Spracherkennung

Peter Flock, Helge Spieker
Hochschule Bonn-Rhein-Sieg
Email: {peter.flock,helge.spieker}@smail.inf.h-brs.de

***Zusammenfassung*—Die vorliegende Arbeit betrachtet den Aufbau und die Funktionsweise von Spracherkennungssystemen von der theoretischen und der praktischen Seite. Sie besteht aus einem grundlegenden Einstieg in die Herausforderungen und Einflüsse der Problematik und vermittelt wie ein typisches System für *Speech-to-Text*-Anwendungen funktioniert. Im praktischen Teil der Arbeit wird ein solches System auf Basis frei verfügbarer Komponenten prototypisch eingesetzt und die Anwendbarkeit auf Heimautomatisierungsaufgaben evaluiert.**

I. Einleitung

Die Sprache ist das wichtigste Kommunikationsmedium für Menschen untereinander. Sie erlaubt es eine vielfältige Bandbreite an Informationen zu übermitteln und ist ein zentraler Bestandteil des Alltags. Die Verarbeitung von Sprache durch Computer (*Automatic Speech Recognition*) ist jedoch ein komplexes Problem mit vielen Facetten.

Der allgemeine Begriff Spracherkennung umschreibt mehrere Teilprobleme mit unterschiedlichen Zielen und Zwecken. Ein wesentlicher Verwendungszweck der Spracherkennung ist die Transkription gesprochener Sprache in Text (*Speech-to-Text*). Darüber hinaus kann die Sprache als biometrisches Merkmal zur Sprecheridentifikation und -verifikation genutzt werden.

In der vorliegenden Arbeit wird das Problem der Sprachtranskription vorgestellt. Es wird auf die theoretischen Grundlagen, sowie den prinzipiellen Aufbau eines Systems zur Erkennung und Verarbeitung von Sprache durch Computer, eingegangen. Dabei werden die einzelnen Komponenten diskutiert und vorgestellt, welche in aktuellen Systemen Verwendung finden. Der zweite Teil der Arbeit beschäftigt sich mit dem praktischen Einsatz eines Spracherkennungssystems auf Basis frei verfügbarer Komponenten.

Im Rahmen der Arbeit liegt der Fokus auf der Fragestellung, inwiefern verfügbare Spracherkennungssysteme für den Einsatz in der Heimautomatisierung anwendbar sind. Dies beeinflusst insbesondere die Gestaltung des praktischen Teils.

Die nachfolgenden Ausführungen bieten einen Überblick über das Gebiet der Spracherkennung, können aber, aufgrund des Themenumfangs und der vielen unterschiedlichen Teilproblemstellungen, keine umfassende Einführung bieten. Dazu wird auf die umfangreichen Werke von Jurafsky und Martin [1], sowie Wölfel und McDonough [2] verwiesen. Eines der frühen Standardwerke stammt von Rabiner und Juang [3]. Eine deutschsprachige Einführung bietet Euler [4]. Spezifische Quellen, die sich mit bestimmten Teilaspekten der Spracherkennung befassen, werden in den jeweiligen Abschnitten angegeben.

Bereits in der Mitte des 20. Jahrhunderts begann die Entwicklung von Spracherkennungssystemen, wobei sich sowohl die damals verwendete Technik als auch ihre Performanz deutlich von aktuellen Systemen unterscheidet. Im Folgenden sind einige Meilensteine der Spracherkennung aufgeführt, um einen Überblick über die Geschichte zu ermöglichen. Ausführliche Beschreibungen der historischen Entwicklung wurden von Juang und Rabiner [5] und Hinton, Deng, Yu u. a. [6] veröffentlicht.

1952	*AT&T Bell Labs* entwickelt auf Basis von *Pattern Matching*-Verfahren ein System, dass sprecherabhängig Zahlen erkennt.
1960	Das *Hidden Markov Modell* wird entwickelt.
1971	Die US-DARPA finanziert ein Forschungsprojekt, an dem sich mehrere Forschungseinrichtungen beteiligen. *CMU Hydra* erfüllt alle Erfolgskriterien, kann aber trotzdem nicht überzeugen. Finanzielle Mittel für Spracherkennung werden gestrichen.
1992	Fortschritte im Bereich statistischer Sprachmodellierung vergrößern die Vokabulare erheblich und steigern die Effizienz der Spracherkennung.
2009	*Deep Learning* und *neuronale Netze* etablieren sich als leistungsfähige Alternative zu traditionellen Ansätzen.

II. Theoretische Grundlagen

Vor der Beschreibung des Aufbaus eines Spracherkennungssystems, werden zunächst die äußeren Parameter diskutiert, welche die unterschiedlichen Anwendungsgebiete der Spracherkennung charakterisieren. Diese Parameter beeinflussen die Konzeption und Planung eines Spracherkennungssystems.

A. Einflussfaktoren

Ein erster Einflussfaktor ist die Anzahl der unterstützten Wörter des Spracherkennungssystem, welche das Vokabular bzw. das Wörterbuch bilden (vgl. [1, S. 319f.]). Die Komplexität der Spracherkennung verringert sich, je kleiner die Menge zu erkennender Wörter ist. Beispielsweise ist es relativ einfach ein Command-and-Control-System zu entwickeln, welches lediglich die zwei Wörter *Ja* und *Nein* oder zehn verschiedene Ziffern unterscheidet. Bei Transkription von Telefongesprächen oder Nachrichtensendungen umfasst der Wortschatz allerdings 20.000 bis 60.000 Wörter, wodurch die Komplexität deutlich höher ist.

Weiterhin wird das Spracherkennungssystem durch die Sprechweise beeinflusst (vgl. [1, S. 320]). Die Erkennung isolierter Wörter, die von Pausen umgeben sind, ist einfacher als die Erkennung kontinuierlicher Sprache. Außerdem ist die Erkennung gelesener oder direkt an das Gerät gerichteter Sprache weniger komplex als die von Konversationen, die ohne Rücksicht auf eine Aufnahme geführt werden. Wenn der

Sprecher gezielt das Gerät anspricht, passt er seine Stimme und Sprechweise an, so dass sie langsamer und simpler strukturiert ist, als im Gespräch mit anderen Personen.

Als dritter Einflussfaktor sind Umgebungsgeräusche und Störungen von Bedeutung (vgl. [1, S. 320]). An dieser Stelle ist insbesondere der Ort, an dem die Aufnahme durchgeführt wird, sowie Qualität und Position des Mikrofons, von Bedeutung. Ruhige Orte ohne Nebengeräusche vereinfachen die Erkennung, genau wie die Verwendung hochqualitativer Headset-Mikrofone, die nah am Kopf getragen werden und dadurch den Einfluss von Störungen reduzieren. Deshalb ist die Aufnahme eines Diktates am Schreibtisch in einem Einzelbüro im Vergleich zu einer Aufnahme aus einem Auto bei offenem Fenster relativ einfach zu verarbeiten.

Letztendlich variiert Sprache auch je nach Sprecher. Zum einen durch unterschiedliche Akzente und Dialekte, aber auch durch individuelle Betonung von Wörtern (vgl. [1, S. 320]). Wenn der Sprecher einen Standarddialekt spricht, oder den Dialekt mit dem das System trainiert wurde, ist die Erkennung leichter zu vollziehen als bei fremden Dialekten oder bei Kindern. Bei der Entwicklung zuverlässiger und robuster Systeme müssen diese Varianzen berücksichtigt werden, insbesondere bei der Auswahl der Trainingsdaten und ihrer Sprecher.

B. Phonetik der Sprache

Die menschliche Sprache besteht aus verschiedenen Einheiten (vgl. [4, Kap. 2]). Auf der obersten Ebene stehen die Sätze, die aus mehreren Wörtern zusammengesetzt sind. Jedes einzelne Wort besteht wiederum aus Silben oder Halbsilben. Die kleinste akustische Einheit der Sprache ist ein Laut (*Phon*). Diese werden wiederum zusammengefasst in der kleinsten bedeutungsunterscheidenden Einheit, den *Phonemen*. Viele Laute einer Sprache klingen ähnlich und haben keine Bedeutungsunterschiede, wenn sie in einem Wort verwendet werden. Laute, deren unterschiedlicher Klang nicht mit einem Bedeutungsunterschied verbunden ist, gehören zu einer Klasse und werden als *Allophone* bezeichnet. Diese Laute bilden ein Phonem. „Ein Phonem ist eine Gruppe von Lauten (Phonen), die ähnlich klingen und niemals einen Bedeutungsunterschied bewirken“ [4, S. 9].

