 Schlüsselwörtern folgende Aufteilung der Rechenzeit:

Funktionsblock	Anteil an der Rechenzeit (%)
Filterung des Sprachsignals	13
Berechnung der Merkmalsvektoren	3

1. Die im Rahmen dieser Arbeit verwendeten Merkmalsvektoren enthalten allerdings eine Komponente, die sich nicht vor Ende der Äußerung berechnen läßt: Die über die gesamte Äußerung normierte Signalenergie (siehe (9) auf Seite 7). Ein Verzicht auf diese Komponente des Merkmalsvektors würde in Verbindung mit entsprechend trainierten Modellen eine zeitsynchrone Verarbeitung ermöglichen, die bei Verwendung der GFSE auch die Äußerungsdetektion überflüssig machen würde.

Funktionsblock	Anteil an der Rechenzeit (%)
Merkmalsbewertung	67
Zustandsbewertungen	14
Summe aller sonstigen Funktionen	3

Das Verhältnis der Rechenzeitanteile kann sich deutlich verändern, wenn Modelle mit sehr viel mehr oder weniger Parametern oder ein erheblich größeres Vokabular verwendet werden. So ergibt z.B. die Analyse durch den Profiler für die gleiche Erkennung mit den für diese Aufgabe trainierten Ganzwortmodellen, die nur ein Zehntel der Parameter haben:

Funktionsblock	Anteil an der Rechenzeit (%)
Filterung des Sprachsignals	49
Berechnung der Merkmalsvektoren	11
Merkmalsbewertung	25
Zustandsbewertungen	7
Summe aller sonstigen Funktionen	8

Obwohl die Aufteilung der Rechenzeit im einzelnen sehr unterschiedlich ist, wird in beiden Fällen der größte Teil der Rechenzeit von den gleichen vier Funktionen verbraucht.

Die Filterung des Sprachsignals *s(n)* erfolgt durch FIR-Filter der Form

$$\tilde{s}(n) = \sum_{i=0}^{K} s\left(n - \frac{N-1}{2} + i\right) k_i \qquad (46)$$

Die Rechenzeit einer solchen Filterung ist von der Ordnung $O(K \cdot n)$, und wird daher im wesentlichen von der Anzahl K der Filterkoeffizienten k_i bestimmt. Um die Telefonbandbreite des zu erkennenden Signals zu gewährleisten, und da Störgeräusche beim Signalempfang durch den Fernsprecher auch außerhalb der Telefonbandbreite auftreten, verwendeten wir für optimale Erkennungsergebnisse hintereinander einen Hochpassfilter mit 300Hz Grenzfrequenz und 127 Koeffizienten sowie einen Tiefpassfilter mit 3400Hz Grenzfrequenz und 61 Koeffizienten. Kann auf etwas Erkennungsrate verzichtet werden (je nach Modell bis zu 3%), so kann

ein Bandpassfilter mit nur 41 Koeffizienten anstelle der beiden vorgenannten eingesetzt werden, was eine Beschleunigung der Filterung um den Faktor 4.6 bewirkt. Die segmentweise Filterung wurde mit Hilfe von Ringpufferspeichern so implementiert, dass Kopien des zu filternden Sprachsignals vermieden werden.

Die zur Berechnung der Merkmalsvektoren beschriebenen Verfahren in (1) bis (11) sind algorithmisch ausoptimiert und konnten daher nur unter Berücksichtigung der Eigenschaften moderner Prozessoren und Compiler so programmiert werden, dass redundante Operationen und Speicherzugriffe weitestgehend vermieden werden.

Die Berechnung der Observablenbewertungen läßt sich bis auf einen Kern von vier extrem häufig wiederholt auszuführenden Grundrechenoperationen reduzieren, der den entscheidenden Rechenzeitanteil ausmacht. Die zentrale Beziehung (14) wird logarithmiert, um in dieser am häufigsten zu berechnenden Formel transzendente Funktionen zu eliminieren.

$$\ln(V^m(\underline{o})) \;=\; \ln\left(\frac{1}{\sqrt{(2\pi)^n \cdot \det(\underline{\underline{\Sigma}}^m)}}\right) - \frac{1}{2}(\underline{o} - \underline{\mu}^m)^T (\underline{\underline{\Sigma}}^m)^{-1} (\underline{o} - \underline{\mu}^m) \qquad (47)$$

Der Term

$$g^m \;=\; \ln\left(\frac{1}{\sqrt{(2\pi)^n \cdot \det(\underline{\underline{\Sigma}}^m)}}\right) \qquad (48)$$

ist zur Laufzeit konstant, und kann daher vorberechnet gespeichert werden. Da die verwendeten KHMM diagonale Kovarianzmatrizen aufweisen, kann die Matrixmultiplikation stark vereinfacht werden. Mit

$$\underline{\underline{\Sigma}}^m \;=\; \begin{bmatrix} \sigma_1^m & 0 & \dots & 0 \\ 0 & \sigma_2^m & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n^m \end{bmatrix} \qquad (49)$$

kann (47) umgeschrieben werden zu

$$\ln(V^m(\underline{o})) \;=\; g^m - \frac{1}{2}\sum_{i=1}^{n} \frac{(o_i^m - \mu_i^m)^2}{\sigma_i^m} \qquad (50)$$

Die Summe auf der rechten Seite wird zur „innersten Schleife“ im Funktionsblock der Merkmalsbewertungen. Das Argument der Summation ist extrem häufig auszurechnen, nämlich für die Anzahl der Merkmalsvektoren mal der Gesamtzahl aller Zustände in allen Phonemmodellen mal der Anzahl Wahrscheinlichkeitsdichtefunktionen pro Zustand mal der Dimensionalität der Merkmalsvektoren. Das sind rund $12 \cdot 10^6$ Summanden für die Bearbeitung einer 4 Sekunden langen Äußerung[1]. Es lohnt daher auch die nur scheinbar unwesentliche Substitution

$$\begin{aligned} \varphi_i^m &= \frac{1}{2\sigma_i^m} \\ \frac{1}{2}\sum_{i=1}^{n}\frac{(o_i^m-\mu_i^m)^2}{\sigma_i^m} &= \sum_{i=1}^{n}\varphi_i^m(o_i^m-\mu_i^m)^2 \end{aligned} \tag{51}$$

σ_i^w und damit auch φ_i^m sind zur Laufzeit konstant, können daher vorberechnet werden, und erlauben eine Division durch eine Multiplikation zu ersetzen, die von den eingesetzten Mikroprozessoren deutlich schneller ausgeführt werden kann[2]. Allein diese Modifikation brachte bezogen auf die Gesamtrechenzeit des Systems eine Beschleunigung um rund 20%.

Der nächste Schritt bei der Merkmalsvektorenbewertung ist die Summation der *M* Wahrscheinlichkeitsdichtefunktionen für einen Zustand. Da es zu Optimierungszwecken und auch aus Gründen der numerischen Darstellbarkeit praktisch ist, mit logarithmierten Bewertungszahlen zu arbeiten, wird aus (13):

$$\ln(b(\underline{o})) = \Psi_{m=1}^{M}\left(\ln(c^m)+\ln(V^m(\underline{o}))\right) \tag{52}$$

mit

$$\Psi_{i=1}^{n}(x_i) = \psi(\ldots\psi(\psi(x_1,x_2),x_3)\ldots,x_n) \tag{53}$$

1. Bei Einsatz der kontextunabhängigen Phonemmodelle
2. Aktuelle omni-purpose-Mikroprozessoren erreichen für Fließkomma-Multiplikationen Durchsatzzeiten von 1 Taktzyklus bei einer Latenz von 3-5 Taktzyklen, wohingegen für Fließkomma-Divisionen mit Durchsatzzeiten von 15-40 Taktzyklen und Latenzen in gleicher Höhe gerechnet werden muß

wobei

$$\psi(x, y) = \ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{\min(x, y) - \max(x, y)}) \qquad (54)$$

$\ln(c^m)$ ist zur Laufzeit konstant und kann daher in g^m eingerechnet werden. Zwar ist die durch $\psi(x, y)$ definierte „logarithmische Addition“ aufwendiger als eine gewöhnliche Addition, durch ihre Verwendung an dieser Stelle werden aber deutlich weniger Berechnungen transzendenter Funktionen benötigt, als bei Verwendung der nicht logarithmierten $V^m(\underline{o})$ und $b(\underline{o})$ erforderlich wären.

Die Zustandsbewertungen werden für die beiden Verfahren PAF und GFSE unterschiedlich berechnet, aber in beiden Fällen ist es vorteilhaft, die Bewertungen in logarithmierter Form zu speichern. Für die GFSE ist der Vorteil in (28) und (30) offensichtlich, für die PAF Methode wird (27) in

$$\upsilon_j^w(t) = \ln\left(\delta_j^w(t)\right) = \max_i(\upsilon_i^w(t-1) + \alpha_{ij}^w) + \ln\left(b_j^w(\underline{o}_t)\right) + \beta_w \qquad (55)$$

umgeformt. Außer den vorberechenbaren Termen $\alpha_{ij}^w = \ln(a_{ij}^w)$ und $\beta_w = \ln(f_w)$ enthält diese Formel nur noch Additionen. Aus diesem Grund, und weil zusätzlich Speicherplatz eingespart werden kann, werden die logarithmierten Zustandsbewertungen in 32-bit Festkommazahlen umgewandelt, so dass die sehr häufig auftretenden Additionen zu elementaren Integer-Operationen für den Prozessor werden, die sehr schnell ausgeführt werden.

Die Zwischenbewertungen $\upsilon_j^w(t)$ werden zusammen mit den $\gamma_j^w(t)$ in einem von zwei Arrays gespeichert, die mit jedem Zeitschritt abwechselnd wiederverwendet werden können, da die Berechnung nur auf den Werten des vorangegangenen Zeitpunkts beruht. Die Maximumbildung in (55) wird durch Kenntnis der erlaubten Zustandsübergänge $a_{ij} > 0$ mit der geringstmöglichen Anzahl Vergleiche realisiert. So ist für strikte links-rechts Modelle die Matrix der Zustandsübergänge sehr dünn besetzt:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & \dots & 0 \\ 0 & a_{22} & a_{23} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & a_{NN} & 1 - a_{NN} \end{bmatrix} \qquad (56)$$

Für diese Übergangsstruktur kann (55) durch

$$\upsilon_j^w(t) \;=\; \max\left(\upsilon_{j-1}^w(t-1) + \alpha_{(j-1)j}^w,\, \upsilon_j^w(t) + \alpha_{jj}^w\right) \;+ \ln\left(b_j^w(\underline{o}_t)\right) + \beta_w \qquad (57)$$

ersetzt werden. Diese Vereinfachung setzt voraus, dass die Übergangsstruktur schon bei der Erstellung des Programms bekannt ist

Die Verknüpfung der Phonemmodelle zu Wörtern ist so realisiert, dass nur Verweise („Pointer“) auf die jeweils verwendeten Modelle abgelegt werden, und auch die Merkmalsvektorenbewertungen nur einmal für jeden Zustand jedes Phonems berechnet werden müssen.

Für die Echtzeitverarbeitung der Sprachsignaldaten benötigt **EFSE** bei Einsatz des Bandpassfilters und der kontextunabhängigen Phonemmodelle eine Rechenleistung nach SPEC[1] von etwa 2.43 CINT95 und 3.11 CFP95. Dies sind Werte, die z. B. von einem Dell Dimension XPS Rechner mit 133MHz Pentium Prozessor erreicht werden[2]. Bei Verwendung der Ganzwortmodelle werden aufgrund der erheblich kleineren Parameteranzahl nur etwa 0.50 CINT95 und 0.64 CFP95 benötigt. Auch auf Rechnern, die zu einer Echtzeitverarbeitung nicht in der Lage sind, kann das Spracherkennersystem bei Verwendung der Äußerungsdetektion durchaus noch arbeiten, es müssen dann allerdings höhere Reaktionszeiten in Kauf genommen werden.

1. „Standard Performance Evaluation Corporation“, eine Organisation, die sich mit der herstellerübergreifenden Messung von praxisbezogenen Rechenleistungsdaten beschäftigt.Ausführliche Erläuterungen zu den Messmethoden und Resultate für viele Systeme sind im InterNet unter http://www.specbench.org/ verfügbar.
2. Die Rechenzeit wird nicht unwesentlich von den architekturabhängigen Optimierungsfähigkeiten des Compilers beeinflusst.

7. Messungen und Ergebnisse

7.1. Verwendete Sprachdaten

Zur Messung von Erkennungs- und Fehlalarmraten wurde der zur Schlüsselworterkennung gedachte Teil des TUBTEL Sprachdatensatzes verwendet. Es handelt sich dabei um Satzäußerungen unterschiedlicher Sprecher, die einen der 5 möglichen Standardsätze

Ich möchte ... bestellen.
Ich würde gern ... kaufen.
Könnten Sie mich vielleicht informieren, ob Sie ... haben?
Gibt es ... bei Ihnen?
Ich brauche ... .

sprechen. Der Platzhalter „ ... “ wird dabei durch eines der 9 möglichen Schlüsselwörter

Rock	**r O k**
Bluse	**b l u: z @**
Kleid	**k l aI d**
Hose	**h o: z @**
Shorts	**S O 6 t s**
Sandalen	**z a n d a: l @ n**
Schuhe	**S u: @**
Strümpfe	**S t r Y m p f @**
Gürtel	**g Y 6 t @ l**

ersetzt und ggf. durch einen unbestimmten Artikel ergänzt. Insgesamt können also 45 unterschiedliche Sätze gesprochen werden. Die Sprecher wurden vom Institut für Fernmeldetechnik der Technischen Universität Berlin durch Verteilung von Aufnahmebögen an Studenten gewonnen, in denen eine Aufwandsentschädigung von 10 DM für das Gespräch mit einem automatischen Dialogsystem im Institut geboten wurde. Durch die Schaffung der Möglichkeit, bis zu 4 Anrufer pro Aufnahmeblatt abzurechnen, sofern mindestens die Hälfte dieser Personen weiblich waren, konnte trotz der deutlichen Überzahl männlicher Studenten am Institut für

Fernmeldetechnik eine relativ gute Gleichverteilung der Geschlechter unter den Anrufern erreicht werden (45% weibliche Anrufer). Anrufe von Nebenstellen des Universitätstelefonnetzes wurden nicht akzeptiert, um zu gewährleisten, dass alle Äußerungen über reale Telefonverbindungen aufgenommen wurden. Der Rauschanteil der Aufnahmen wurde subjektiv den Kategorien „wenig“ (7.7%), „moderat“ (85.9%) und „viel“ (6.5%) zugeordnet. Etwa 8% der Äußerungen weisen Hintergrundgeräusche wie z.B. Musik oder Gespräche anderer Personen auf. Etwa 12% der Sätze wurden von Personen gesprochen, deren Muttersprache nicht Deutsch ist. Die Aufnahmebögen enthielten das Wort „Str**u**mpfe“ statt „Str**ü**mpfe“, dieser Schreibfehler wurde von etwa 65% der Sprecher ignoriert, während die anderen tatsächlich „Strumpfe“ sprachen. Alle anderen Schlüsselwörter wurden von 90% der Sprecher korrekt ausgesprochen.

Im Rahmen der vorliegenden Arbeit wurden Beginn und Ende der Schlüsselwörter innerhalb der Äußerungen manuell markiert, um eine vollständige Auswertung der Ergebnisse und das Training der Ganzwortmodelle zu ermöglichen. Nachdem einige Äußerungen verworfen worden waren, die mehr als eine Schlüsselwortäußerung enthielten, verblieben 443 Sätze mit einer Gesamtlänge von etwa 1 Stunde 10 Minuten. Diese Äußerungen wurden in zwei etwa gleich große Gruppen aufgeteilt, eine „Trainingsmenge“ und eine „Testmenge“, wobei auf eine gleichmäßige Aufteilung der Schlüsselwörter geachtet wurde.

7.2. Sprachmodelle

Zu den Messungen wurden zwei Arten von Modellen verwendet: kontextunabhängige Phonemmodelle und Ganzwortmodelle der Schlüsselwörter. Die kontextunabhängigen Phonemmodelle wurden am PHONDAT Sprachdatensatz trainiert. Dieser enthält abgelesene Geschichten und einzelne Sätze, die von insgesamt 76 verschiedenen Sprechern geäußert wurden. Die Auswahl der Sprecher erfolgte dabei gezielt, um eine Ausgewogenheit hinsichtlich Alter, Geschlecht und regionaler Aussprachevariationen zu erreichen. Insgesamt enthielt die Trainingsmenge für die kontextunabhängigen Phonemmodelle etwa $9 \cdot 10^4$ Phonem Realisationen. Um einzelne, schlecht trainierte Modelle zu vermeiden wurde das Phoneminventar des PHONDAT Sprachdatensatzes um einige nur selten vorkommende Laute reduziert, so z.B. „mittellange“ Vokale und diphthongähnliche Doppelvokale[10]. In Tabelle 1 ist das resultierende Phoneminventar dargestellt. Die Modellierung der kontextunabhängigen Phoneme erfolgte durch kontinuierliche HMM, die aus drei Zuständen pro Phonem mit strikter links-rechts Übergangsstruktur aufgebaut sind. Zur Merkmalsvektorenbewertung werden je Zustand

Tabelle 1: Phoneminventar

Phonem	Beispiel	Transkription
Vokale		
I	S**i**tz	**zIts**
E	Ges**e**tz	**g@zEts**
a	S**a**tz	**zats**
O	Tr**o**tz	**trOts**
U	Sch**u**tz	**SUts**
Y	h**ü**bsch	**hYpS**
9	m**ö**chte	**m9Ct@**
i:	L**ie**d	**li:t**
e:	B**ee**t	**be:t**
E:	sp**ä**t	**SpE:t**
a:	T**a**t	**ta:t**
o:	r**o**t	**ro:t**
u:	Bl**u**t	**blu:t**
y:	s**üß**	**sy:s**
2:	bl**ö**d	**bl2:t**
Schwache Vokale		
@	bitt**e**	**bIt@**
6	bess**er**	**bEs6**
Diphthonge		
aI	**Ei**s	**aIs**
aU	H**au**s	**haUs**
OY	Kr**eu**z	**krOYts**
Liquide		
l	**L**eine	**laIn@**
r	**r**ein	**raIn**

Phonem	Beispiel	Transkription
Frikative		
f	**f**ast	**fast**
v	**w**as	**vas**
s	Tas**s**e	**tas@**
z	Ha**s**e	**ha:z@**
S	was**ch**en	**vaS@n**
C	si**ch**er	**zIC6**
j	**J**ahr	**ja:r**
x	Bu**ch**	**bu:x**
h	**H**and	**hant**
Plosive		
p	**P**ein	**paIn**
b	**B**ein	**baIn**
t	**T**eich	**taIC**
d	**D**eich	**daIC**
k	**K**unst	**kUnst**
g	**G**unst	**gUnst**
Nasale		
m	**m**ein	**maIn**
n	**n**ein	**naIn**
N	Di**ng**	**dIN**
Sonstige		
Q	Verein	**fE6QaIn**
p:	paraling. Phänomen[a]	
sil	Pause / Stille	

a. Das paralinguistische Phänomen repräsentiert Geräusche, die durch Räuspern, Atmen, Schlucken usw. entstehen

10 Gaußdichten mit diagonaler Kovarianzmatrix verwendet, zusammen mit den Übergangswahrscheinlichkeiten ergibt sich daraus für die 43 Phonemmodelle eine Gesamtparameterzahl von etwa $7 \cdot 10^4$. Diese Modelle erreichten am PHONDAT Sprachdatensatz Phonemerkennungsraten von 57.6%[1].

Die Ganzwortmodelle für die 9 Schlüsselwörter wurden an der TUBTEL Trainingsmenge trainiert, die zwischen 20 und 28 Äußerungen jedes Schlüsselworts enthält. Die Zahl der Zustände je Schlüsselwortmodell beträgt drei mal die Anzahl der in der Standardtranskription enthaltenen Phoneme, also z. B. 9 Zustände für das Wort „Rock“ und 24 Zustände für das Wort „Sandalen“. Insgesamt enthalten die 9 Schlüsselwortmodelle 132 Zustände, in der Gesamtanzahl unterschiedlich bewerteter Zustände unterscheiden sich Ganzwort- und Phonemmodelle ($43 \cdot 3 = 129$ Zustände) also kaum. Aufgrund der kleinen Trainingsmenge kommt man für die Ganzwortmodelle mit nur einer Gaußdichte pro Zustand aus, damit ergibt sich für diese Modelle eine Gesamtparameterzahl von etwa $7 \cdot 10^3$.

Zur Schwellenwertberechnung für die GFSE mußten die mittleren Längen und Bewertungen der Phoneme sowie deren Standardabweichungen experimentell ermittelt werden. Zu diesem Zweck wurde mit den Äußerungen aus dem PHONDAT 2 Sprachdatensatz eine kontinuierliche Spracherkennung durchgeführt, wobei die bekannte Abfolge der Phoneme in diesen Äußerungen als Grammatik vorgegeben und alle Phoneme als einzelne Wörter behandelt wurden. Dabei wurden die Bewertungen der Zustände entsprechend (28) bis (30) berechnet. Die Ergebnisse dieser Messung sind in Abbildung 6 dargestellt. Sie lassen keine Korrelation der mittleren Phonemdauer mit der mittleren Bewertung des Phonems erkennen, was darauf hindeutet, dass die Normierung der Bewertungen über die Pfadlänge den gewünschten Effekt erzielt.

7.3. Experimente zur 1-Schlüsselwort-Erkennung

Bei der Analyse von Fehlerkennungen hatte sich herausgestellt, dass in einigen Fällen vermeintlich erkannte Schlüsselwörter unrealistisch langen Äußerungsteilen zugeordnet wurden. Aus diesem Grund wurde das Spracherkennersystem so ergänzt, dass Worthypothesen, deren Längen außerhalb der doppelten Standardabweichung der zu erwartenden Werte liegen, grundsätzlich verworfen werden. Der große positive Einfluß dieser einfachen Maßnahme ist in Tabelle 2 dokumentiert. Die zu erwartenden Längen der Schlüsselwörter lassen sich aufgrund

1. Der angegebene Wert gilt für die „Correctness“, die nur Auslassungen und Verwechslungen als Fehler zählt. Werden auch Einfügungen mitberücksichtigt, ergibt sich eine Rate von 50.1%

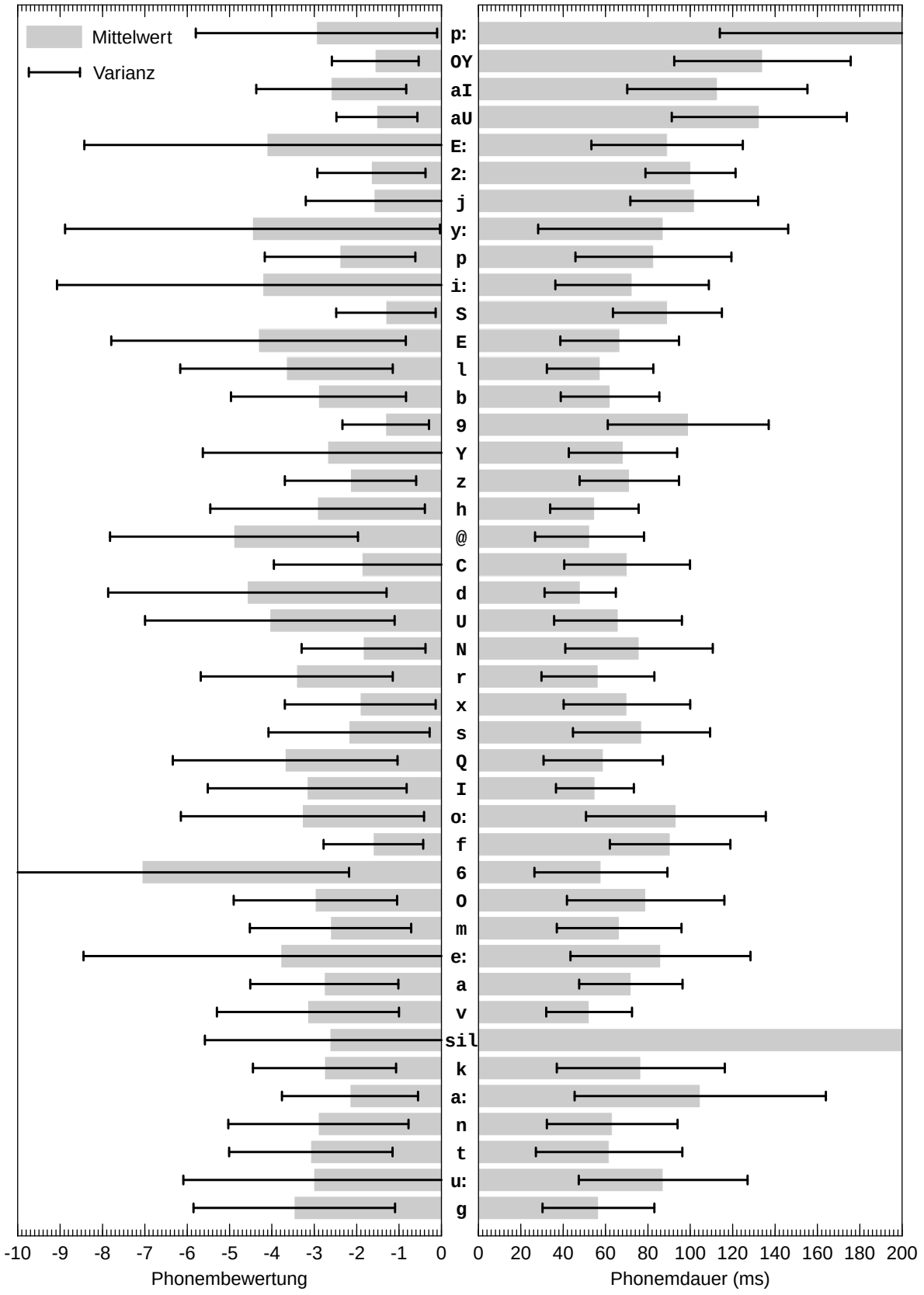

Abbildung 6: Gemessene Bewertungen und Längen gesprochener Phoneme in Phondat II

Tabelle 2: Einfluß der Zeitlimitierung auf die Ergebnisse der 1-Schlüsselwort-Erkennung

	Phonemmodelle				Ganzwortmodelle	
	PAF		GFSE		GFSE	
	Training ER[%]	Test ER[%]	Training ER[%]	Test ER[%]	Training ER[%]	Test ER[%]
Ohne Zeitlimit[a]	75.0	75.3	75.0	75.8	88.2	66.4
Mit Zeitlimit	81.8	79.8	80.5	79.4	96.8	88.8

a. Gesamterkennungsraten für optimierte f_w und K_w

der mittleren Länge der Phoneme abschätzen, wie Abbildung 7 zeigt. Abweichungen vom wahren Wert sind dabei aufgrund der großzügig gewählten Toleranz recht unkritisch. Ein Versuch, die harte Entscheidung durch eine „weiche“ Nachbewertung anhand der Hypothesenlänge zu ersetzen, hatte keinen Erfolg; vermutlich ist die Längenabschätzung der

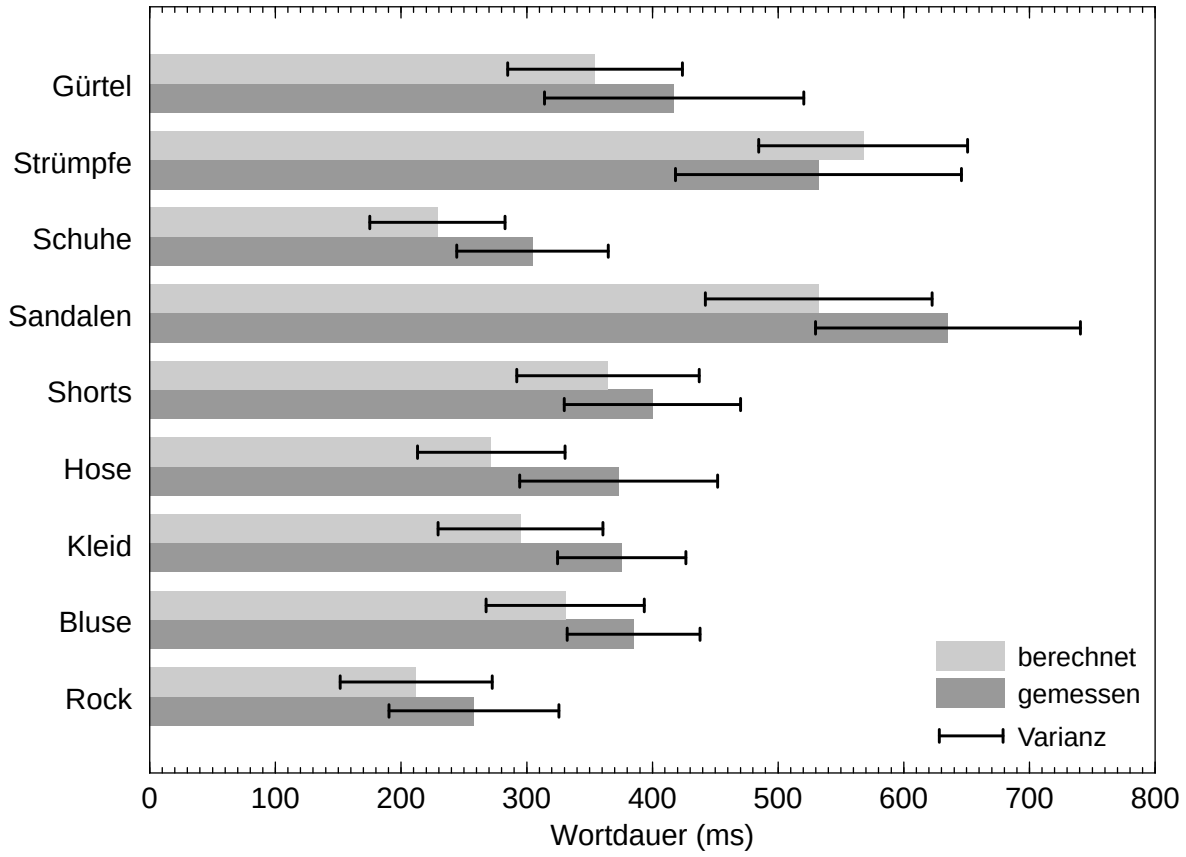

Abbildung 7: Vorberechnete und gemessene mittlere Längen der Schlüsselwörter

Schlüsselwörter anhand der Phoneme hierzu zu ungenau. Auffällig ist, dass die zeitliche Limitierung der Worthypothesen bei den phonembasierten Modellen ganz erheblich weniger zusätzliche Erkennungsrate bringt, als dies bei Ganzwortmodellen der Fall ist - ein Hinweis darauf, dass trotz identischer Zustandszahl und Übergangsstruktur die Fähigkeit der Ganzwortmodelle zur zeitlichen Modellierung der Wörter weniger gut ist. Alle im folgenden genannten Resultate wurden mit zeitlicher Limitierung der Worthypothesen ermittelt.