Für die Phone eines Phonems (diese heißen *Allophone*) gilt, dass bei dem Austausch untereinander ein Wort seine Bedeutung nicht verändert. Wenn allerdings durch den Austausch eines Phones zwei Wörter mit unterschiedlicher Bedeutung entstehen folgt aus diesem *Minimalpaar*, dass es sich um zwei unterschiedliche Phoneme handelt, welche diese Phone beinhalten. Phoneme werden in Anlehnung an die Lautschriftsymbole des Internationalen Phonetischen Alphabets dargestellt, mit der Unterscheidung, dass sie in Schrägstrichen statt eckigen Klammern geschrieben werden, z. B. */r/* statt *[r]*. Die deutsche Sprache verwendet circa 50 Phoneme (vgl. [4, S. 10]).

Da Phoneme aus unterschiedlichen Phonen zusammengesetzt sind und es darüber hinaus unterschiedliche Varianten der Aussprache von Wörtern gibt - beispielsweise in Abhängigkeit des Dialektes, der Stimme oder des Emotionszustandes des Sprechers - bestehen verschiedene Möglichkeiten, wie Wörter aus Phonen gebildet werden. Daraus resultiert die Komplexität der Spracherkennung und die Motivation stochastische Modelle und Prozesse einzusetzen.

Tabelle I. WORD ERROR RATES AKTUELLER SYSTEME FÜR STANDARDTESTS (STAND: 2006) [1, S. 320, ABB. 9.1]

Aufgabe	Vokabular	Fehlerrate (in %)
TI Digits	11 (zero-nine, oh)	0.5
Wall Street Journal read speech	5.000	3
Wall Street Journal read speech	20.000	3
Broadcast News	64.000+	10
Conversational Telephone Speech	64.000+	20

C. Qualitätskriterien

Als Metrik für die Fehlerrate eines Systems wird seit den 80er Jahren des 20. Jahrhunderts die *Word Error Rate* (WER) verwendet (vgl. [2, Kap. 1.3 und 14.3]). Die WER gibt den prozentualen Anteil an Fehlern im erkannten Text im Vergleich zum Referenztext an. Es existieren drei Arten von Fehlern bei der Transkription (siehe nachfolgendes Beispiel). Wenn das System ein gesprochenes Wort nicht ausgibt, ist dies eine Löschung (*Deletion*, D). Eine Einfügung (*Insertion*, I) liegt vor, wenn ein nicht gesprochenes Wort transkribiert wird und um eine Ersetzung (*Substitution*, S) handelt es sich, wenn ein Wort falsch erkannt wird.

```
Referenz: —— die Katze lief im Schnee
Erkannt:  und sie Katze lief —— Schnee
Fehler:     I   S             D
```

Zur Berechnung der WER ist es erforderlich, dass der erkannte Text und der Referenztext aufeinander ausgerichtet werden, so dass verglichen werden kann an welchen Stellen Unterschiede auftreten (*Maximum Substring Matching*).

$$\text{Word Error Rate} = \frac{S + D + I}{\text{Anzahl Wörter im Referenztext}} \quad (1)$$

Trotz der weiten Akzeptanz gibt es Kritik an der WER, da sie alle Wörter gleich wertet und keine Gewichtung des Kontextes vornimmt, bei der die informativen Wörter höher gewichtet werden als nicht-informative Wörter, wie beispielsweise Artikel. Da eine solche Gewichtung jedoch nicht trivial realisierbar ist und die Komplexität der Metrik dadurch deutlich erhöht wird, hat sie sich nicht durchgesetzt (vgl. [2, Kap. 1.3]). Weitere Metriken zur Bewertung der Worterkennung werden bei McCowan, Moore, Dines u. a. diskutiert [7].

Als Referenz für wissenschaftliche Arbeiten und als Grundlage für Trainingsdaten halten verschiedene Institutionen allgemein zugängliche Sprachdatenbanken vor. Für den deutschsprachigen Raum wird diese Aufgabe von mehreren Institutionen übernommen, die in der übergreifenden Organisation *Common Language Resources and Technology Infrastructure Deutschland*[1] (CLARIN-D) organisiert sind.

International werden wissenschaftliche Entwicklungen in der Spracherkennung üblicherweise in der englischen Sprache verglichen. Zu diesem Zweck werden die Testverfahren des amerikanischen *National Institute of Standards and Technology*[2] (NIST) verwendet. Tabelle I gibt einen Überblick über die Fehlerraten aktueller Systeme bei internationalen Standardtests. An den Namen der Tests und der Vokabulargröße ist bereits die Komplexität der Tests ersichtlich, beim Test *Conversational Telephone Speech* ist beispielsweise die

[1] CLARIN-D: http://clarin-d.net/

[2] NIST: http://nist.gov/itl/iad/mig/

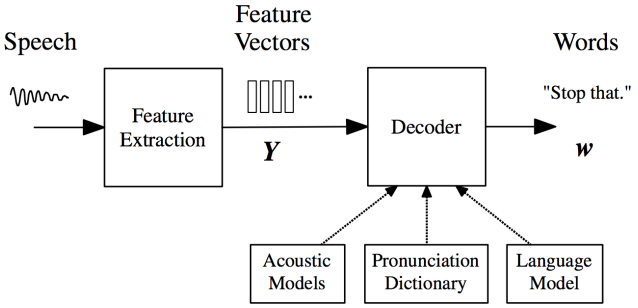

Abbildung 1. Aufbau eines Spracherkennungssystems: Aufteilung in einen Vorverarbeitungsschritt und die Erkennung im Dekoder mit Modellen [8, S. 201]

Transkription mitgeschnittener Telefongespräch durchzuführen, was aufgrund der zuvor beschriebenen Einflussfaktoren (siehe Abschnitt II-A) eine komplexe Aufgabe darstellt.

Der Unterschied zwischen den Aufgaben *Wall Street Journal* und den *Broadcast News* liegt neben der unterschiedlichen Vokabulargröße auch darin, dass im ersteren ein Text vorgelesen wird, der zur schriftlichen Kommunikation gedacht ist, wohingegen beim letzteren die Intention auf der akustischen Übermittlung der Informationen liegt. Diese Unterschiede haben Auswirkungen auf die Textart und Satzstrukturen und spiegeln sich in der Komplexität des Spracherkennungsproblems wieder. Insgesamt ist deutlich zu erkennen, dass die Komplexität und damit verbundenen die Fehlerrate mit der Wörterbuchgröße zusammenhängt und das Systeme für kleine Vokabulare als robust angesehen werden können.

D. Aufbau eines Spracherkennungssystems

Konkrete Anwendungen zur Spracherkennung variieren je nach Anwendungsfall und Einsatzort, aber im allgemeinen sind die Anwendungen aufgebaut wie in Abbildung 1 dargestellt.

Im Spracherkennungsprozess wird die Schallwelle durch das Mikrofon digitalisiert, dann zunächst vorverarbeitet und an Ruhepunkten in einzelne Äußerungen (*Utterances*) aufgeteilt. Für jede einzelne Äußerung wird anschließend versucht die Phonem- oder Wortfolge zu erkennen, die am wahrscheinlichsten zur Aufnahme gehört. Aktuelle Spracherkennungsarchitekturen nutzen dazu ein statistisches Modell, um die Worte zu ermitteln, welche den eingegebenen Lauten am wahrscheinlichsten entsprechen. Für diesen Erkennungsprozess ist es notwendig aus dem Signal eine Menge numerischer Merkmale (Merkmalsvektor, *feature vector*) zu extrahieren mit denen der Wellenausschnitt beschrieben wird.

Die eigentliche Erkennung (auch *Dekodierung*) wird dann mit Hilfe der Merkmalsvektoren auf Basis eines akustischen und eines Sprachmodells durchgeführt. Das *akustische Modell*, in Zusammenarbeit mit einem *phonetischen Wörterbuch*, dient dazu den Merkmalsvektoren diejenigen Phoneme zuzuordnen, welche am wahrscheinlichsten gesprochen wurden. Im *Sprachmodell* sind letztendlich die Wahrscheinlichkeiten hinterlegt, mit denen die ermittelten Phonemkombinationen ein bestimmtes Wort bilden.

Im Folgenden werden die einzelnen Bestandteile eines Spracherkennungssystems, sowie die einzelnen Modelle genauer beschrieben und jeweils verbreitete Umsetzungsmöglichkeiten vorgestellt.

E. Merkmalsextraktion

In diesem Abschnitt wird ein Überblick über die Schritte der Vorverarbeitung und der Merkmalsextraktion gegeben, in denen die aufgenommene Schallwelle in Merkmalsvektoren überführt wird. Als Ausgangsbasis für den Spracherkennungsprozess wird die akustische, analoge Wellenform der gesprochenen Sprache angenommen. Das Ziel der Vorverarbeitung ist es aus dieser Wellenform eine Sequenz von Merkmalsvektoren, mit möglichst geringer Dimension, zu extrahieren. Diese Ableitung der Merkmalsvektoren wird *speech feature extraction*, *acoustic preprocessing* oder *front-end processing* genannt (vgl. [2, S. 135]).

Die Schallwelle ist eingangs kontinuierlich und wird zunächst an den Pausen in grobe Teilstücke zerlegt. Da diese Teilstücke zum einen unterschiedlich lang und zum anderen zu lang für eine praktikable Verarbeitung sind, werden sie weiter in Fenster mit fester Länge aufgeteilt. Diese Darstellung betrachtet das Sprachsignal als eine Aneinanderreihung kurzer Blöcke mit konstanten Eigenschaften (vgl. [4, Kap. 4.2.1]).

Die Aufteilung in Fenster mit fester, kurzer Länge folgt daher, dass Sprache über einen kurzen Zeitraum von 5 bis 25 ms als statisch angesehen werden kann, das bedeutet es liegt ein fester Bereich vor, der analysiert werden kann (*Short-Time Spectral Analysis*, vgl. [2, S. 136]).

Damit in dieser Analyse auch schnelle Veränderungen im Sprachsignal berücksichtigt werden, beginnt alle 5 ms ein neues Fenster, wodurch sich die einzelnen Fenster überlappen. Dies führt dazu, dass letztendlich bis zu 100 Fenster pro Sekunde entstehen, in denen auch die schnellen Veränderungen berücksichtigt werden.

Des Weiteren kommt es, durch die zeitliche Aufteilung, an den Grenzen zu Signalsprüngen, was in der folgenden Frequenzanalyse zu Verfälschungen des Ergebnisses führt. Um dies zu vermeiden, werden die Fenster über eine Fensterfunktion (*window function*), üblicherweise die *Hamming-Funktion*, an den Grenzen gedämpft (vgl. [2, Kap. 5.1.1]). Durch die Überlappung benachbarter Fenster gehen keine Informationen verloren, aber die Erkennungsrate verbessert sich (vgl. [1, Kap. 9.3.2]).

Anschließend erfolgt als wichtiger Schritt der Vorverarbeitung, mit wesentlichem Einfluss auf der Endergebnis, die Extraktion von numerischen Merkmalen. Diese Merkmalsextraktion überführt einen der Blöcke in einen Merkmalsvektor, der dessen Frequenzspektrum und die jeweilige Energie kompakt darstellt. Zur Darstellung der Merkmalsvektoren existieren verschiedene Formate. Weit verbreitet ist die Darstellung als *Mel Frequency Cepstral Coefficients* (MFCC), welche im Folgenden vorgestellt wird. Die Erläuterungen orientieren sich an Jurafsky und Martin [1, Kap. 9.3], tiefergehende Details sind bei Logan [9] beschrieben.

Aufgrund der Einteilung der Blöcke nach festen zeitlichen Grenzen lassen sich die Sprachsignale zum Vergleich nicht genau übereinander legen, weshalb eine Analyse im Zeitbereich nicht möglich ist. Stattdessen werden die Signale im Frequenzbereich analysiert. Der Beitrag jeder Frequenzkomponente, die *spektrale Information*, wird dann als Kenngröße verwendet. Die Ermittlung dieser Kenngrößen erfolgt per *diskreter Fouriertransformation*. Mit dem Ergebnis liegen dann

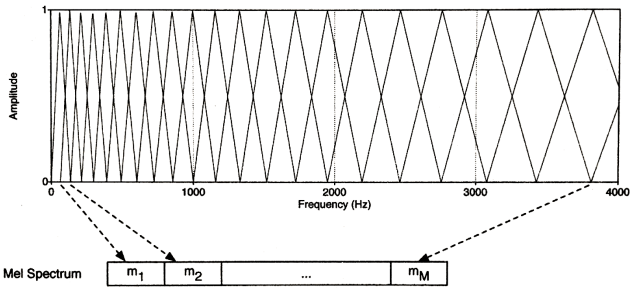

Abbildung 2. Die Mel-Filter-Bank nach Davis und Mermelstein (1980). Jeder dreieckige Filter sammelt die Energie eines Frequenzbandes. Unterhalb von 1000 Hz sind sie linear, oberhalb davon logarithmisch verteilt [1, S. 334]

Informationen über die Energiemenge pro Frequenzband vor. Das menschliche Gehör nimmt aber nicht alle Frequenzbänder gleich sensibel wahr, es ist weniger sensibel für hohe Frequenzen ab circa 1000 Hz (vgl. [1, Kap. 9.3.4]). Die Berücksichtigung dieser variierenden Sensibilität bei der Merkmalsextraktion erhöht die Performanz von Spracherkennungssystemen. Dies geschieht indem das Ergebnis der Fouriertransformation auf die *Mel-Skala* übertragen wird.

Ein *Mel* ist die Einheit für die wahrgenommene Tonhöhe. Sie ist so definiert, dass zwei Töne, die als gleich weit voneinander entfernt wahrgenommen werden, den gleichen Toneinheitswert erhalten. Für Frequenzen unterhalb von 1000 Hz ist das Verhältnis zwischen Frequenz und Mel-Skala linear, oberhalb davon ist es logarithmisch.

Während der MFCC-Berechnung wird die Umwandlung in Mel-Werte über eine Reihe von Filtern (*Mel Filter Bank*) erreicht, welche jeweils die Energie einer bestimmten Frequenz übertragen. Die Arbeitsweise einer Mel Filter Bank ist in Abbildung 2 dargestellt. Abschließend wird der Logarithmus der gesammelten Mel-Werte gebildet, da das menschliche Gehör ebenfalls logarithmisch auf unterschiedliche Signale reagiert. Dies bedeutet Unterschiede bei hohen Amplituden werden weniger ausgeprägt wahrgenommen als Unterschiede bei niedrigen Amplituden. Die Verwendung der Log-Werte ergibt ein robusteres Ergebnis, sodass das System insgesamt weniger anfällig für Variationen der Eingabedaten ist. Solche Variationen können zum Beispiel auftreten, wenn der Sprecher die Distanz zum Mikrofon minimal variiert.

Die extrahierten Mel-Werte könnten bereits zur Repräsentation der Merkmale verwendet werden, die Erkennungsrate des Spracherkennungssystems profitiert jedoch von weiteren Optimierungen der Merkmale. Daher wird im nächsten Schritt der MFCC-Berechnung das *Cepstrum* (der Name folgt aus Umkehrung der ersten vier Zeichen des Wortes *spectrum*) berechnet. Zum Verständnis des Cepstrums ist es hilfreich, zu vergegenwärtigen, dass die Schallwellen der Sprache im Körper an einer zentralen Stelle erzeugt werden. Die Form der Phone und der eigentlichen Sprache kommt aber durch die Form des Vokaltrakts von Lunge bis zu den Lippen zustande. Dieser kann aktiv geformt werden und dient als Filter (*Quelle-Filter-Modell*, vgl. [4, Kap. 2.1])

Die eigentliche Quelle des Schalls ist für die Unterscheidung von Phonen weniger wichtig als die Filter, die eine Veränderung an der Welle vorgenommen haben. Für die Merkmale ist es demnach hilfreich, eine Möglichkeit zu haben Quelle und Filter zu trennen und nur die Filterinformationen weiterzuverwenden. Das Cepstrum bietet diese Möglichkeit. Mathematisch ist es definiert als die inverse Fouriertransformation des Logarithmus der Fouriertransformation des Signals (vgl. [1, Kap. 9.3.5]).

Über das Ergebnis der Berechnung ist zum einen bekannt, dass die Varianz der einzelnen Merkmale unabhängig voneinander ist. Dies vereinfacht das akustische Modell, welches die Merkmale verarbeitet, weil weniger Parameter notwendig sind, denn die Kovarianz zwischen den MFCC-Merkmalen muss nicht repräsentiert werden. Außerdem sind die ersten Merkmalswerte ausreichend, um die Phoneme zu bestimmen. Die übrigen Merkmalswerte sind nur relevant um die Tonhöhe zu bestimmen, was aber bei der Sprachtranskription nicht erforderlich ist. Zur Berechnung der MFCC werden die ersten 12 Cepstralwerte verwendet, die separiert die Informationen über die Form des Vokaltrakts repräsentieren.