Wenn keine Möglichkeit besteht, die Konstanten f_w bzw. K_w wortabhängig zu berechnen oder zu bestimmen, z. B. wenn dem System zur Laufzeit neue Wörter hinzugefügt werden, muß mit einem fest vorgewählten Wert für alle Wörter gearbeitet werden. In Tabelle 4 wird gezeigt, welche Resultate sich erzielen lassen, wenn für alle Wörter $f_w = 6.5$ bzw. $K_w = 0.0$ verwendet wird. Die Ganzwortmodelle liefern in diesem Fall durchaus gute Resultate, während die phonembasierten Modelle gegenüber den in Tabelle 3 gezeigten Resultaten für optimierte f_w bzw. K_w stark abfallen. Dies deutet darauf hin, dass sich die Ganzwortmodelle allen Wörtern gleichermaßen gut anpassen, während die Diskrepanzen zwischen den aus kontextunabhängigen

Tabelle 4: Ergebnisse der 1-Schlüsselwort-Erkennung bei Verwendung nicht optimierter Werte

	Phonemmodelle				Ganzwortmodelle	
	PAF		GFSE		GFSE	
Schlüsselwort	Training ER[%]	Test ER[%]	Training ER[%]	Test ER[%]	Training ER[%]	Test ER[%]
Rock	96.2	76.9	96.2	88.5	92.3	88.5
Bluse	62.5	79.1	62.5	79.2	95.8	91.7
Kleid	76.2	63.6	66.7	81.8	95.2	81.8
Hose	88.5	85.2	76.9	81.5	80.8	81.5
Shorts	79.2	88.0	66.7	92.0	95.8	92.0
Sandalen	71.4	69.0	75.0	69.0	96.4	89.7
Schuhe	85.0	63.2	85.0	68.4	100.0	94.7
Strümpfe	30.4	52.2	39.1	47.8	100.0	82.6
Gürtel	50.0	67.9	39.3	53.6	92.9	82.1
Gesamt	70.9	72.2	67.3	73.5	94.1	87.0

Tabelle 3: Ergebnisse der 1-Schlüsselwort-Erkennung bei Verwendung wortangepasster f_w bzw. K_w

	Phonemmodelle						Ganzwortmodelle		
	PAF			GFSE			GFSE		
Schlüsselwort	f_w	Training ER[%]	Test ER[%]	K_w	Training ER[%]	Test ER[%]	K_w	Training ER[%]	Test ER[%]
Rock	3.1875	92.3	76.9	.75	92.3	76.9	-0.6	92.3	88.5
Bluse	6.625	70.8	79.2	-0.281	87.5	91.7	0.0	95.8	91.7
Kleid	3.5	76.2	54.5	0.375	81.0	77.3	0.12	100.0	81.8
Hose	4.75	80.8	88.9	0.5	69.2	77.8	-0.62	92.3	92.6
Shorts	8.5	91.7	96.0	0.0	83.3	96.0	0.6	95.8	84.0
Sandalen	9.75	96.4	93.1	0.0	85.7	79.3	-0.5	100.0	93.1
Schuhe	6.0	85.0	84.2	0.125	80.0	63.2	-0.9	100.0	100.0
Strümpfe	11.0	60.9	69.6	-0.5	73.9	78.3	0.09	100.0	78.3
Gürtel	6.0	78.6	71.4	0.0	71.4	71.4	-0.3	96.4	89.3
Gesamt		81.8	79.8		80.5	79.4		96.8	88.8

Phonemen zusammengesetzten Modellen unbedingt durch wortabhängige Faktoren kompensiert werden müssen.

Wie in Abschnitt 5.1 und 5.2 erläutert wurde, sind die f_w bzw. K_w stark voneinander abhängig, und müssen daher gemeinsam iterativ optimiert werden, wenn eine maximale Erkennungsrate erreicht werden soll. Die f_w für die PAF Methode wurden zunächst alle auf den Wert 0.0 gesetzt, und eine Erkennung an der Trainingsmenge durchgeführt. Anschließend wurde f_w für jedes Wort einzeln um eine Schrittgröße s=20.0 erhöht, und erneut eine Erkennung durchgeführt. War das Ergebnis dieser Erkennung nicht besser als das vorhergehende, so wurde f_w wieder auf den letzten Wert zurückgesetzt, und stattdessen um s vermindert. Wenn sich das Ergebnis verbessert hatte, wurde f_w ein weiteres mal um s erhöht oder vermindert, und der Versuch wurde wiederholt. Sobald weder durch Erhöhung, noch durch Verminderung von f_w um s eine Steigerung der Erkennungsrate erreicht werden konnte, wurde s halbiert, und die Variation von f_w fortgeführt. Die Optimierung wurde beendet, wenn $s < 0.01$ wurde, denn es hatte sich gezeigt, dass kleinere Variationen keinen Einfluß mehr auf die Erkennungsrate haben. Auf diese Weise wurden die in Tabelle 3 genannten optimalen Werte der f_w für die Trainingsmenge ermittelt. Analog wurden auch die optimalen K_w für die Trainingsmenge ermittelt, nur dass hierbei die anfängliche Schrittweite auf 1.0 und die minimale Schrittweite auf 0.005 festgesetzt wurden.

Interessant ist, dass sich für optimierte f_w und K_w der Unterschied in den Erkennungsraten von Trainings- und Testmenge bei Verwendung der phonembasierten Modelle gegenüber den unoptimierten Werten deutlich verkleinert, während er sich bei den Ganzwortmodellen noch vergrößert. Auch ist bemerkenswert, dass sich die Erkennungsraten der einzelnen Wörter für die PAF und GFSE Methode teilweise stark (bis zu 23%) voneinander unterscheiden, während die Gesamterkennungsraten recht dicht beieinander liegen. Die großen Unterschiede zwischen den Erkennungsraten der einzelnen Wörter weisen auch darauf hin, wie stark die Gesamterkennungsrate eines Systems vom Vokabular abhängig ist. Derselbe Schlüsselworterkenner erreicht für lange und unähnliche Schlüsselwörter erheblich bessere Resultate als für kurze oder ähnlich lautende.

7.4. Experimente zur n-Schlüsselwort-Erkennung

Da die Erkennungsrate eines n-Schlüsselwort-Erkenners abhängig ist von der tolerierten Fehlalarmrate, wurden für eine Reihe von Fehlalarmraten zwischen Null und Zehn Fehlalarme pro Schlüsselwort und Stunde passende f_w und G_w ermittelt. Da im allgemeinen nur die durchschnittliche Fehlalarmrate aller Schlüsselwörter angegeben wird, könnte man die Werte so wählen, dass bei Einhaltung einer vorgegebenen durchschnittlichen Fehlalarmrate die Erkennungsrate maximiert wird. Dieser Ansatz erzeugt allerdings starke Abhängigkeiten der f_w und G_w untereinander, und würde dadurch die Wahl dieser Werte zu einer aufwendigen Optimierung und Anpassung an die Sprachdaten machen. Aus diesem Grund wird hier ein anderer Ansatz verfolgt: Die Werte werden so eingestellt, dass die maximale Fehlalarmrate für jedes Schlüsselwort individuell eingehalten wird[1]. Dazu wurden die G_w bei Null beginnend solange

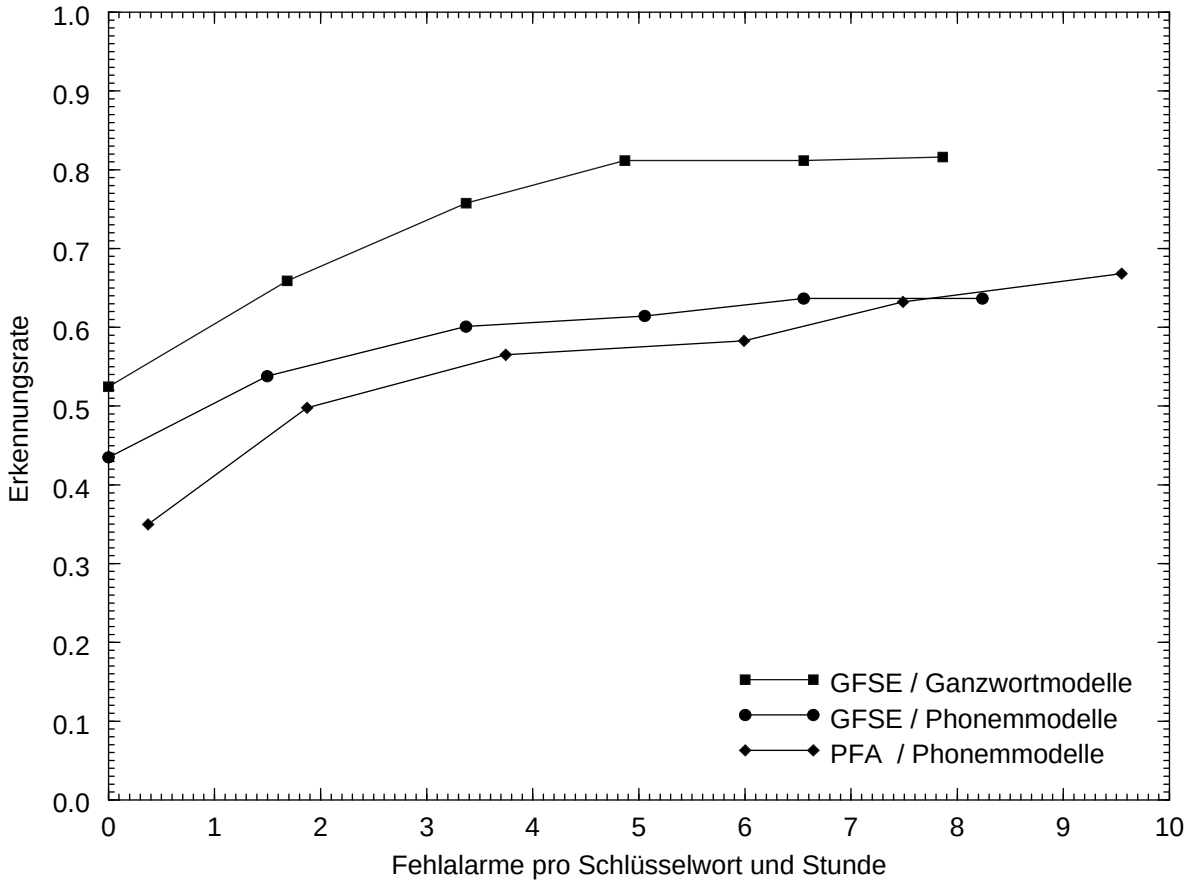

Abbildung 8: Ergebnisse der n-Schlüsselworterkennung für die getesteten Verfahren

1. Die f_w sind anders als die G_w nicht völlig unabhängig voneinander. Die Abhängigkeit tritt jedoch nur bei ähnlichen Schlüsselwörtern oder hohen Fehlalarmraten zutage, wenn Schlüsselwörter untereinander in zeitlich überlappenden Äußerungsabschnitten konkurrieren. Die Fehlalarmrate mancher Schlüsselwörter weicht dadurch für die PFA Methode geringfügig vom vorgewählten Wert ab.

um abnehmende Beträge vermindert, wie dies die maximale Fehlalarmrate zuließ. Analog wurden die f_w bei Null beginnend solange um abnehmende Beträge erhöht, wie dies ohne Überschreitung der maximalen Fehlalarmrate möglich war. Je kleiner die maximale Fehlalarmrate ist, desto geringer ist die statistische Stabilität der so erzielten Ergebnisse, da die Bewertung von immer weniger Äußerungen darüber entscheidet, wie hoch bzw. niedrig die f_w und G_w gewählt werden dürfen. Weil die Testmenge nur etwa 35 Minuten Sprache enthält, bedeutet ein einziger Fehlalarm hochgerechnet eine Fehlalarmrate von 1.69 Fehlalarme pro Stunde für dieses Schlüsselwort. Die Aussagekraft der Ergebnisse für kleine Fehlalarmraten ist also gering, sie könnte durch eine Vergrößerung der Testmenge verbessert werden.