Nach den vorherigen Schritten liegen an dieser Stelle zu jedem Block 12 Cepstralkoeffizienten vor. Zusätzlich wird jeweils die Gesamtenergie des Blocks als 13. Merkmal ergänzt, da diese mit dem gesprochenen Phonem korreliert und somit eine zusätzliche Information liefert. Unter anderem haben Zischlaute und Vokale eine höhere Energie als Verschlusslaute (beispielsweise das *t* in „Lust“, vgl. [1, Kap. 9.3.6]).

Abschließend wird bei der Merkmalsextraktion berücksichtigt, dass das Sprachsignal nicht über alle gebildeten Fenster konstant ist, sondern dass es sich dynamisch verhält und verändert, z. B. an Übergängen und Verschlusslauten. Diese Änderungen spiegeln sich ebenfalls in der Veränderung der Cepstralkoeffizienten zwischen benachbarten Blöcken wieder. Es werden deshalb weitere Merkmale aus den Veränderungen der Koeffizienten (*delta* oder *velocity*) erzeugt. Zu jedem der 13 Merkmale wird je ein *Delta-Merkmal* (auch *Geschwindigkeitsmerkmal*) und ein *Delta-Delta-Merkmal* (auch *Beschleunigungsmerkmal*) gebildet. Das Delta-Merkmal repräsentiert die Änderung der Cepstralwerte zwischen zwei Blöcken, das Delta-Delta-Merkmal repräsentiert wiederum die Änderung der Delta-Werte zwischen zwei Blöcken.

Letztendlich besteht ein MFCC-Vektor aus 39 Merkmalen, welche die Sprachinformation eines Blockes, sowie die Veränderung gegenüber dem vorherigen Block, beinhalten. Mit diesen Merkmalsvektoren kann, anhand des akustischen Modells, das am wahrscheinlichsten repräsentierte Phonem bestimmt werden.

Bevor diese Phonembestimmung genauer erläutert wird, werden zunächst zwei Arten von Modellen eingeführt, die auch in aktuellen Spracherkennungssystemen Verwendung finden.

F. Repräsentation von Modellen

1) Hidden Markov Modelle: Hidden Markov Modelle (HMM), basierend auf Markov-Ketten, vereinen die Variabilität von Sprachsignalen (und damit verbunden auch der Merkmalsvektoren) und die Struktur der gesprochenen Sprache in einem stochastischen Modell. Da Sprache variabel ist - abhängig z. B. von der Betonung und dem Akzent des Sprechers - unterscheiden sich Sprachaufnahmen von unterschiedlichen Sprechern, auch wenn diese Aufnahmen im Text - ihrer linguistischen Struktur - gleich sind. Diese zugrundeliegende Struktur wird durch das HMM repräsentiert, gemeinsam mit

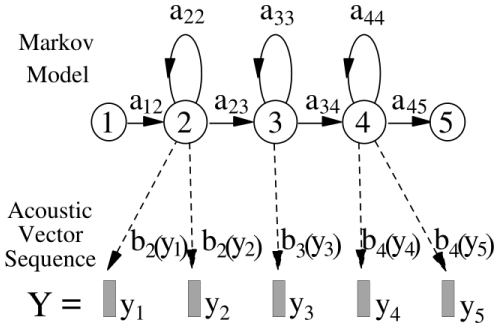

Abbildung 3. HMM-basiertes akustisches Modell zur Erkennung von Phonen [10, S. 204]

einer Menge von Wahrscheinlichkeiten, welche die Variabilität der tatsächlichen Sprachaufnahme abbilden.

Ein HMM repräsentiert die Wahrscheinlichkeitsverteilung über eine Reihe von Beobachtungen, wie es beispielhaft in Abbildung 3 dargestellt ist. Die Beobachtung zum Zeitpunkt t wird durch die Variable Y_t beschrieben, wobei die Zeitpunkte t gleichmäßig voneinander entfernt und die möglichen Beobachtungswerte diskretisiert sind. Eine charakterisierende Eigenschaft der HMM ist, dass jede Beobachtung zum Zeitpunkt t aus einem Prozess folgt, dessen innerer Zustand versteckt ist und nicht direkt beobachtet werden kann - das HMM ist Zeit-invariant. Die zweite charakterisierende Eigenschaft ist, dass die Markov-Eigenschaft erfüllt ist. Das bedeutet, die Übergangswahrscheinlichkeit zum nächsten Zustand hängt nur vom aktuellen Zustand ab und die vorhergehenden Zustände werden nicht berücksichtigt.

Für die Spracherkennung ist dies so zu verstehen, dass die Merkmalsvektoren die Beobachtungen darstellen und die internen Zustände den wahrscheinlichen Sprachinhalt. Da unterschiedliche interne Zustände (unterschiedliche Wortbestandteile) die gleiche Beobachtung (den gleichen Laut) hervorrufen können, ist es außerdem erforderlich, nicht nur eine einzelne Beobachtung, sondern eine Folge von Beobachtungen zu berücksichtigen. Über das zugrundeliegende Sprachmodell kann in diesem Fall ermittelt werden, welche Kombination von internen Zuständen am wahrscheinlichsten die Folge von Beobachtungen hervorruft.

Um ein HMM erfolgreich in Spracherkennungssystemen einsetzen zu können, sind zwei Fragestellungen zu lösen (vgl. [11]). Zum einen das zuvor erläuterte HMM-Erkennungsproblem: Gegeben eine Sequenz $y_1 : k$ akustischer Beobachtungen, wie kann die Zustandsfolge $x_1 : k$ bestimmt werden, die am wahrscheinlichsten $y_1 : k$ erzeugt hat?

Zum anderen das HMM-Trainingsproblem: Gegeben eine Sequenz $w_1 : k_w$ von Wörtern und eine Sequenz $y_1 : k$ akustischer Beobachtungen, wie können die Parameter so gewählt werden, dass den Beobachtungen später wieder die korrekten Zustände zugeordnet werden?

Zum Training des HMM werden Aufnahmen bekannter Texte benötigt, außerdem muss die Parametrisierung festgelegt werden (siehe [2, Kap. 8] und [1, Kap. 9.7]). Dies kann effizient mittels des Baum-Welch-Algorithmus erfolgen, der bei Gales und Young beschrieben ist (siehe [8, Kap. 4.2]).

Bei der Verwendung des HMM ist es wichtig, die Serie an Zuständen mit der höchsten Wahrscheinlichkeit effizient bestimmen zu können, da dies das Ergebnis der Spracherkennung darstellt. Der naive Ansatz zur Bestimmung dieser Folge ist eine Brute Force-Suche über alle möglichen Zustände, was mit einer hohen Komplexität verbunden und somit nicht praktikabel ist. Stattdessen wird das Problem mittels dynamischer Programmierung und dem *Viterbi*-Algorithmus gelöst, womit auf effiziente Weise die beste verborgene Pfadsequenz durch ein HMM gefunden wird. Dies geschieht, indem ein Suchbaum über alle möglichen Zustände gebildet und per Breitensuche durchlaufen wird, was trotzdem rechen- und speicheraufwändig ist.

In konkreten Implementationen des Algorithmus werden daher, zur Performanceoptimierung, alternative Ansätze umgesetzt. Zu diesen Ansätzen gehört es den Suchbaum dynamisch, während der Suche, zu expandieren oder parallel mehrere Pfadsequenzen zu suchen (vgl. [8, S. 213]).

Detaillierte Einführungen in HMM wurden von Rabiner [11] und Ghahramani [12] veröffentlicht.

2) Neuronale Netze: Tiefe neuronale Netze (*deep neural networks*, eine Form neuronaler Netze) sind ein moderner Ansatz, um die Erkennung von Phonemen und Wörtern durchzuführen. Durch die Weiterentwicklung der Methodik der neuronalen Netze und die höhere Leistungsfähigkeit aktueller Systeme, ist die Anwendbarkeit dieser Systeme gestiegen und erzielt vergleichsweise hohe Erkennungsraten. Eine Einsatzmöglichkeit in der Spracherkennung ist die Kombination mit oder der Ersatz von Hidden Markov Modellen zur Dekodierung der Merkmalsvektoren in Phoneme, wobei neuronale Netze, aufgrund ihrer Methodik, bessere Ergebnisse bei der Modellierung von kontextabhängigen Schwankungen in der Betonung von Worten versprechen.

Neuronale Netze (*Artificial Neural Networks*, ANN) wurden bereits 1988 zur Spracherkennung eingesetzt, konnten aber keine vergleichbaren Ergebnisse zu HMM erzielen, unter anderem aufgrund der Schwierigkeit in klassischen neuronalen Netzen zeitliche Abfolgen zu modellieren (vgl. [5, S. 15]). In den folgenden Jahren konnten durch die Anwendung von *Time Delayed Neural Networks* (vgl. [13]) und *Recurrent Neural Networks* (vgl. [14]) die Erkennungsraten verbessert werden, aber sie konnten die Ergebnisse der HMM-basierten Systeme nicht übertreffen, da es nicht möglich war neuronale Netze mit einer großen Tiefe effizient einzusetzen. Darüber hinaus haben HMM den Vorteil, dass sie mit weniger Aufwand auf große Trainingskorpora trainiert werden können.

Durch die Möglichkeit tiefere neuronale Netze einzusetzen (sogenannte *Deep Neural Networks* (DNN) zeichnen sich dadurch aus, dass zwischen der Ein- und Ausgabeschicht eine große Anzahl an versteckten Schichten liegt) wurden die Erkennungsraten verbessert (vgl. [15]). Das Training per *Backpropagation* unterscheidet sich allerdings nicht zu den bisherigen ANNs, weshalb das Training weiterhin aufwändig war (vgl. [5]).

Ein Fortschritt wurde erzielt als Hinton, Osindero und Teh in 2006 ein effizientes Lernverfahren für *Deep Belief Networks* (DBN) vorstellte (vgl. [16]). Bei einem DBN wird jede Schicht isoliert vorab trainiert, was sehr effizient möglich ist. Außerdem war es durch den Einsatz von GPU-Berechnungen möglich, die Trainingsverfahren weiter zu beschleunigen. Traditionelle Trainingsverfahren werden anschließend nur noch

zur Optimierung des Trainingsstandes verwendet. Durch diese Entwicklung ist es möglich Deep Belief Networks auf Basis neuronaler Netze mit großer Tiefe für Spracherkennungssysteme einzusetzen. Dies geschieht allerdings weiterhin häufig in Kombination mit HMM als hybrides Verfahren, bezeichnet als *context-dependent deep neural network hidden Markov model*. Mittels dieser Kombination ist eine Verbesserung der Erkennungsrate um bis zu 9,2% gegenüber traditionellen Ansätzen möglich (vgl. [17]).

G. Akustisches Modell

Die Aufgabe des akustischen Modells ist es, die wahrscheinlichsten Phone auf Basis der zuvor abgeleiteten Merkmalsvektoren zu bestimmen. Wie zuvor beschrieben ist jedes Wort aus einer Sequenz von Phonemen zusammengesetzt, die wiederum durch unterschiedliche Phone abgebildet werden. Dies heißt, dass im akustischen Modell unterschiedliche Betonungen berücksichtigt werden müssen.

Zur Bildung des akustischen Modells werden üblicherweise Hidden Markov Modelle eingesetzt. In diesen Modellen entspricht jeder einzelne akustische Merkmalsvektor einem spezifischen Zustand, wie dargestellt in Abbildung 3 und Abschnitt II-F1. Dies bedeutet eine Sequenz von n Merkmalsvektoren entspricht n aufeinanderfolgenden Zuständen. Darüber hinaus werden in den Modellen Null-Zustände modelliert, denen als akustische Beobachtung Stille zugeordnet ist. Für die Modellierung der HMM wird angenommen, dass jede Sequenz in einem Null-Zustand beginnt und endet.

Als Teilkomponente des akustischen Modells kommt dem phonetischen Wörterbuch (auch Wortlexikon, *Phonetic Dictionary*) eine besondere Bedeutung zu. Es spezifiziert, welche Wörter durch das Spracherkennungssystem erkannt werden können und aus welchen Phonemen diese bestehen. Zu jedem Wort können mehrere Zusammensetzungen hinterlegt sein, um die unterschiedlichen Arten der Aussprache abzubilden. Ein phonetisches Wörterbuch kann im einfachsten Fall eine Auflistung der Wörter und seiner Bestandteile sein, es kann aber auch als HMM umgesetzt werden, dann wird es als *Phonemmodell* bezeichnet (vgl. [4, S. 101]).

Nachfolgend ist ein beispielhaftes phonetisches Wörterbuch mit zwei Wörtern dargestellt. Das Wort „BLINDS“ besteht aus 6 Phonemen, es liegt nur eine Sprechweise vor. Für das Wort „CLOSE“ liegen zwei Sprechweisen vor, die sich im letzten Phonem unterscheiden. Da viele Phoneme bereits auf unterschiedliche Weisen ausgesprochen werden können (siehe Abschnitt II-B), wird bereits durch diese einfache Abbildung ein breites Spektrum an Aussprachen berücksichtigt.

```
BLINDS     B L AY N D Z
CLOSE      K L OW S
CLOSE(2)   K L OW Z
```

H. Sprachmodell

Das Sprachmodell gibt die Wahrscheinlichkeit an, mit der eine Sequenz von Wörtern einen Satz der zugehörigen Sprache bildet (vgl. [1, Kap. 9.5]). Die Aufgabe des Sprachmodells ist die Repräsentation der Grammatik einer Sprache. Sprachmodelle können auf mehrere Arten repräsentiert werden, dazu gehören unter anderem N-Gramm-Sprachmodelle und kontextfreie Grammatiken.

Im Allgemeinen sind N-Gramm-Modelle probabilistische Modelle zur Vorhersage von Nachfolgeelementen innerhalb einer definierten Sequenz (vgl. [1, Kap. 4]). Für die Spracherkennung hilft dies zur Vorhersage des nächsten Wortes in einem Satz.

Für die Vorhersage wird nicht nur die Wahrscheinlichkeit eines einzelnen Wortes bestimmt, sondern die Wahrscheinlichkeit des Vorhandenseins einer bestimmten Folge von Wörtern, inklusive des Ausgangswortes. Ein N-Gramm ist hier eine feste Sequenz aus N Worten oder Wortfragmenten, die als Einheit der Vorhersage dient. Bestandteile dieser Sequenz können, je nach Anwendungsfall, nicht nur vollständige Worte, sondern auch Wortbestandteile sein. Ein 2-Gramm (auch *Bigramm*) besteht aus zwei Elementen, z. B. „bitte wenden“ auf Wortbasis, ein 3-Gramm (auch *Trigramm* genannt) aus drei Elementen. Ein gültiges Trigramm ist zum Beispiel „BRS“, basierend auf Buchstaben.

Die Wahl der Größe N für das N-Gramm-Modell ist bereits ein Optimierungsparameter. 1-Gramm- oder 2-Gramm-Modelle haben eine relativ niedrige Erkennungsrate, insbesondere bei einem großen Vokabular, Trigramm-Modelle benötigen aber mehr Speicherplatz, was auf eingebetteten Systemen ein Problem ist. In der Praxis werden in leistungsfähigen Systemen oftmals auch 5-Gramm- und 4-Gramm-Modelle eingesetzt.

Jedem N-Gramm innerhalb eines Modells wird eine bedingte Wahrscheinlichkeit, hinsichtlich benachbarter Elemente, zugeordnet. Damit diese Wahrscheinlichkeiten gebildet werden können, ist es erforderlich einen Gesamtkorpus möglicher Elemente zu haben, der die Grundlage für das N-Gramm-Modell bildet. Die Bildung des N-Gramm-Modells erfolgt dann dermaßen, dass alle N-Tupel innerhalb des Korpus gezählt und die relative Wahrscheinlichkeit jedes Tupels berechnet und auf einen Wert zwischen 0 und 1 normalisiert wird. Dies wird *Maximum Likelihood Estimation* genannt (vgl. [1, S. 122]) Jeder Kombination aus N Elementen ist damit eine Wahrscheinlichkeit zugeordnet, mit der sie in einer Sequenz vorkommt.

In der Praxis werden bei der Bildung eines N-Gramm-Modells häufig auch die kleineren (N-1)-Gramme, (N-2)-Gramme usw. gebildet. Falls die aufzufindende Sequenz im größten Modell nicht gefunden wird, erfolgt die Suche dann im nächstkleineren Modell. Da die Anzahl möglicher Kombinationen des N-Gramm-Modells im Verhältnis zu N exponentiell wächst (ein Trigramm-Modell mit 50 Phonemen hat $50^3 = 125,000$ Kombinationen) werden als Datenstruktur häufig Entscheidungsbäume eingesetzt, um das Modell zu strukturieren und die Suche zu vereinfachen (vgl. [10, S. 8f.]).