Wenn die Möglichkeit zu einer iterativen Bestimmung der f_w und G_w nicht gegeben ist, z. B. weil ein unbekanntes Wort zur Laufzeit dem System hinzugefügt werden soll, dann müssen entweder wortunabhängig fest voreingestellte Werte verwendet werden, die dann ggf. vom Anwender nachjustiert werden können, oder man versucht durch Kenntnis der phonetischen Transkription geeignete Werte zu berechnen, wie dies in Abschnitt 5.2 erläutert wurde. Gemäß

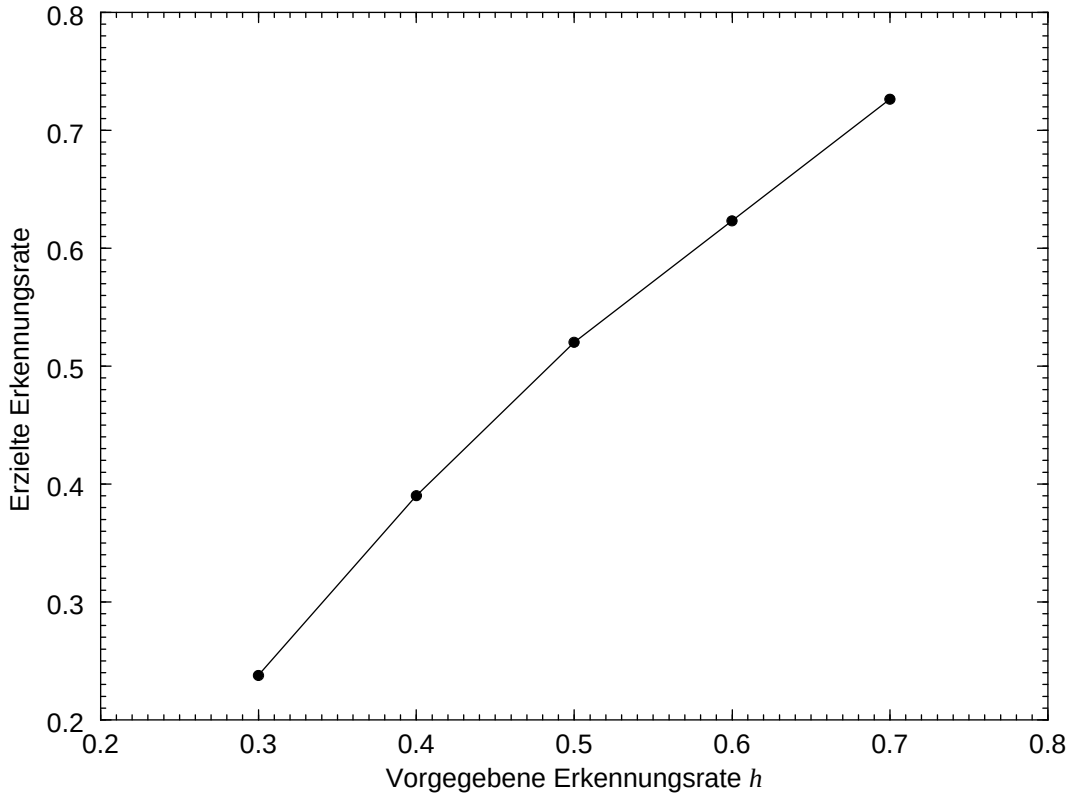

Abbildung 9: Vergleich der vorgegebenen und tatsächlich erzielten Gesamterkennungsraten unter Verwendung der phonembasierten GFSE

(45) wurden unter Verwendung der in Abbildung 6 genannten Werte für h = 0.3, 0.4, 0.5, 0.6 und 0.7 Schwellenwerte für alle Schlüsselwörter berechnet. Abbildung 9 zeigt eine sehr gute Übereinstimmung der vorgegebenen mit den tatsächlich erzielten Erkennungsraten. Diese erwünschte Übereinstimmung gilt aber nur für das Gesamterkennungsergebnis, wie die Resultate für die einzelnen Schlüsselwörter in Tabelle 5 zeigen. Die Fehlalarmraten sind ebenfalls sehr uneinheitlich und - wie Abbildung 10 zeigt - insgesamt deutlich höher als sie bei optimal angepaßten G_w wären. Selbst wortunabhängige, konstante Schwellenwerte, die durch Mittel-

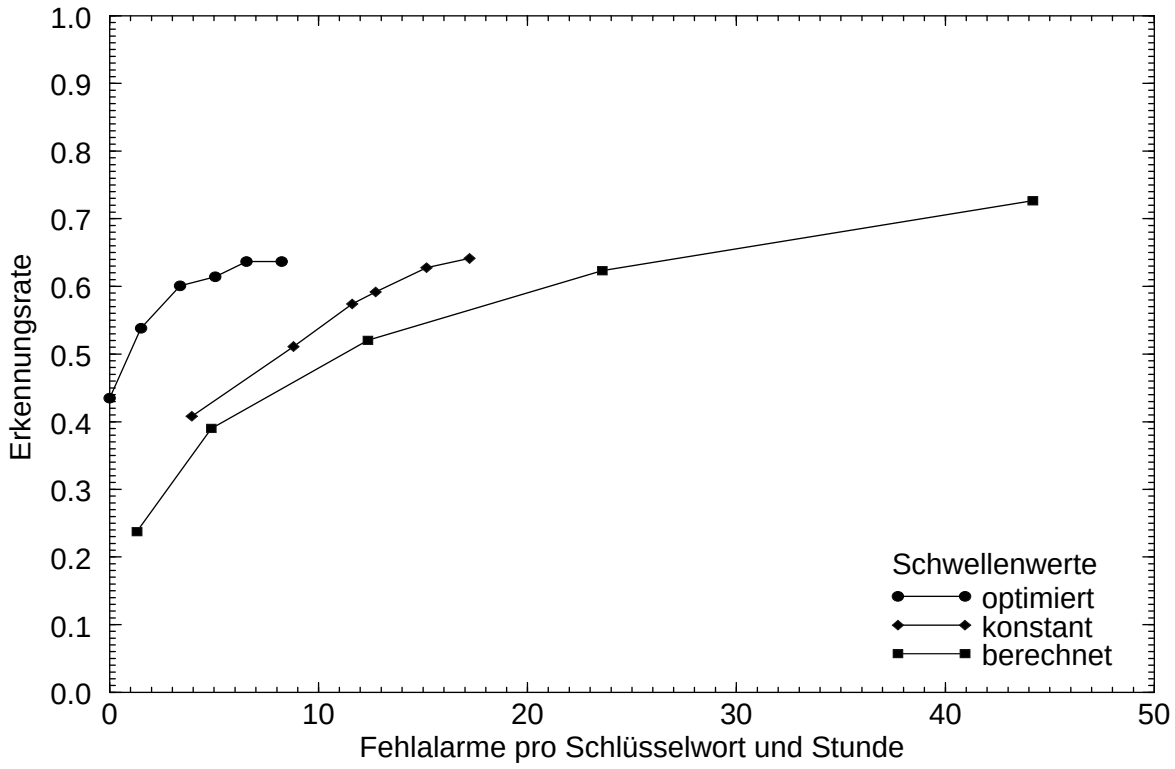

Abbildung 10: Vergleich der Erkennungs- und Fehlalarmraten der phonembasierten GFSE bei optimal angepassten, wortunabhängig konstanten und berechneten Schwellenwerten

wertbildung aus den optimal angepassten Schwellenwerten errechnet wurden, erzielten ein besseres Verhältnis von Erkennungs- und Fehlalarmrate. Die offenbar unzureichende Anpassung der berechneten Schwellenwerte an bestimmte Schlüsselwörter wird ersichtlich, wenn man die vorberechneten Mittelwerte und Standardabweichungen der Schlüsselwortbewertungen mit den gemessenen vergleicht. Abbildung 11 zeigt, dass die vorberechneten Werte genau für jene Schlüsselwörter (Strümpfe, Gürtel) stark von den Messwerten abweichen, die gemäß

Tabelle 5: Resultate der phonembasierten GFSE bei vorgegebenen Erkennungsraten

	h=0.3		h=0.4		h=0.5		h=0.6		h=0.7	
Schlüsselwort	ER[%]	$\frac{FA}{KW \cdot h}$	ER[%]	$\frac{FA}{KW \cdot h}$	ER[%]	$\frac{FA}{KW \cdot h}$	ER[%]	$\frac{FA}{KW \cdot h}$	ER[%]	$\frac{FA}{KW \cdot h}$
Rock	26.9	0.0	50.0	3.37	61.5	13.48	76.9	38.75	80.8	77.51
Bluse	37.5	0.0	45.8	0.0	54.2	0.0	75.0	3.37	87.5	6.74
Kleid	18.2	0.0	45.5	5.06	50.0	13.48	54.5	21.91	63.6	40.44
Hose	33.3	1.69	40.7	8.43	59.3	20.22	74.1	35.39	85.2	74.14
Shorts	36.0	0.0	40.0	0.0	60.0	1.69	80.0	1.69	96.0	3.37
Sandalen	17.2	0.0	34.5	0.0	51.7	0.0	55.2	0.0	72.4	0.0
Schuhe	5.3	0.0	10.5	0.0	47.4	1.69	57.9	1.69	63.2	8.43
Strümpfe	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.4	0.0
Gürtel	32.1	10.11	71.4	26.96	75.0	60.66	75.0	109.53	89.3	187.04
Gesamt	23.8	1.31	39.0	4.87	52.0	12.36	62.3	23.59	72.6	44.19

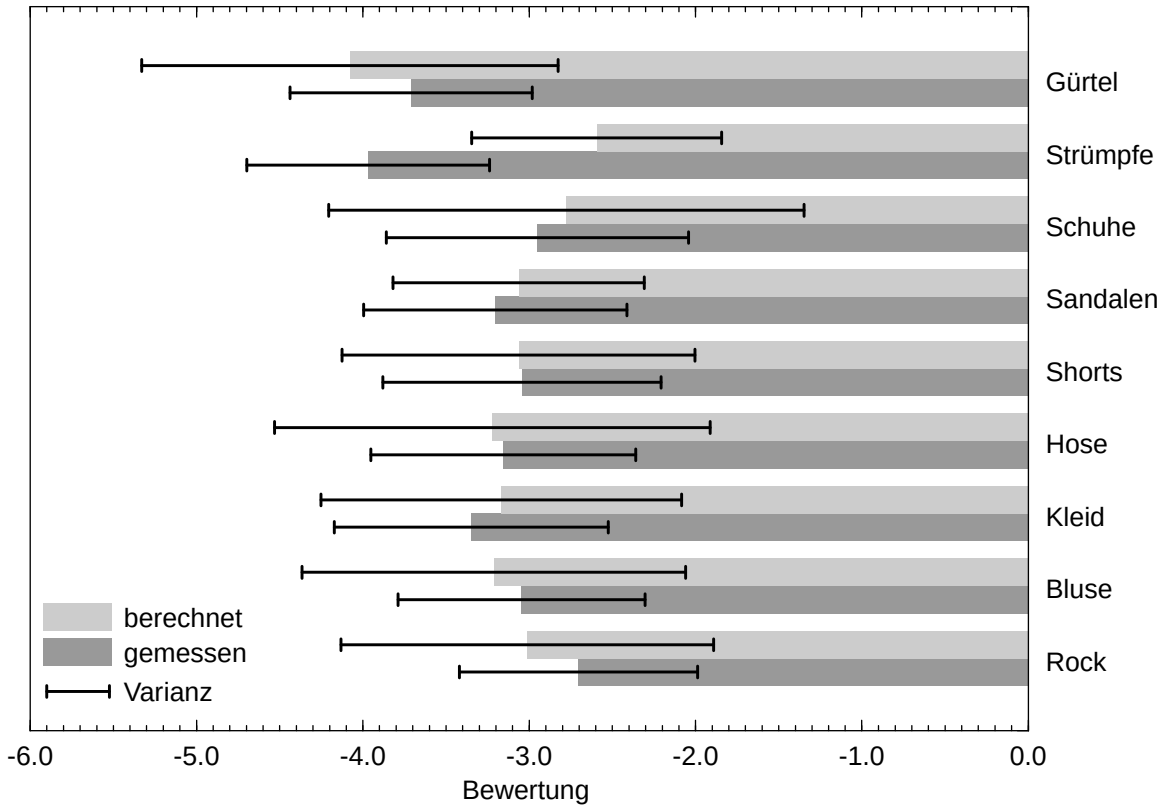

Abbildung 11: Vorberechnete und gemessene Bewertungen der Schlüsselwörter

Tabelle 5 nicht die gewünschten Erkennungsraten erzielten. Ein wichtiger Grund für die Diskrepanzen ist, dass die für (41) gemachte Annahme der Unabhängigkeit aufeinanderfolgender Phonembewertungen eine zu weitgehende Vereinfachung ist. Koartikulationseffekte an den Phonemgrenzen sind nicht vernachlässigbar. Eine weitere Ursache liegt in der unzureichend genauen Näherung der Wahrscheinlichkeitsverteilungen der Phonembewertungen durch Gauß-Funktionen. Abbildung 12 zeigt einen Vergleich zwischen den am PHONDAT II Sprachdatensatz gemessenen Verteilungen ausgewählter Phoneme und ihren Approximationen durch Gauß-Funktionen.

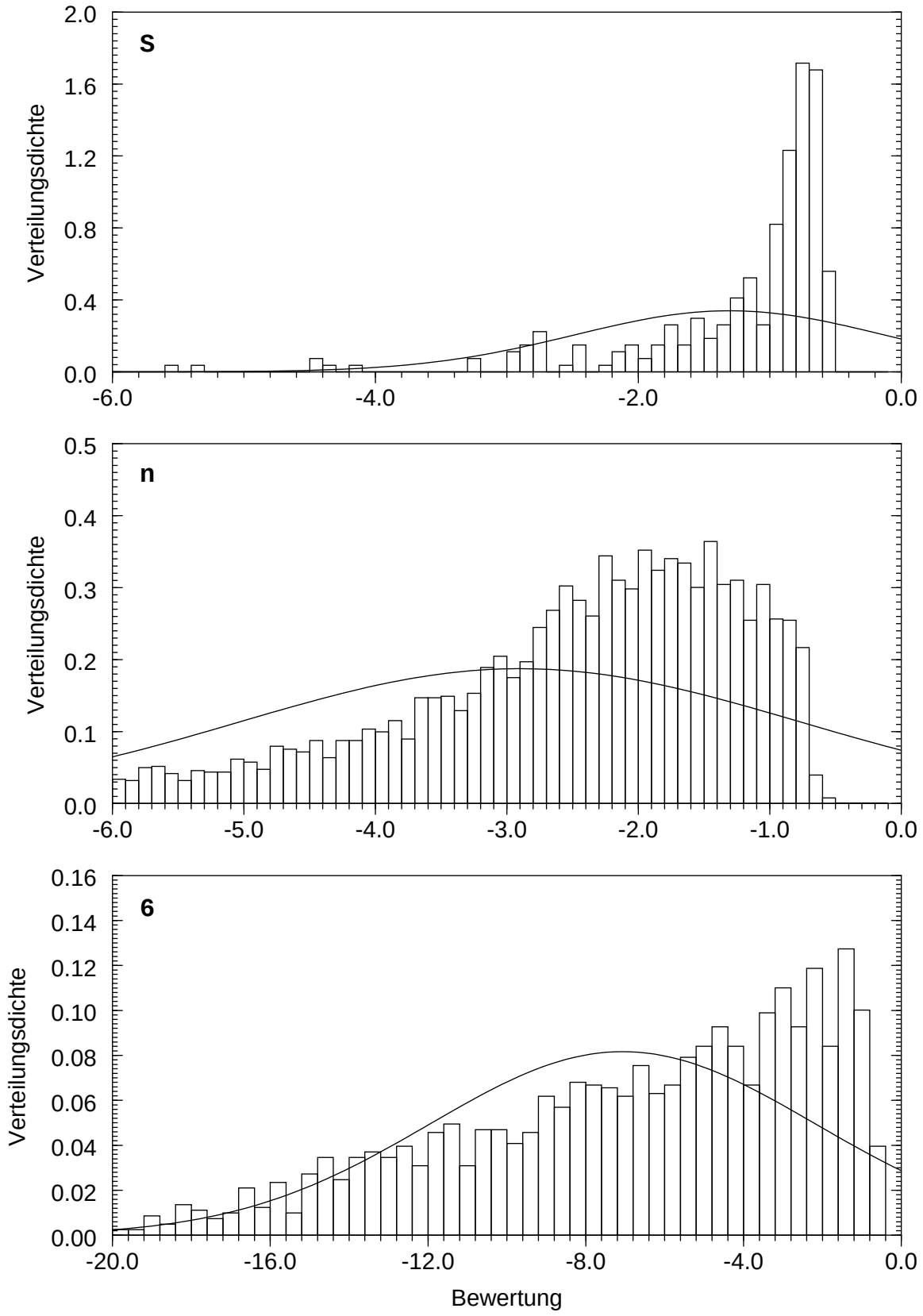

Abbildung 12: Verteilungsdichten der Bewertungen für die Phoneme /**S**/, /**n**/ und /**6**/

8. Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde ein Spracherkennungssystem realisiert, das sowohl phonembasierte als auch wortbasierte Modelle zur sprecherunabhängigen Schlüsselworterkennung im Kontext fließender Sprache verwenden kann. Das System erlaubt dabei die Wahl zwischen zwei grundlegend verschiedenen Verfahren: Entweder kann die Bewertung von Äußerungen durch Schlüsselwortmodelle mit gewählten Schwellenwerten verglichen werden, wobei eine Schwellenwertüberschreitung die Erkennung eines Schlüsselwortes signalisiert, oder es werden beliebige Phonemfolgen als Füllmodelle verwendet, die mit den Schlüsselwortmodellen konkurrieren. Der Schlüsselworterkenner kann sowohl zur 1-Schlüsselwort-Erkennung, bei der vorausgesetzt wird, dass sich in jeder Äußerung exakt ein Schlüsselwort befindet, als auch zur n-Schlüsselwort-Erkennung verwendet werden, bei der sich eine beliebige Anzahl Schlüsselwörter in jeder Äußerung befinden kann. Durch eine effiziente Implementation wurde die Fähigkeit zur Echtzeitverarbeitung auf verfügbaren Arbeitsplatzrechnern erreicht.

Die Leistungen der verschiedenen Verfahren zur Schlüsselworterkennung wurden anhand realer Telefonsprache aus dem TUBTEL Sprachdatensatz untersucht. Die beiden phonembasierten Schlüsselworterkennungverfahren erwiesen sich als etwa ebenbürtig. Die Erkennungsrate für die 1-Schlüsselwort-Erkennung konnte für die phonembasierten Modelle von 72.2% (freie Phonemfolgen als Füllmodelle) bzw. 73.5% (Schwellenwertverfahren) bei nicht wortangepassten Parametern durch Optimierung derselben an der Trainingsmenge bis auf 79.8% bzw. 79.4% gesteigert werden. Etwa 4% Erkennungsrate konnten dabei durch die zeitliche Limitierung der Schlüsselworthypothesen auf den Bereich der doppelten Standardabweichung um die auf Basis der gemessenen Phonemdauern vorberechneten Werte gewonnen werden.

Bei der n-Schlüsselwort-Erkennung lagen die Ergebnisse der Füllmodellmethode bis zu einer für jedes Schlüsselwort individuell eingehaltenen Fehlalarmrate von etwa 7.5 $\frac{\mathrm{FA}}{\mathrm{KW} \cdot \mathrm{h}}$ unter denen des Schwellenwertverfahrens, bei höheren Fehlalarmraten gab es leichte Vorteile für das Füllmodellverfahren. Bei einer mittleren Fehlalarmrate von etwa 5 $\frac{\mathrm{FA}}{\mathrm{KW} \cdot \mathrm{h}}$ wurden Erkennungsraten von etwa 58% (Füllmodellmethode) bzw. 61% (Schwellenwertmethode) erreicht.

Die Berechnung von Schwellenwerten für die n-Schlüsselworterkennung hinsichtlich einer vorgewählten Gesamterkennungsrate war möglich, jedoch wiesen die damit erzielten Erkennungsergebnisse für einzelne Schlüsselwörter sehr große Abweichungen vom Gesamtergebnis auf. Das Verhältnis von Erkennungs- und Fehlalarmraten war bei Verwendung der vorberechneten Schwellenwerte deutlich schlechter als bei angepassten und immer noch erkennbar schlechter als bei wortunabhängig konstant gewählten Werten.

Die Erkennungsleistung der wortbasierten Modelle profitierte besonders stark von der zeitlichen Limitierung der Hypothesendauern, so verbesserte sich die Erkennungsrate im Falle der 1-Schlüsselwort-Erkennung an der Testmenge um 22% bei Verwendung optimierter Schwellenwerte. Insgesamt schnitten die wortbasierten Modelle sehr gut ab, so erzielten sie bei der 1-Schlüsselwort-Erkennung bereits mit nicht optimierten, wortunabhängigen Schwellenwerten eine Erkennungsrate von 87%, die mit optimierten Schwellenwerten auf 88.8% anstieg. Auch bei der n-Schlüsselwort-Erkennung übertrafen die Erkennungsleistungen der wortbasierten klar die der phonembasierten Modelle. So wurde für eine mittlere Fehlalarmrate von etwa $5\ \frac{\mathrm{FA}}{\mathrm{KW}\cdot\mathrm{h}}$ eine Erkennungsrate von 81% erreicht. Beim Vergleich der wort- und phonembasierten Modelle ist jedoch zu berücksichtigen, dass erstere an Datenmaterial trainiert wurden, dass hinsichtlich der Störgeräusch- und Aufnahmesituation sowie des Schlüsselwortkontextes der Testmenge gleichgeartet war, da es demselben Sprachdatensatz entstammte, während die Phonemmodelle an ungestörten Laboraufnahmen anderer Texte trainiert worden waren.

Die Experimente haben insgesamt gezeigt, dass die Verwendung phonembasierter Modelle zur sprecherunabhängigen Schlüsselworterkennung möglich ist, und gegenüber wortbasierten Modellen große Vorteile hinsichtlich der Flexibilität und Anpassungsfähigkeit an wechselnde Rahmenbedingungen aufweist. Allerdings geht die hohe Flexibilität der phonembasierten Modelle noch mit einer niedrigeren Erkennungsleistung gegenüber den wortbasierten Modellen einher.

Es zeichnen sich klare Ansätze für weiterführende Forschungsthemen zur Schlüsselworterkennung ab: So lässt die Verwendung kontextabhängiger Phonemmodelle auf eine Verbesserung der Erkennungsleistung hoffen, ebenso könnte die durch Optimierung der Parameter anhand von Trainingsäußerungen erzielbare Steigerung der Erkennungsraten auch für den Fall der nicht-Verfügbarkeit solcher Trainingsäußerungen realisiert werden, wenn es gelänge, eine

zuverlässige Methode zur Berechnung der Parameter auf Basis der Messergebnisse an einzelnen Phonemen zu finden.

Mein herzlicher Dank gilt
Herrn Prof. Dr. D. Wolf für die Aufnahme in die Arbeitsgruppe Digitale Signalverarbeitung
Herrn PD Dr. H. Reininger für die Anregung zu dieser Arbeit, seinen fachlichen Rat und die gründliche Durchsicht des Manuskripts
Herrn Prof. Dr. K. Fellbaum für die Überlassung des TUBTEL Datensatzes
den Diplom Physikern Klaus Kasper, Gerhard Schmidt, Martin Schlothauer und Harald Wüst für konstruktive Diskussionen und die Erstellung der Sprachmodelle.

9. Literaturverzeichnis

[1] S. K. Mitra, J. F. Kaiser: „Handbook for Digital Signal Processing“, Wiley-Interscience Publication, 1993, pp. 1157

[2] A. V. Oppenheim, R. W. Schafer: „Digital Signal Processing“, Prentice-Hall, 1975

[3] R. E. Blahut: „Fast Algorithms for Digital Signal Processing“, Addison-Wesley, 1985

[4] L. R. Rabiner, R. W. Schafer: „Digital Processing of Speech Signals“, Prentice-Hall, 1978

[5] L. R. Rabiner: „A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition“, Proc. IEEE, vol. 77, 1989, pp. 257-285

[6] G. D. Forney: „The Viterbi Algorithm“, Proc. IEEE, vol. 61, 1973, pp. 268-278

[7] L. E. Baum: „An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process“, Inequalities 3, 1972, pp. 1-8

[8] J. Junkawitsch, L. Neubauer, H. Höge, G. Ruske: „A new keyword spotting algorithm with pre-calculated optimal thresholds“

[9] S. J. Young: „HTK: Hidden Markov Model Toolkit Reference Manual - Version 1.4“, Speech Group, Cambridge University Engineering Dept., Cambridge, England, 1992

[10] G. Schmidt: „Vokabularunabhängige Phonemmodelle auf der Grundlage kontinuierlicher Hidden-Markov-Modelle“, Institut für Angewandte Physik, Johann Wolfgang Goethe-Universität Frankfurt am Main, Juni 1996

[11] B. Stroustrup: „The C++ Programming Language“, Addison-Wesley, 1992

A. Beschreibung der Spracherkennungprogramme

Dieser Abschnitt soll dem an der Verwendung oder Erweiterung des Spracherkennersystems Interessierten Kenntnisse zu Aufbau und Bedienung der Programme verschaffen. Das System wurde mit Ausnahme der graphischen Benutzeroberfläche, die mit Tcl/Tk realisiert ist, objektorientiert in C++ programmiert. Die folgenden Erläuterungen nehmen Bezug auf die Klassen- und Methodennamen[1] des Quelltextes (z.B. **`Grammar::compile`**). Die Deklaration einer Klasse und ihrer Methoden ist in der Regel in einer Datei zu finden, deren Name dem der Klasse entspricht (z. B. „Grammar.h“), ausgenommen manche kleine oder nur lokal verwendete Klassen, deren Deklaration und Implementation gemeinsam mit einer übergeordneten Klasse erfolgte (z.B. ist **`Atom`** zusammen mit **`Grammar`** in „Grammar.h“ deklariert). Analog befinden sich die Methodenkörper einer Klasse in der Regel in einer Datei, deren Name dem der Klasse entspricht (z.B. „Grammar.cc“), Ausnahmen hiervon sind die Methoden einiger sehr umfangreicher Klassen, deren Methoden über mehrere Quelltextdateien verteilt wurden (z.B. befinden sich die Methoden der Klasse **`Recognizer`** in mehreren Dateien, deren Namen durch Anhängen eines Zusatzes an den Klassennamen gebildet wurden). Die Quelltexte der Hauptprogramme befinden sich in den Dateien „csr1.cc“ und „etb.cc“, die nicht Klassen zugeordneten Unterprogramme derselben in Dateien des Namens „csr1_*.cc“[2] und „etb_*.cc“.

Die folgende Tabelle nennt in chronologischer Reihenfolge die wichtigsten Methoden und deren zugeordnete Teilschritte des Programmablaufs. Der Klasse **`Recognizer`** fällt dabei eine zentrale Bedeutung zu, da sie die Schnittstelle zwischen modelltypabhängigen und -unabhängigen Teilen des Programms **`EFSE`** definiert. **`Recognizer`** ist eine abstrakte (nicht instanzierbare) Basisklasse, deren rein virtuelle Methoden von allen abgeleiteten Modellklas-

1. Terminologisch entsprechen den Klassen und Methoden der objektorientierten Programmierung (stark vereinfacht) die Datenstrukturen und Funktionen der prozeduralen Programmierung. Ein voll qualifizierter Methodenname wie „**`Grammar::compile`**“ baut sich aus dem Namen der Klasse („**`Grammar`**“), dem „scope resolution operator“ („**`::`**“) und dem eigentlichen Methodennamen („**`compile`**“) auf. Der Anwendung dieser Methode entspricht, wieder stark vereinfachend gesagt, in der prozeduralen Programmierung der Aufruf einer Funktion „**`compile`**“ mit einer Datenstruktur vom Typ „**`Grammar`**“ als erstem Parameter. Die „Instanz einer Klasse“ entspricht etwa der „Variable eines Typs“ in der prozeduralen Programmierung. Für eine detaillierte Beschreibung der C++ Syntax und eine Einführung in objektorientierte Programmierkonzepte siehe [11]
2. Der „*“ steht in Dateinamen als Platzhalter für beliebige Zeichenketten.

sen (**`Schlauer`**, **`KF2`** usw., im folgenden ersetzt durch ***`Modellklasse`***) implementiert werden müssen.

Aktion	EFSE	ISEO
Initialisierung der graphischen Benutzeroberfläche		Das **`ISEO`** Hauptprogramm startet die Funktion **`etb_relais`** als eigenen Prozess, und beginnt anschließend mit der Abarbeitung des Tcl-Skripts „etb.tcl“, das eine graphische Benutzeroberfläche erzeugt und über spezielle Befehle mit dem Relais-Prozess kommuniziert.
Start eines separaten Prozesses zur Sprachaufnahme		Die Funktion **`etb_relais`** startet die Funktion **`record_piper`** als separaten Prozess.
Start externer Spracherkennerprozesse		Das Tcl-Skript veranlasst die **`etb_relais`** Funktion, gemäß den eingestellten Optionen externe Spracherkennerprozesse zu starten (i.d.R. **EFSE**).
Laden eines Sprachmodells	Der gewünschte Modelltyp wird durch Aufruf der statischen Methode **`Recognizer::get_modeltype`** ermittelt. Eine Instanz der entsprechenden Modellklasse wird konstruiert, dabei dient die Methode **`Recognizer::init`** zum Lesen einer existierenden **EFSE**-Sprachmodelldatei oder die Methode ***`Modellklasse`*`::init`** zur Erstellung des Sprachmodells aus textuellen Quelldateien.	
Übersetzen der Grammatik	Durch **`Grammar::compile`** werden kompakte Datenstrukturen mit den Wortmodell- und -übergangsinformationen erzeugt.	
Gewinnung des digitalisierten Sprachsignals	Bei direkter Sprachsignalaufnahme wird in der Funktion **`csr1_audio`**...	In der Funktion **`record_piper`** wird...
	... eine Instanz der Klasse **`AudioSystem`** erzeugt, die eine systemabhängige Ableitung der Klasse **`AudioInterface`** ist. Duch Aufruf der Methode **`AudioSystem::record_fillbuf`** wird ein Block Sprachsignaldaten vom Betriebssystem angefordert.	