Für eine detaillierte Beschreibung der N-Gramm-Modelle und der Grundlagen der Trainingsverfahren sei auf Jurafsky und Martin [1, Kap. 4] verwiesen.

I. Zusammenfassung

In den vorherigen Abschnitten wurden die einzelnen Bestandteile eines Spracherkennungssystems vorgestellt. Wesentliche Bedeutung liegt dabei auf den verschiedenen Model-

Tabelle II. SPRACHERKENNUNGSKOMPONENTEN MIT OPEN SOURCE-LIZENZIERUNG

Name	Entwicklung	Sprachmodell	Akustikmodell	Lizenz	Quellen
Sphinx Familie	1986 Carnegie Mellon Univ.	Hidden-Markov (CMU)	ARPA N-Gram	BSD	[18][19]
Julius	1997 Nagoya Institute of Technology	Hidden-Markov (HTK)	ARPA N-Gram	Julius Lizenz	[20]
Kaldi	2011 Johns Hopkins University	Hidden-Markov / Deep Learning	FST / Recurrent Neural Network	Apache 2.0	[21]
RWTH ASR	2001 RWTH Aachen	Hidden-Markov / Deep Learning	N-Gram	RASR Lizenz auf QPL Basis	[22][23]

len, in denen die bekannten Sprachen und Sprachbestandteile abgebildet sind und welche die wesentliche Qualitätsmerkmale leistungsfähiger Spracherkennungssysteme darstellen. Die zugrundeliegenden Algorithmen, Datenstrukturen und -repräsentationen basieren auf langjähriger Forschung. Sie sind ausgereift und in Anwendungsgebieten mit kleinem und mittlerem Vokabular erprobt.

Aktuelle Forschungen beschäftigen sich mit der Entwicklung von Verfahren für Spracherkennung mit großem Vokabular. Hier sind insbesondere der Einsatz tiefer neuronaler Netze in Kombination mit oder als Ersatz für HMM zu nennen, welche durch jüngste Forschungsergebnisse und steigende Rechenleistungen möglich wurden.

III. PRAKTISCHE ANWENDUNG

Im Folgenden werden frei verfügbare Front- und Backends zur Spracherkennung, sowie Werkzeuge zur Handhabung von Sprachmodellen vorgestellt.

A. Komponenten

Je nach Anwendungsfall werden Aufnahmegerät und Hardwareplattform individuell ausgewählt. Das Mikrofon bestimmt Parameter wie Sampling- und Bitrate der Tonaufnahme. In Desktopumgebungen sind Bitraten von 16 Bit und Samplingraten von 16 kHz üblich, bei Telefonverbindungen werden 8 Bit und 8 kHz verwendet. Diese Werte wirken sich auf das akustische Modell aus, da je nach gewählten Werten unterschiedlich viele Daten zur Erkennung vorliegen. Weitere Einflussfaktoren durch das Mikrofon auf die Erkennungsrate sind Rauschen in der Aufnahme, Umgebungsgeräusche, sowie die Distanz des Sprechers zum Mikrofon.

Die vom Mikrofon aufgenommene Sprache wird zur Nutzerinteraktion semantisch ausgewertet. Dazu wird die aufgenommene Sprache an ein Backend, entweder lokal oder auf einem Server, übertragen. Anhand des transkribierten Textes werden anschließend Aktionen ausgeführt (z. B. ein Termin im Kalender erstellt).

Auf Mobilgeräten ist *Apple Siri* ein exemplarisches Frontend, welches auf Befehle reagiert und den Dialog mit dem Benutzer führt. Analog arbeiten Command-and-Control- (C&C) und Diktiersysteme, welche entweder mit kurzen Befehlen arbeiten oder den diktierten Text in ein Dokument schreiben. Besonders bei Open Source-Systemen sind die Frontends häufig erweiterbar, so dass sie auf Stichwörter oder -phrasen reagieren und anschließend benutzerdefinierte Aktionen, beispielsweise zur Heimautomatisierung, ausführen. Beispiele für Frontends sind *Jasper*[3] und *Simon*[4].

[3]Jasper: http://jasperproject.github.io/

[4]Simon: https://simon.kde.org/

Im Backend findet die eigentliche Dekodierung von Sprachaufnahmen statt. Dabei können unterschiedliche Algorithmen zur Vorverarbeitung und Transkription verwendet werden. Die verbreiteten Softwarepakete unterscheiden sich in den verwendeten Algorithmen, bei der benötigten Rechenkapazität als auch beim Format der Modelle.

Lokal ausführbare Backends werden in Tabelle II[5][6][7][8] aufgeführt. Diese Komponenten werden von Forschungseinrichtungen als Grundlage für Forschungsarbeiten entwickelt. Zur Verwendung sind neben der Software die passenden Modelle nötig, auf deren Verwendung im Abschnitt III-B eingegangen wird. Die einzelnen Backends unterscheiden sich durch die implementierten Algorithmen, sind aber durch ihren modularen Aufbau verhältnismäßig einfach erweiterbar. Eine Verwendung in kommerziellen oder Endanwenderprodukten ist möglich, wobei dies in jedem Fall mit Entwicklungsaufwand verbunden ist und Kenntnisse im Bereich der Spracherkennung notwendig sind, um ein performantes und zuverlässiges System zu entwickeln. Bei kommerzieller Verwendung muss des Weiteren die Lizenz beachtet werden.

Cloud-basierte Spracherkennung wird durch Dienste von Anbietern wie Google, AT&T oder wit.ai ermöglicht, deren kommerzielle Systeme über Jahre entwickelt und optimiert wurden. Die Sprachaufnahme wird an den jeweiligen Webservice geschickt, welcher den transkribierten Text zurücksendet. Je nach Auslastung des Servers kann die Antwortzeit allerdings variieren. Zusätzlich ist eine Registrierung beim Anbieter erforderlich. Die Menge der Transkriptionen ist bei den kostenlosen Angeboten auf Tagesbasis begrenzt. Eine Übertragung an den Anbieter impliziert, dass sämtliche gesprochene Sprache an einen fremden Server übertragen und dort ausgewertet wird. Daher entsteht die Notwendigkeit einer ständigen Internetverbindung, sowie unter Umständen Datenschutzbedenken.

Im Gegensatz zu lokalen System kann die cloud-basierte Spracherkennung nicht trainiert werden und kennt domänenspezifische Fachwörter häufig nicht. Die Erkennung alltäglicher Sprache ist allerdings, durch die großen Mengen an Sprachsamples, die jeder Benutzer dieser Systeme beisteuert, genauer als bei eigenen Modellen.

B. Modelle

Im Folgenden wird die Beschaffung und Handhabung von Modellen für praktische Spracherkennungskomponenten beschrieben. Modelle werden auf Basis eines Sprachkorpus und dazugehöriger Trainingsdaten erstellt. Je nach Spracherkennungssoftware unterscheiden sich die einzelnen Formate und sind nicht immer kompatibel. Neben den Werkzeugen

[5]CMU Sphinx: http://cmusphinx.sourceforge.net/

[6]Julius: http://julius.osdn.jp/

[7]Kaldi: http://kaldi.sourceforge.net/

[8]RWTH ASR: http://www-i6.informatik.rwth-aachen.de/rwth-asr/

der jeweiligen Forschungseinrichtung, welche auf das jeweilige Spracherkennungskomponente abgestimmt sind, existieren Tools für häufig verwendete Formate (wie z.B. *ARPA N-Gram*[9]).

Das *VoxForge*-Projekt[10] veröffentlicht freie Akustikmodelle für Julius, Sphinx und HTK. Die Qualität der Modelle bemisst sich hauptsächlich nach der Menge der freiwillig zur Verfügung gestellten Daten. Insgesamt existieren Modelle für 18 verschiedene Sprachen. Ausgangspunkt für das Projekt, ist das Problem, dass die Anwendungen zwar Sprachmodelle mitbringen, aber keine Information darüber verfügbar ist, welche Wörter trainiert wurden. Die Korpora und Trainigsdaten werden meist unter einer restriktiven Lizenz eingekauft und können daher nicht veröffentlicht werden.

Die Qualität der Sprachmodelle ist von zentraler Bedeutung für die Qualität der Spracherkennung. Das trainieren dieser Modelle benötigt viel Sprachmaterial und erfahrene Arbeitskräfte um ein hochwertiges Modell zu erstellen (vgl. Vertanen[24]). Die Modelle auf VoxForge sind in der Regel von geringer Qualität, da nicht genügend Sprachbeispiele zur Verfügung stehen und Sprecherunabhängige Modelle zu erstellen. Daher müssen diese Modelle meist für den jeweiligen Zweck nachtrainiert werden. Eine Einführung in die Adaption von Modellen findet sich in der Sphinx-Wiki[11] und .

Kommerzielle Spracherkennungssoftware kann, besonders bei Cloud-basierten Diensten wie Google Speech und Siri, auf große Datenmengen der Benutzer zurückgreifen um das Sprachmodell weiter zu verbessern.

IV. Praktische Erkenntnisse

Abschließend wird prototypisch eine C&C-Anwendung zur Verwendung in der Heimautomatisierung aufgebaut. Dazu wird zunächst die Software ausgewählt und eine Reihe von Befehlen definiert. Anschließend werden verschiedene Versuche durchgeführt, um einen Überblick über mögliche Fallstricke zu bekommen. Das Ziel des praktischen Versuchs ist es, Erkenntnisse über die Wahl eines Schlüsselwortes, mögliche false-positive Interpretationen durch Gespräche und generelle Fehlinterpretation bei der Systeminteraktion zu erhalten. Dazu wird ein vordefiniertes, sprecher-unabhängiges akustisches Modell und ein selbstdefinierter Befehlssatz, mit generiertem Sprachmodell, verwendet.

A. Auswahl der Software

Im Bereich der Heimautomatisierung wird zur Spracherkennung häufig auf die Software *Jasper* verwiesen. Diese kann auf einem Raspberry Pi ausgeführt werden und übernimmt die Nutzerinteraktion, indem es, abhängig von Schlüsselwörtern, Module und Aktionen aufruft (z.B. „Jasper, tell the time“ ruft ein Modul auf, das über eine Sprachausgabe die Uhrzeit ausgibt). Es können verschiedene Backends zur Sprachtranskription eingebunden werden. Dieses Transkript wird auf Schlüsselwörter untersucht und die jeweiligen Module aufgerufen. Eine Anbindung an Heimautomatisierungssysteme wie *FHEM* oder *OpenHAB* ist per Python-Programmierung und Standardschnittstellen (z. B. REST) möglich.

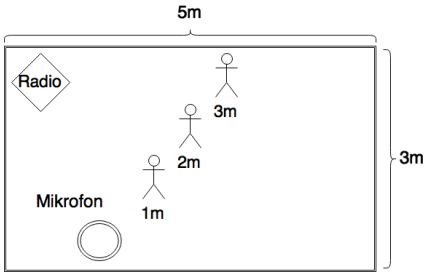

Abbildung 4. Versuchsaufbau, Platzierung von Mikrofon und Radio. Position des Sprechers bei verschiedenen Distanzen

Die Erkennungsrate mit Pocketsphinx 0.8 als Backend und dem mitgelieferten Sprachmodell ist stark von den Rahmenbedingungen abhängig. Je nach Raumakustik, Mikrofon, Sprecher und Distanz vom Mikrofon ist die Verwendung praktisch nicht möglich, da das Aktivierungswort „Jasper“ nur selten erkannt wird. Um diesem Problem zu begegnen wurden während der Softwareauswahl verschiedene Backends wie *Google STT* und *wit.ai*, sowie *Julius* und verschiedene frei verfügbare Sprachmodelle getestet.

Die eigenständige Verwendung der aktuellen Version von Pocketsphinx (Version *pocketsphinx-5prealpha*), unabhängig von Jasper, liefert hingegen deutlich bessere Erkennungsraten. Diese decken sich mit den bekannten Werten aus der Literatur (Word Error Rate von circa 10%, vgl. [19]). Der Grund für die geringen Erkennungsraten im Zusammenhang mit Jasper konnte im Rahmen der Softwareauswahl nicht ermittelt werden.

Aufgrund der Wahl des Raspberry Pi als Plattform und dem Anspruch, aus Gründen der Privatsphäre keine cloudbasierten Anbieter zu verwenden, reduziert sich die Auswahl der möglichen Backends auf Pocketsphinx und Julius, da diese beide auf dem Raspberry Pi lauffähig sind. Aufgrund der besseren Toolunterstützung, besserer Performanz und einer weniger restriktiven Lizenz wurde Pocketsphinx ausgewählt.

B. Versuchsaufbau

Pocketsphinx wird auf einem *Raspberry Pi B+* mit dem *Samson UB1* USB-Mikrofon (16 Bit, 48kHz, omnidirektional) ausgeführt. Das Mikrofon ist in einem normal eingerichteten Raum mit Teppichboden platziert. Die Platzierung des Mikrofons, eines Radios und der Position des Sprechers bei verschiedenen Entfernungen ist in Abbildung 4 schematisch dargestellt.

Das eingesetzte Sprachmodell wurde mit der Onlineversion des *CMU lmtool*[12] zur Erstellung von Sprachmodellen auf Basis eines selbstentworfenen Befehlssatzes für die Heimautomatisierung erstellt. Als akustisches Modell wird das von Pocketsphinx mitgelieferte englische Modell verwendet. Die Auswertung der transkribierten Texte erfolgt in einem separaten Python-Skript.

Mit der eingesetzten Hardware ist es notwendig, das Sprachmodell und das Wörterbuch zu begrenzen, um das

[9] Sphinx-Wiki N-Gram: http://cmusphinx.sourceforge.net/wiki/sphinx4:standardgrammarformats

[10] VoxForge: http://www.voxforge.org/

[11] Sphinx-Wiki Adaptation: http://cmusphinx.sourceforge.net/wiki/tutorialadapt

[12] CMU lmtool: http://www.speech.cs.cmu.edu/tools/lmtool.html

Tabelle III. DIE FÜNF HÄUFIGSTEN UND DIE DREI SELTENSTEN VORKOMMEN VON WÖRTERN IM OUT-OF-DICTIONARY GESPRÄCH (INSGESAMT 2119 WÖRTER).

the	on	time	tell	lights	kitchen	television	blinds
846	279	229	143	74	19	17	16

System echtzeitfähig zu halten. Im Folgenden wird daher ein Vokabular aus 32 Wörtern und 10 Befehlen verwendet. Für jeden Befehl existiert eine Lang- und eine Kurzform. Die Befehle sind in der Langform in Tabelle V aufgelistet. In der Kurzform entfällt jeweils ein Wort (z. B. ist „open window blinds“ in Kurzform „open blinds“).

C. Durchführung

1) False-positives bei Gesprächen: Da die Spracherkennung kontinuierlich auf Befehle und Aktivierungswörter hört, ist die Wahl des Aktivierungswortes von entscheidender Bedeutung um irrtümliche Befehle zu vermeiden. Während eines zweistündigen Gesprächs, in deutscher Sprache mit 8 beteiligten Personen in einem Labor der Hochschule, läuft das System mit und transkribiert konstant die gesprochene Sprache. Sämtliche erkannten Wörter können, da sie aus dem Englischen stammen, als fehlerhafte Transkriptionen angenommen werden. Dies basiert auf der Tatsache, dass keine Wörter erkannt werden, die nicht im Vokabular vorkommen, sondern immer das Wort mit der höchsten Wahrscheinlichkeit zurückgegeben wird. Zur Auswertung wird das Transkript auf gültige Befehle in Lang- und Kurzform durchsucht und die Wörter einzeln gezählt. Da keine Häufigkeitsverteilung der erkannten Worte bekannt ist, wird ein beliebiges Wort („Time“) aus dem Wörterbuch als Schlüsselwort festgelegt.

Die Anzahl der Transkriptionen einzelner Wörter sind in Tabelle III aufgelistet. Dabei ist zu erkennen, dass einfache und kurzsilbige Wörter wesentlich häufiger transkribiert werden als komplexe Wörter.

Das Ergebnis der Befehlserkennung ist in Tabelle IV dargestellt. Es zeigt sich, dass Befehle aus häufig erkannten Wörtern auch häufig im Transkript vorkommen. Der Einfluss der Grammatik zeigt sich hier bei „Television“, da fast alle Vorkommen des Wortes in Verbindung mit einem weiteren Befehlswort stehen.