Aktion	EFSE	ISEO
Laden von Sprachsignaldaten	Sollen existierende Sprachsignaldateien geladen werden, geschieht dies durch Aufruf der Methode **`Preprocessor::filter`** aus der Methode **`Recognizer::recog_file`** heraus. **`Preprocessor::filter`** filtert das Signal auch gemäß den eingestellten Optionen und detektiert ggf. Äußerungen.	
Filterung	Bei direkter Sprachaufnahme werden die Sprachsignalblöcke in der Funktion **`csr1_audio`** gemäß den eingestellten Optionen bandpassgefiltert oder der normierten Mikrophoncharakteristik angepasst.	Die Sprachsignalblöcke werden in der Funktion **`record_piper`** gemäß den eingestellten Optionen bandpassgefiltert oder der normierten Mikrophoncharakteristik angepasst.
Interprozesskommunikation		Die Sprachsignaldaten werden von der Funktion **`record_piper`** an den Relais-Prozess übergeben
Äußerungsdetektion	Bei direkter Sprachaufnahme wird das Sprachsignal...	Das Sprachsignal wird...
	...der Methode **`UtteranceDetector::add_sigbuf`** übergeben, die Anfang und Ende von Äußerungen feststellt und diese in einem internen Puffer speichert.	
Interprozesskommunikation	Falls **EFSE** als externer Spracherkennerprozess läuft, erhält die Funktion **`csr1_server`** fertig gefilterte Äußerungen oder, bei zeitsynchronem Betrieb, Äußerungsteile.	**`etb_relais`** sendet komplette Äußerungen oder, bei zeitsynchronem Betrieb, Äußerungsteile an die externen Erkennerprozesse. Der zeitsynchrone Betrieb kann durch Definition des Preprozessormacros **`TIMESYNCHRONUS`** in der Datei „etb_relais.cc“ vor der Übersetzung aktiviert werden. Er funktioniert jedoch nur mit Modelltypen, die geeignete Merkmalsvektoren verwenden (siehe Abschnitt 6.2).

Aktion	EFSE	ISEO
Modellabhängige Merkmalsvektoren-berechnung.	Die rein virtuelle Methode **`Recognizer::recog_sigbuf`** erhält eine Instanz der Klasse **`SignalBuffer`** übergeben, die das Sprachsignal enthält. Daraus berechnet sie geeignete Merkmalsvektoren für den gewählten Modelltyp. Für kontinuierliche HMM werden hierzu z.B. die Methoden **`Observable::compute_ceps`** und **`compute_deltas`** verwendet. Für jeden berechneten Merkmalsvektor wird eine Instanz der Klasse **`Observable`** erzeugt. Ein Array dieser Instanzen wird der Methode ***`Modellklasse`*`::recog_obs`** übergeben. Beim zeitsynchronen Betrieb wird die virtuelle Methode **`Recognizer::recog_ts`** aufgerufen, die unter Verwendung interner Zwischenspeicher Merkmalsvektorenfolgen für Teiläußerungen berechnet, und diese der Methode ***`Modellklasse`*`::recog_ts`** übergibt.	
Modellinitialisierung	Vor der ersten Erkennung nach dem Programmstart wird **`Recognizer::coldstart`** aufgerufen. Zwischen den Erkennungen einzelner Äußerungen wird der Status des Systems mit der Methode **`Recognizer::reset`** zurückgesetzt.	
Verarbeitung eines Merkmalsvektors (allgemein)	Die Verarbeitung eines Merkmalsvektors beginnt mit der Schaffung einer **`TimeStep`** Instanz für den aktuellen Zeitpunkt und der Auswertung der durch **`EntryScore`** Instanzen repräsentierten Wortmodelleingänge. Diese beiden Aktionen werden unabhängig vom Modelltyp in der Methode **`Recognizer::timestep_nb_begin`** durchgeführt.	

Aktion	EFSE	ISEO
Verarbeitung eines Merkmalsvektors (modellabhängig)	Für die vom Modelltyp abhängige Bewertung der Merkmalsvektoren und Zustände existiert in den von **`Recognizer`** abgeleiteten Modellklassen eine Methode mit dem Namen ***`Modellklasse`*`::timestep_nb`**, die von den Methoden ***`Modellklasse`*`::recog_obs`** bzw. ***`Modellklasse`*`::recog_ts`** aus aufgerufen wird. Die Arbeitsweise dieser Methoden wird im Anschluß an diese Tabelle näher erläutert.	
Hypothesengenerierung	Nach der Abarbeitung aller Merkmalsvektoren wird die virtuelle Funktion **`Recognizer::get_nbest`** aufgerufen, die auf Basis der in den **`TimeStep`** Instanzen gespeicherten Informationen über die Bewertungen der Wortmodelle eine Instanz der Klasse **`Result`** mit den *n*-besten Ergebnishypothesen füllt. Die Hypothesengenerierung bedient sich eines modifizierten A* Algorithmus zur Auswertung des durch die **`TimeStep`** Instanzen definierten „Wortgitters“. Die Modellklassen können die **`get_nbest`** Methode überladen, wenn eine spezielle Auswertung erforderlich ist, wie das z.B. bei der Schlüsselworterkennung mit den „ruske*“ Modelltypen der Fall ist.	
Interprozesskommunikation	Falls **`EFSE`** als externer Spracherkennerprozess läuft, wird das durch Aufruf der Methode **`Result::add_to_Str`** in textuelle Form umgewandelte Resultat von der Funktion **`csr1_server`** an den steuernden Prozess zurückgeschickt.	Die Resultate der externen Spracherkennerprozesse werden von der Funktion **`etb_relais`** entgegengenommen und zur Abholung durch das Tcl-Skript in einer Warteschlange bereitgestellt.
Ergebnisauswertung	Die Ergebnisse werden gemäß den gewählten Optionen auf den Bildschirm ausgegeben oder in Labelfiles gespeichert.	Das Tcl-Skript holt in regelmäßigen Abständen Resultate vom Relais-Prozess ab, und reagiert je nach Einsatzzweck durch simple Darstellung oder andere Aktionen.

Das Entity-Relationship-Model in Abbildung 13 zeigt einen Überblick der Datenstrukturen im Umfeld der **`Recognizer`** Klasse, die im wesentlichen unabhängig vom Modelltyp sind. Drei Bereiche im Diagramm sind besonders gekennzeichnet: Als „dynamisch“ werden die Datenstrukturen bezeichnet, deren Inhalt sich mit jedem Erkennungsvorgang ändert, also z.B. die aktuellen Zustandsbewertungen. Die „semi-statischen“ Datenstrukturen werden beim Laden des Sprachmodells auf Basis der „statischen“ Daten initialisiert und verändern ihren Wert nicht, solange keine Modifikation des Modells zur Laufzeit erfolgt, etwa das Hinzufügen eines neuen Wortes zum Vokabular. Am unteren Rand der Abbildung 13 sind die Modellklassen dargestellt, die von **`Recognizer`** abgeleitet sind. Diese enthalten je nach Modelltyp zusätzliche Datenstrukturen, die alle Parameter der kleinsten unteilbaren Sprachmodelleinheiten (entsprechend den **`Atom`** Instanzen in der Grammatik) definieren. So enthält z.B. Klasse **`Schlauer`** ein Array von **`Phonem`** Instanzen, die jeweils die Parameter eines Phonem-KHMM definieren. Einige Modellklassen (**`KF2`**, **`Ruske`**, **`Schlauer`**) sind als Template-Klassen ausgelegt, damit die Variation von Parametern wie der Zustandsanzahl oder der Anzahl Wahrscheinlichkeitsdichtefunktionen pro Zustand, die zur Laufzeit konstant sind, nicht eine weitere Zergliederung der Modellparameter in dynamisch belegte Speicherbereiche erfordert und dem Compiler Optimierungen wie das „loop-unrolling“ ermöglicht werden.

Aus Effizienzgründen wurde ein großer Teil der Funktionalität in die Modellklassen verlagert; nur so können z.B. die Eigenschaften spezieller Übergangsstrukturen u.ä. zur Geschwindigkeitssteigerung ausgenutzt werden. Andererseits enthalten die Modellklassen dadurch stellenweise recht ähnlichen Quellcode, was der Übersichtlichkeit abträglich ist. Bei der Initilisierung werden durch die Modellklassen zwei Arrays aus **`OwnerScore`** Instanzen (**`Recognizer::scores1`**, **`Recognizer::scores2`**) erzeugt, die alternierend zur Speicherung der $\delta_j^w(t)$ (siehe Abschnitt 4) verwendet werden. Die Zeiger **`Recognizer::now`** und **`Recognizer::next`** zeigen auf jenes der beiden Arrays, das gerade zur Speicherung der $\delta_j^w(t)$ bzw. $\delta_j^w(t+1)$ dient. Ausser den Zustandsbewertungen enthalten die **`OwnerScore`** Instanzen auch einen ($\gamma_j^w(t)$ in (24) entsprechenden) Verweis auf die **`TimeStep`** Instanz, die bei Beginn der zugeordneten Zustandsfolge erzeugt wurde. In der Methode ***`Modellklasse`*`::timestep_nb`** werden nach Auswertung der möglichen Wortmodellzugänge durch **`Recognizer::timestep_nb_begin`** die Merkmalsvektoren bewertet und anschließend unter Berücksichtigung der möglichen Zustandsübergänge die in

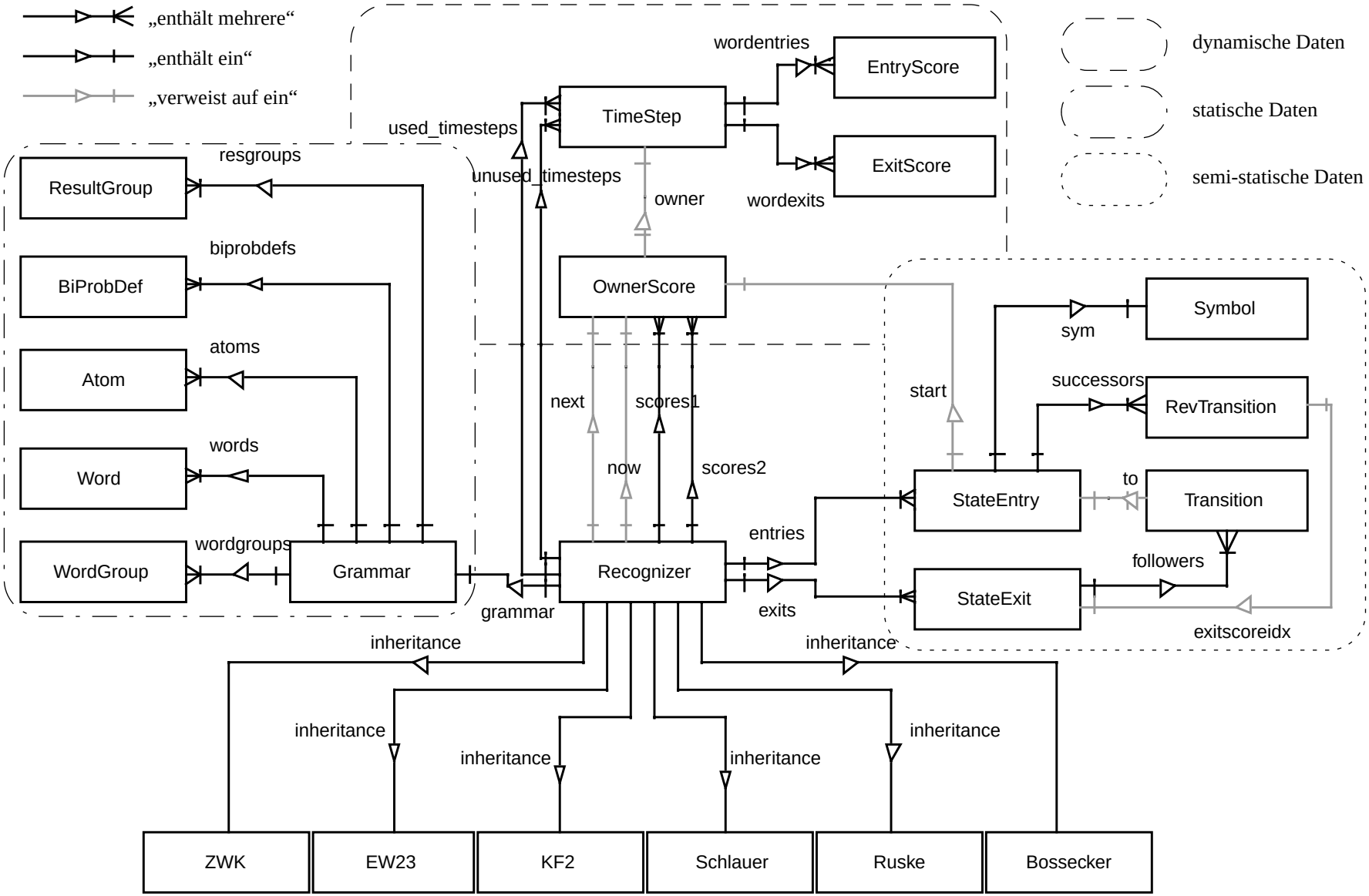

Abbildung 13: Übersicht der Datenstrukturen im Umfeld der Recognizer Klasse

`Recognizer::next` zu speichernden Werte der $\delta_j^w(t+1)$ berechnet. Wenn der Endzustand eines Wortes erreicht ist, wird seine Bewertung in den **`ExitScore`** Instanzen des aktuellen **`TimeStep`** vermerkt. Außerdem werden gemäß den Daten der **`StateExit`** Instanzen Bewertungen für die Übergänge in andere Wörter berechnet, und diese in den **`StateEntry`** Instanzen der Zielwörter zur Verarbeitung (und Speicherung in einer **`EntryScore)`** Instanz im nächsten Zeitschritt abgelegt. Die Wortübergangsauswertung kann auch - z.B. bei der Modellklasse **`Ruske`** - entfallen.

Im (über 40000 Zeilen umfassenden) Programmcode befinden sich noch eine ganze Reihe weitere Klassen und Methoden, die weniger komplex zusammenhängen, und deren Bedeutung aus den Quelltexten in der Regel leicht ersichtlich ist. Dazu gehören u.a. die zum Aufbau der Grammatik verwendeten Klassen, deren Funktion auch aus der in der folgenden Tabelle erläuterten Syntax der Grammatikdatei hervorgeht:

Befehlssyntax	Erläuterung
# <Kommentar>	Leere Zeilen, oder solche, die mit dem Buchstaben „#“ beginnen, werden ignoriert und können zu Dokumentationszwecken verwendet werden.
ATOM <Atom-Name> <Info>	Ein Atom entspricht der kleinsten unteilbaren Einheit des Sprachmodells. Diese kleinsten Einheiten können z.B. Phonem- oder Wortmodelle sein, sie müssen mit dem ATOM Befehl deklariert werden, bevor sie verwendet werden können. Der Parameter <Atom-Name> wählt einen beliebigen Namen, <Info> wird abhängig vom Modelltyp verwendet, um bei Verwendung der EFSE Option „-loadasc“ die Quelldatei für das entsprechende Modell zu finden.
WORD <Wort-Name> <Atom-Name> [<Atom-Name> ...]	Ein Wort entspricht der kleinsten Einheit, die im Erkennungsergebnis ausgewiesen wird. Mit dem WORD Befehl werden Wörter als Aneinanderreihung von einem oder mehreren Atomen definiert.
WORDGROUP <Gruppen-Name> [<Wort-Name> ...]	Mit dem WORDGROUP Befehl können Wörter zu Gruppen zusammengefasst werden. Viele Befehle akzeptieren als Parameter Wörter oder Wortgruppen, dies wird im folgenden mit „<Wörter...>“ abgekürzt. Es ist möglich, leere Wortgruppen zu generieren.

Befehlssyntax	Erläuterung
BIPROB <Wörter...> <Wörter...> <Übergangswahrscheinlichkeit>	Während die Übergangswahrscheinlichkeiten der einzelnen Zustände in den Quelldateien des Sprachmodells gespeichert sind, werden die Wortübergänge erst mit dem BIPROB Befehl ermöglicht. Die Übergänge sind gerichtet, d.h. die Zeile BIPROB Haus Garten 1.0 gestattet nur einen Übergang vom Ende des Wortes „Haus“ zum Beginn des Wortes „Garten“, nicht aber umgekehrt. Für Quelle und Ziel des Übergangs können mehrere Wörter oder Wortgruppen angegeben werden, jedoch müssen ggf. Anführungszeichen verwendet werden: BIPROB „Maler Maurer“ „Meister Geselle“ 0.5 gestattet z.B. die Sequenzen „Maler Meister“, „Maler Geselle“, „Maurer Meister“ und „Maurer Geselle“. Die Übergangswahrscheinlichkeiten werden „spät“ ausgewertet: Wenn z.B. ein Übergang zwischen zwei Wortgruppen ermöglicht wurde, und zur Laufzeit wird einer der Gruppen ein Wort hinzugefügt, so gilt die Übergangswahrscheinlichkeit der Wortgruppe auch für das neue Wort.
RESULTGROUP <Resultatgruppenname> <Wörter...> <Wörter...>	Für jede Resultatgruppe, die mit diesem Befehl definiert werden kann, erzeugt das System einen Satz von Ergebnishypothesen. Jeder Resultatgruppe müssen Wörter zugeordnet werden, mit denen die Hypothesen beginnen und enden dürfen. So könnte etwa die Definition WORDGROUP z eins zwei drei WORDGROUP k halt weiter ende RESULTGROUP Zahlen z z RESULTGROUP Befehle k k dazu dienen, aus jedem Erkennungsvorgang eine Zahlwortketten- und eine Kommandowörterhypothese zu erhalten. Zwei Resultatgruppen sind nur dann voneinander unabhängig, wenn es keine Übergänge zwischen den ihnen angehörenden Wörtern gibt.

Befehlssyntax	Erläuterung
PENALTY <Wert> <Wörter...>	Jedem Wort ist gemäß (27) ein Faktor f_w zugeordnet, durch den eine benachteiligende ($f_w < 1$) oder bevorzugende ($f_w > 1$) Bewertung gegenüber anderen Wörtern erreicht werden kann. Dieser Faktor kann mit dem PENALTY Befehl für ein oder mehrere Wörter gewählt werden. Der Befehl arbeitet „multiplikativ“, das heißt, die Befehlsfolge PENALTY 2.0 Haus Garten PENALTY 3.0 Haus ist äquivalent PENALTY 6.0 Haus PENALTY 2.0 Garten Der PENALTY Befehl darf für die Modelltypen „ruske3_10“ und „ruske3_1“ nicht verwendet werden.
THRESHOLD <Wert> <Wörter...>	Bei den Modelltypen „ruske3_10“ und „ruske3_1“ ist gemäß Abschnitt 5.2 jedem Wort ein K_w (für die 1-Schlüsselwort-Erkennung) bzw. ein G_w (für die n-Schlüsselwort-Erkennung) zugeordnet. Dieser Wert kann mit dem THRESHOLD Befehl gewählt werden. Der THRESHOLD Befehl darf für andere als die genannten Modelltypen nicht verwendet werden.
INSIGNIFICANT <Wörter...>	Der Befehl INSIGNIFICANT erlaubt es, Wörter in der Ausgabe der Erkennungsergebnisse zu ignorieren (z.B. einzelne Phoneme bei der PAF Methode). Auf die Erzeugung der n-besten Hypothesen hat dies keinen Einfluß, das heißt, Hypothesen können sich ggf. um ausschließlich nicht sichtbare Wörter voneinander unterscheiden.
TIMEVARTOLERANCE <Wert>	Der mit diesem Befehl spezifizierte Wert gibt an, um welches Vielfache der Standardabweichung die Länge eines Wortes von der mittleren Länge abweichen darf, die mit dem „WORDTIMEMEANVAR“ Befehl festgelegt wurde. 2.0 hat sich als brauchbarer Wert erwiesen, und ist daher voreingestellt.
WORDTIMEMEANVAR <Mittelwert> <Standardabweichung> <Wörter...>	Wird mit diesem Befehl die mittlere Länge eines Wortes und ihre Standardabweichung vorgegeben, so wird dieses Wort nicht erkannt, falls die vermutete Dauer in der Äußerung um mehr als das mit „TIMEVARTOLERANCE“ gewählte Vielfache der Standardabweichung vom Mittelwert abweicht.

Das Spracherkennungsprogramm **EFSE** kann unmittelbar gestartet werden, oder es kann als externer Spracherkennerprozess von **ISEO** aus genutzt werden, das eine graphische Benutzeroberfläche zur Verfügung stellt und permanente Sprachaufnahme sowie die Veränderung des Vokabulars zur Laufzeit gestattet. Beim direktem Start von der Kommandozeile stellt **EFSE** dem Anwender eine Vielzahl von Optionen zur Verfügung, die in der folgenden Tabelle erläutert werden:

EFSE Option	Gültige Parameter Voreinstellung	Erläuterung
-help	Keine	Führt zur Ausgabe eines Hilfstextes, der ähnlich dieser Tabelle die verfügbaren Optionen nennt. Anschließend wird das Programm verlassen.
-cwd	Verzeichnisname „.“	**EFSE** macht das angegebene Verzeichnis zum aktuellen Arbeitsverzeichnis. Alle mit anderen Optionen spezifizierten Pfad- und Dateinamen werden relativ zum aktuellen Arbeitsverzeichnis interpretiert.
-setup	Dateiname „csr1.setup“	Wählt eine Datei zur Speicherung der aktuellen Optionen aus. Gemäß dieser Datei werden beim Programmstart die Optionen gesetzt, anschließend werden die auf der Kommandozeile angegebenen Optionen ausgewertet. Wird das Programm regulär beendet, werden die gewählten Optionen in der Datei gespeichert.
-show	Keine	Zeigt die aktuell gewählten Optionen an. Anschließend wird das Programm verlassen.
-audiogain	Ganze Zahlen von 0 bis 100 70	Legt die Eingangsaussteuerung bei direkter Sprachaufnahme fest. Diese Option ist nur unter HP-UX, Linux und bei Verwendung eines „DeskLab“ wirksam.
-audioin	„none“, „mic“, „line“ none	Wählt die Quelle für direkte Sprachaufnahmen aus. Diese Option ist nur unter HP-UX, Linux und bei Verwendung eines „DeskLab“ wirksam.
-bandpass	„true“ oder 1, „false“ oder 0 0	Schaltet den Bandpassfilter für direkte Sprachaufnahmen oder die Verarbeitung von Sprachsignaldateien ein oder aus.
-batch	Keine	Veranlasst **EFSE**, die auf der Kommandozeile angegebenen Dateien als Listen von Dateinamen zu interpretieren, und die Ausgabe der Fortschrittsanzeige zu unterlassen. Ohne diese Option werden alle Kommandozeilenparameter, die keine bekannte Option darstellen, als Namen von Dateien interpretiert, die zu erkennende Sprachsignale oder Merkmalsvektoren enthalten.

EFSE Option	Gültige Parameter Voreinstellung	Erläuterung
-beam	Ganze Zahlen von 0 bis 2000 0	Wird ein Wert ungleich 0 spezifiziert, so werden Pfade durch das Modell nicht weiterverfolgt, wenn ihre Bewertung um den angegebenen Betrag unter der Bewertung des besten Zustands liegt. Diese Option ist bei den Modelltypen „ruske3_10“ und „ruske3_1“ ohne Wirkung.
-bestspot	„true“ oder 1, „false“ oder 0 0	Wählt bei den Modelltypen „ruske3_10“ und „ruske3_1“ zwischen n-Schlüsselworterkennung und 1-Schlüsselworterkennung.
-binname	Dateiname „test.rbi“	Legt den Namen der Datei fest, aus der die binäre Repräsentation des Sprachmodells geladen wird, sofern nicht „-loadasc“ oder „-loadptbl“ spezifiziert wurden. Wird die Option „-savebin“ verwendet, so wird das Sprachmodell unter diesem Namen gespeichert.
-byteswap	„true“ oder 1, „false“ oder 0 0	Falls Sprachsignaldateien verarbeitet werden, wählt diese Option aus, ob die höher- und niederwertigen 8 bit eines gelesenen 16-bit Wortes vertauscht werden.
-delta	„htk“, „5point“ „htk“	Wählt die Methode aus, nach der die zeitlichen Ableitungen der Merkmalsvektoren berechnet werden. Zur Wahl stehen ein HTK-kompatibles Verfahren und die 5-Punkte-Differentiation
-falign	Fließkommazahlen von 2.2e-308 bis 1.7e+308 0.0	Wird ein Wert ungleich 0.0 spezifiziert, so findet ein sogenanntes „forced alignment“ statt: Jeweils zwei Dateinamen werden dann für einen Erkennungsvorgang verwendet, wobei der erstere nicht als Dateiname, sondern als Transkription des Sprachsignals in der mit dem zweiten Namen spezifizierten Datei interpretiert wird. Die Grammatik des geladenen Sprachmodells wird verworfen, und durch eine anhand der Transkription neu erzeugte ersetzt. Dabei wird der spezifizierte Wert als „PENALTY“ für alle Wortmodelle eingesetzt.
-featdir	Verzeichnisname „.“	Namen von Sprachsignal- oder Merkmalsvektorendateien wird der spezifizierte Verzeichnisname vorangestellt, sofern es sich nicht um absolute Dateinamen handelt.
-featext	Zeichenkette „“	Spezifiziert eine Zeichenkette, die den Namen der Sprachsignal- oder Merkmalsvektorendateien hinzugefügt wird.
-ftype	„cep“, „mel“ , „cis“, „wav“ „cep“	Legt den Typ der zu verarbeitenden Dateien fest: Cepstralkoeffizienten, Mel-Cepstralkoeffizienten, Codebuch-Index-Sequenzen (nur für diskrete HMM) oder Sprachsignaldateien.

EFSE Option	Gültige Parameter Voreinstellung	Erläuterung
-grammar	Dateiname „test.grammar“	Spezifiziert den Namen der Grammatik-Datei, die gelesen wird, wenn die „-loadasc“ Option verwendet wird.
-hipass	„true“ oder 1, „false“ oder 0 0	Schaltet den Hochpassfilter für die Verarbeitung von Sprachsignaldateien ein oder aus.
-ils	„true“ oder 1, „false“ oder 0 0	Teilt **EFSE** mit, ob Sprachsignaldateien mit einem ILS-Header beginnen.
-labdir	Verzeichnisname „./lab“	Legt den Namen des Verzeichnisses fest, in das Labeldateien gespeichert werden, wenn die „-labout“ Option verwendet wird.
-labout	„true“ oder 1, „false“ oder 0 0	Falls aktiviert, werden Labeldateien in einem HTK-ähnlichen Format gespeichert: In jeder Zeile der Datei werden Start- und Endzeitpunkt eines Wortes, das Wort selbst sowie seine Bewertung gespeichert.
-loadasc	Keine	Veranlasst **EFSE**, statt eines binär gespeicherten Sprachmodells textuelle Quelldateien einzulesen, deren Format vom eingestellten Modelltyp abhängt. Außer den modelltypabhängigen Quelldateien wird eine Grammatikdatei eingelesen, deren Name mit der „-grammar“ Option gewählt werden kann.
-loadptbl	Keine	Veranlasst **EFSE**, statt einer binären Sprachmodelldatei eine maschinenunabhängig textuell gespeicherte einzulesen. Auf diese Weise können Sprachmodelle zwischen Rechnern unterschiedlicher Architektur ausgetauscht werden, ohne die modelltypabhängigen Quell- und Grammatikdateien kopieren zu müssen.
-lopass	„true“ oder 1, „false“ oder 0 0	Schaltet den Tiefpassfilter für die Verarbeitung von Sprachsignaldateien ein oder aus.
-micro	„true“ oder 1, „false“ oder 0 0	Schaltet den Mikrophoncharakteristikfilter für direkte Sprachaufnahmen oder die Verarbeitung von Sprachsignaldateien ein oder aus. Dieser Filter überhöht bestimmte Frequenzbereiche in gleicher Weise wie dies bei genormten Telefonmikrophonen der Fall ist.
-modeldir	Verzeichnisname „./hmms“	Legt das Verzeichnis fest, aus dem Quelldateien für Sprachmodelle gelesen werden, wenn die Option „-loadasc“ verwendet wird.
-modellist	Dateiname „	Weitgehend obsolete Option, dient nur noch für den Modelltyp „bossecker“ zur Wahl des Namens der Datei, in der das RNN gespeichert ist.

EFSE Option	Gültige Parameter Voreinstellung	Erläuterung
-modeltype	„schlauer“, „ew23“, „schmidt“, „volk“, „bossecker“, „zwk“, „kf2“, „gerhard4“ „,ruske3_10“, „ruske3_1“ „schlauer“	Legt den Typ des zu ladenden Modells fest, wenn die „-loadasc“ Option verwendet wird. Die (unsystematisch gewählten) Namen stehen für verschiedene Modellstrukturen und Observablenbewertungsverfahren, nicht alle Typen lassen alle möglichen Arten von Erkennungsmodi zu. „schmidt“, „volk“ und „schlauer“ stehen für kontinuierliche HMM mit 1, 5 und 10 Wahrscheinlichkeitsdichtefunktionen pro Zustand, 3 Zustände pro Wortuntereinheit, und strikter links-rechts Übergangsstruktur. „gerhard4“ steht für ein KHMM mit 4 Zuständen pro Wortuntereinheit und strikter links-rechts Übergangsstruktur. „kf2“ steht für ein KHMM mit spezieller Übergangsstruktur. „zwk“ steht für ein semi-kontinuierliches HMM mit 13 Zuständen pro Wort. „ew23“ steht für ein diskretes HMM mit 5 Zuständen pro Wort und interner Verweildauermodellierung. „ruske3_1“ und „ruske3_10“ stehen für KHMM zur Schlüsselworterkennung mit einer oder 10 Wahrscheinlichkeitsdichtefunktionen pro Zustand, die auf spezielle Weise ausgewertet werden.
-nbest	Ganze Zahlen von 1 bis 5000 1	Wählt die Zahl der auszugebenden „n-besten“ Hypothesen aus. Die Wahl von Werten ungleich 1 ist nur sinnvoll, wenn Grammatik und Modelltyp die Erzeugung mehrerer Hypothesen zulassen.
-normal	„true“ oder 1, „false“ oder 0 0	Falls aktiviert, werden gelesene Sprachsignaldateien als erstes auf die maximal mögliche Amplitude normiert. Dies sollte in der Regel keinen Einfluss auf die Erkennungsleistung haben.
-pbest	Ganze Zahlen von 0 bis 2000 0	Zahlen ungleich 0 aktivieren die Ausgabe aller „n-besten“ Hypothesen, deren Bewertung um nicht mehr als den angegebenen Betrag geringer als die maximale Bewertung ist. Die mit der Option „-nbest“ eingestellte Hypothesenanzahl kann dabei nicht überschritten werden.
-phondat	„true“ oder 1, „false“ oder 0 0	Teilt **EFSE** mit, ob einzulesende Sprachsignaldateien einen PHONDAT II header enthalten.
-pipe	Keine	Aktiviert einen Modus, in dem **EFSE** über **`stdin`** und **`stdout`** mit dem aufrufenden Programm (z.B. **`ISEO`**) kommuniziert.

EFSE Option	Gültige Parameter Voreinstellung	Erläuterung
-print	Keine	Veranlasst den Ausdruck aller Modellparameter zur Unterstützung bei der Fehlersuche
-printscore	„true“ oder 1, „false“ oder 0 0	Falls aktiviert, werden zu ausgegebenen Hypothesen auch die entsprechenden Bewertungen ausgegeben.
-ptblname	Dateiname „test.ptbl“	Legt den Namen der Datei fest, aus der die maschinenunabhängige textuelle Repräsentation des Sprachmodells geladen wird, wenn die „-loadptbl“ Option verwendet wird. Wird die Option „-saveptbl“ verwendet, so wird das Sprachmodell unter diesem Namen gespeichert.
-pvmi	TID, -1	Falls ungleich -1, wird der Wert als ID des PVM-Prozesses interpretiert, von dem Befehle entgegen genommen werden sollen (ähnlich „-pipe“)
-pvmo	TID	Falls mit „-pvmi“ eine ID spezifiziert wurde, wird diese Option verwendet, um **EFSE** mitzuteilen, an welchen Prozess die Resultate zu senden sind.
-r16	„true“ oder 1, „false“ oder 0 0	Falls aktiviert, werden zu lesende Sprachsignaldateien vor der Verarbeitung von 16kHz auf 8kHz Abtastrate gewandelt.
-record	Keine	Wählt den Betriebsmodus „direkte Sprachaufnahme“ aus, in dem wiederholt eine Äußerung aufgenommen, verarbeitet und das Resultat ausgegeben wird. Diese Option steht nur zur Verfügung, wenn der Rechner über Audio-Hardware verfügt. Während der Verarbeitung der Äußerungen findet keine Sprachaufnahme statt.
-refdir	Verzeichnisname	Obsolet, wird nicht mehr unterstützt.
-refext	Zeichenkette	Obsolet, wird nicht mehr unterstützt.
-savebin	Keine	Veranlasst **EFSE**, eine binäre Repräsentation des Sprachmodells zu sichern. Siehe auch Option „-binname“.
-saveptbl	Keine	Veranlasst **EFSE**, eine maschinenunabhängige, textuelle Repräsentation des Sprachmodells zu sichern. Siehe auch Optionen „-ptblname“ und „-loadptbl“.
-udetect	„true“ oder 1, „false“ oder 0 0	Falls aktiviert, werden Sprachsignaldateien vor der Erkennung durch den Äußerungsdetektor vorverarbeitet. Es wird nur eine Äußerung pro Sprachsignaldatei extrahiert.

Das Programm **ISEO** ist als benutzerfreundliche Oberfläche für konkrete Anwendungsfälle der automatischen Spracherkennung konzipiert, die Funktion der in Abbildung 14 gezeigten Bedienelemente sollten möglichst intuitiv klar sein.

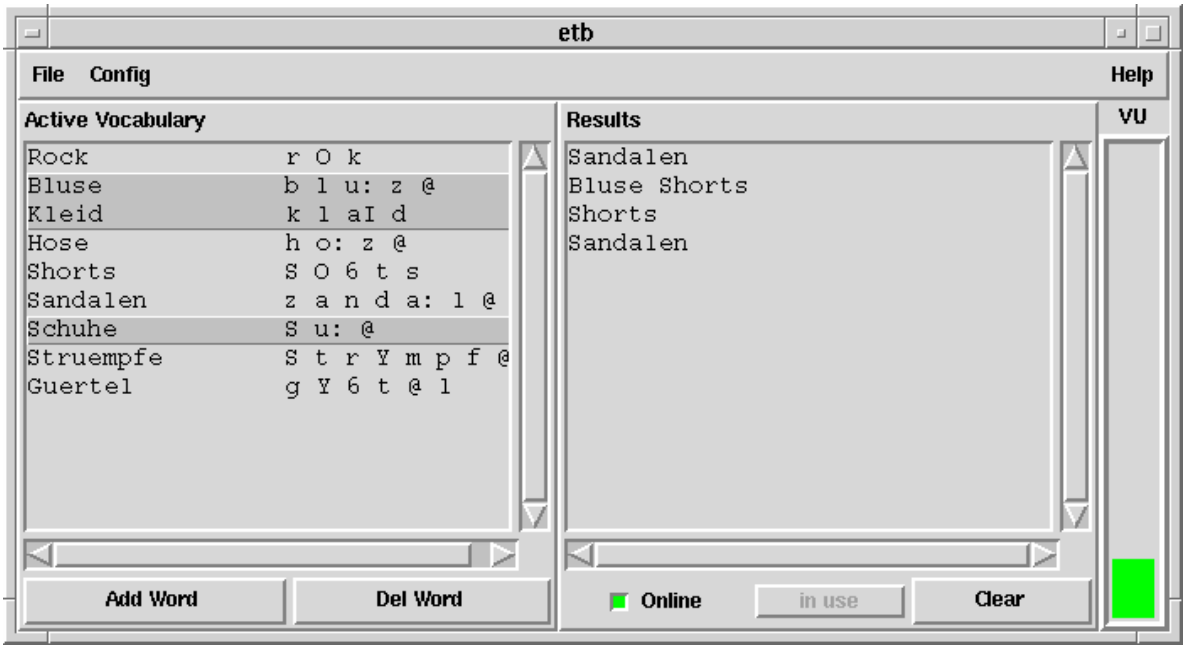

Abbildung 14: Hauptfenster der graphischen Benutzeroberfläche des Programms **ISEO**

Im „File“ Menü können Wortlisten geladen oder gespeichert werden, das „Config“ Menü erlaubt die (De-)Aktivierung der Aussteuerungsanzeige und die Auswahl eines erweiterten Resultatanzeigemodus. Zwei Listen stellen das aktuelle Vokabular und die Erkennungsresultate der Äußerungen dar. Ein Doppelklick auf eines der Wörter in der Vokabularliste lässt einen Dialog zur Einstellung wortabhängiger Parameter erscheinen. Mit „Add Word“ und „Del Word“ können Wörter dem Vokabular hinzugefügt oder aus demselben gelöscht werden. Mit dem „Online“ Schalter kann das System (de-)aktiviert werden, der „Clear“ Knopf löscht die Erkennungsresultatliste. Eine „in use“ Anzeige informiert darüber, ob die externen Erkennerprozesse gerade eine Äußerung verarbeiten.

ISEO kann durch Modifikation des Tcl-Skripts („etb.tcl“) verschiedenen Aufgaben angepasst werden, ohne dass hierzu Detailkenntnisse der restlichen Programmteile erforderlich sind. So wurde **ISEO** u.a. so verändert, dass es als Sprachsteuerung eines externen WWW-Browsers (Netscape Navigator) dienen konnte. **ISEO** kennt nur eine kleine Anzahl von Konfigurations-

optionen, die beim Start von der Kommandozeile gewählt werden können. Sie sind in der folgenden Tabelle erläutert:

Option	Gültige Parameter Voreinstellung	Erläuterung
-help	Keine	Führt zur Ausgabe eines Hilfstextes, der ähnlich dieser Tabelle die verfügbaren Optionen nennt. Anschließend wird das Programm verlassen.
-cwd	Verzeichnisname „.“	**ISEO** macht das angegebene Verzeichnis zum aktuellen Arbeitsverzeichnis. Alle mit anderen Optionen spezifizierten Pfad- und Dateinamen werden relativ zum aktuellen Arbeitsverzeichnis interpretiert.
-setup	Dateiname „etb.setup“	Wählt eine Datei zur Speicherung der aktuellen Optionen aus. Gemäß dieser Datei werden beim Programmstart die Optionen gesetzt, anschließend werden die auf der Kommandozeile angegebenen Optionen ausgewertet. Wird das Programm regulär beendet, werden die gewählten Optionen in der Datei gespeichert.
-show	Keine	Zeigt die aktuell gewählten Optionen an. Anschließend wird das Programm verlassen.
-argw	Zeichenkette „“	„-argw“ und „-exe“ spezifizieren Zeichenketten, deren Verwendung vom Tcl-Script abhängt. In der Regel nennt „-argw“ eine Argumentenliste, die beim Start externer Erkennerprozesse an diese übergeben wird, „-exe“ wählt den Programmnamen für externe Erkennerprozesse.
-exe	Zeichenkette „csr1“	
-maxuttlen	Ganze Zahlen größer 1000 32000	Falls eine Äußerung detektiert wird, die mehr Abtastwerte enthält, als mit dieser Option eingestellt wurde, so findet keine Erkennung statt, und die Äußerung wird verworfen.
-niceinc	Ganze Zahlen, gültiger Bereich ist betriebssystemabhängig. 5	Der „nice-level“ externer Erkennerprozesse wird um den angegebenen Wert relativ zum **ISEO** Prozess erhöht. Auf diese Weise kann erreicht werden, dass bei nicht verteiltem Betrieb die Aufnahme, Filterung und Äußerungsdetektion hinreichend viel Rechenzeit erhält.
-audiogain	Ganze Zahlen von 0 bis 100 70	Legt die Eingangsaussteuerung bei direkter Sprachaufnahme fest. Diese Option ist nur unter HP-UX, Linux und bei Verwendung eines „DeskLab“ wirksam.
-audioin	„none“, „mic“, „line“ none	Wählt die Quelle für direkte Sprachaufnahmen aus. Diese Option ist nur unter HP-UX, Linux und bei Verwendung eines „DeskLab“ wirksam.

Option	Gültige Parameter Voreinstellung	Erläuterung
-bandpass	„true“ oder 1, „false“ oder 0 0	Schaltet den Bandpassfilter für direkte Sprachaufnahmen oder die Verarbeitung von Sprachsignaldateien ein oder aus.
-micro	„true“ oder 1, „false“ oder 0 0	Schaltet den Mikrophoncharakteristikfilter für direkte Sprachaufnahmen oder die Verarbeitung von Sprachsignaldateien ein oder aus. Dieser Filter überhöht bestimmte Frequenzbereiche in gleicher Weise wie dies bei genormten Telefonmikrophonen der Fall ist.

Abbildungsverzeichnis

Übergangsstruktur eines strikten links-rechts HMM 8
PAF Grammatik für die 1-Schlüsselwort-Erkennung 14
PAF Grammatik für die n-Schlüsselwort-Erkennung 14
Exemplarischer Verlauf der Wortbewertungen eines in der Äußerung enthaltenen sowie eines nicht enthaltenen Wortes 19
Kommunikationsstruktur des Erkennersystems **ISEO** 24
Gemessene Bewertungen und Längen gesprochener Phoneme in Phondat . . . 35
Vorberechnete und gemessene mittlere Längen der Schlüsselwörter 36
Ergebnisse der n-Schlüsselworterkennung für die getesteten Verfahren 40
Vergleich der vorgegebenen und tatsächlich erzielten Gesamterkennungsraten unter Verwendung der phonembasierten GFSE 41
Vergleich der Erkennungs- und Fehlalarmraten der phonembasierten GFSE bei optimal angepassten, wortunabhängig konstanten und berechneten Schwellenwerten 42
Vorberechnete und gemessene Bewertungen der Schlüsselwörter 44
Verteilungsdichten der Bewertungen für die Phoneme /**S**/, /**n**/ und /**6**/ 45
Übersicht der Datenstrukturen im Umfeld der **Recognizer** Klasse 56
Hauptfenster der graphischen Benutzeroberfläche des Programms **ISEO** 65

Verzeichnis der Tabellen

Phoneminventar 33

Einfluß der Zeitlimitierung auf die Ergebnisse der 1-Schlüsselwort-Erkennung 36

Ergebnisse der 1-Schlüsselwort-Erkennung bei Verwendung nicht optimierter Werte 37

Ergebnisse der 1-Schlüsselwort-Erkennung bei Verwendung wortangepasster bzw.. 38

Resultate der phonembasierten GFSE bei vorgegebenen Erkennungsraten 43