Das Ergebnis zeigt, wie wichtig die Verwendung eines Aktivierungswortes ist, insofern das System Wörter, die nicht im Wörterbuch vorkommen, verwerfen kann. Ebenso ist die Verwendung eines *Keyphrase-Modus*, welcher nur auf einzelne Schlüsselwörter reagiert und anschließend in den kontinuierlichen Modus umschaltet, sinnvoll. Weiterhin ist es empfehlenswert, die Interaktion mit dem Gerät nicht mit einzelnen, kurzen Kommandos, sondern mit kurzen Sätzen durchzuführen. Dies erhöht einerseits den Aufwand für das Spracherkennungssystem, verringert andererseits aber das Risiko von False-Positives.

2) Einfluss von Distanz und Hintergrundgeräuschen: Um die Erkennungsrate bei verschiedenen Distanzen zu evaluieren wird das Mikrofon wie in Abbildung 4 platziert und Befehle in ihrer Langform aus unterschiedlichen Distanzen zum Mikrofon gesprochen. Dieser Vorgang wird sowohl bei eingeschaltetem Radio (circa 55 dB) als auch ohne Radio durchgeführt. Ein Befehl zählt dann als erkannt, wenn im erkannten Text der Befehl in der angegebenen Reihenfolge vorkommt. Dadurch werden Einsetzungsfehler am Anfang und am Ende des Strings ignoriert. Jeder Befehl wird zehnmal aus 1, 2 und 3 Metern Entfernung zum Mikrofon gesprochen.

Tabelle IV. DIE FÜNF HÄUFIGSTEN VORKOMMEN VON BEFEHLEN IM OUT-OF-DICTIONARY GESPRÄCH.

Befehl	Vorkommen	Vorkommen nach „Time“
lights on	29	4
tell the time	29	1
lights off	8	0
television on	7	0
television off	4	0

Tabelle V. ERKENNUNGSRATE UND WORD ERROR RATE IN PROZENT AUF VERSCHIEDENEN DISTANZEN. IN KLAMMERN: MIT RADIO.

Befehl	1 Meter	WER (1 m)	2 Meter	3 Meter
dim kitchen lights	30 (20)	10	40 (10)	60 (10)
turn lights on	10 (10)	53	30 (10)	30 (0)
turn lights off	10 (10)	50	0 (10)	0 (0)
open window blinds	100 (80)	0	60 (20)	60 (0)
close window blinds	100 (80)	0	60 (30)	60 (0)
open front door	100 (90)	0	100 (80)	100 (20)
close front door	100 (90)	0	100 (70)	100 (10)
tell the time	70 (80)	20	80 (60)	90 (10)
turn television on	70 (40)	16	30 (10)	40 (0)
turn television off	50 (30)	14	20 (10)	30(0)

Die Ergebnisse sind in Tabelle V aufgelistet. Dabei werden für jeden Befehl die Erkennungsraten in Prozent angegeben. In Klammern steht die Erkennungsrate bei eingeschaltetem Radio. Während des Versuchs bestätigte sich, dass die Aussprache einen wesentlichen Beitrag zur Erkennungsrate gibt. Die Befehle wurden nicht konstant gleich ausgesprochen, sondern variieren während des Versuchs.

Durch das eingeschaltete Radio wird nicht nur die Erkennung der Wörter beeinträchtigt, sondern es ist auch eine lautere Aussprache notwendig, da Pocketsphinx aufgrund der Hintergrundlautstärke den Schwellwert für die Erkennung anhebt. In einem Fall wurde ein Befehl mit Radio häufiger erkannt, was sich dadurch erklären lässt, dass die Wörter „tell the time“ häufiger transkribiert werden. Somit kann ein Geräusch des Radios zufällig die Transkription des korrekten Wortes verursacht haben.

Neben der Erkennungsrate von Befehlen wird die *Word Error Rate* (siehe Abschnitt II-C) bei einem Meter Distanz bestimmt. Dabei fällt auf, dass unter anderem bei „dim kitchen lights“ häufig „lights“ fälschlicherweise transkribiert wird.

V. ZUSAMMENFASSUNG

Spracherkennung ist breit erforscht und findet Verwendung in vielen Anwendungen. Für einfache Aufgaben mit kleinem Vokabular und wenigen Störeinflüssen können aktuelle Systeme als ausgereift und robust angesehen werden. Beispiele dafür finden sich im Alltag, sei es im Auto-Navigationsgerät oder beim Fernseher mit Sprachsteuerung. Eine Herausforderung besteht weiterhin bei der Transkription mit einem umfangreichen, natürlichsprachlichen Vokabular oder auch dann, wenn die Aufnahme nicht originär an das Spracherkennungssystem gerichtet ist.

Für die Entwicklung sprachgesteuerter Systeme stehen Anwendungskomponenten zur freien Verfügung. Dazu gehören

sowohl lokale Komponenten als auch cloud-basierte Dienste mit begrenztem, kostenfreien Zugriff. Die Einstiegshürde für Projekte, bei denen Spracherkennung als Werkzeug dient und nicht primär im Fokus der Entwicklung steht, ist relativ gering, wobei komplexe Spracherkennungslösungen trotzdem auch kritisch betrachtet werden sollten, wie das Beispiel Jasper gezeigt hat. Kenntnisse über die Funktionsweise und die Parametrisierung sind aber in jedem Fall notwendig, um eine zufriedenstellende Erkennungsrate zu erreichen.

Es wurde diskutiert, dass die Erkennungsrate im Wesentlichen von der Qualität des akustischen und des Sprachmodells abhängt. Frei verfügbare Modelle mit guter Qualität existieren für die englische Sprache, für andere Sprachen sind die Modelle weniger ausgereift. Dies macht es erforderlich, entweder ein eigenes Modell vollständig selbst zu trainieren oder bestehende Modelle zu adaptieren, was bei Verwendung von Sprachdatenbanken zwar möglich, aber aufwändig ist.

Für die eingangs gestellte Frage nach der Verwendbarkeit für die Heimautomatisierung zeigt sich, dass dies möglich und mit den existierenden Komponenten realisierbar ist. Ein wesentlicher Grund dafür ist, dass nur ein geringes Vokabular ohne sprecherabhängiges Training notwendig ist. Bei der Konzeption sollten allerdings die diskutierten Kriterien, wie die geschickte Wahl eines Schlüsselwortes, Berücksichtigung finden.

LITERATUR

[1] D. Jurafsky und J. H. Martin, *Speech and Language Processing*, 2nd. Pearson Education International, 2009.

[2] M. Wölfel und J. McDonough, *Distant Speech Recognition*. Chichester, UK: Wiley, 2009.

[3] L. L. Rabiner und B.-H. B. Juang, *Fundamentals of speech recognition*, 1993.

[4] S. Euler, *Grundkurs Spracherkennung*. Springer-Verlag, 2007.

[5] B. H. Juang und L. R. Rabiner, „Automatic speech recognition – a brief history of the technology development“, *Elsevier Encyclopedia of Language and Linguistics*, S. 1–24, 2005.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath u. a., „Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups“, *Signal Processing Magazine, IEEE*, Bd. 29, Nr. 6, S. 82–97, 2012.

[7] I. A. McCowan, D. Moore, J. Dines, D. Gatica-Perez, M. Flynn, P. Wellner und H. Bourlard, „On the use of information retrieval measures for speech recognition evaluation“, IDIAP, Techn. Ber., 2005.

[8] M. Gales und S. Young, „The application of hidden markov models in speech recognition“, *Foundations and trends in signal processing*, Bd. 1, Nr. 3, S. 195–304, 2008.

[9] B. Logan, „Mel frequency cepstral coefficients for music modeling“, in *Int. Symposium on Music Information Retrieval*, 2000.

[10] G. Zweig und M. Picheny, „Advances in large vocabulary continuous speech recognition“, *Advances in Computers*, Bd. 60, S. 249–291, 2004.

[11] L. Rabiner, „A tutorial on hidden markov models and selected applications in speech recognition“, *Proceedings of the IEEE*, Bd. 77, Nr. 2, S. 257–286, 1989.

[12] Z. Ghahramani, „An introduction to hidden markov models and bayesian networks“, *International Journal of Pattern Recognition and Artificial Intelligence*, Bd. 15, Nr. 01, S. 9–42, 2001.

[13] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano und K. Lang, „Phoneme recognition using time-delay neural networks“, *Acoustics, Speech and Signal Processing, IEEE Transactions on*, Bd. 37, Nr. 3, S. 328–339, März 1989.

[14] A. Robinson und F. Fallside, *Phoneme recognition from the TIMIT database using recurrent error propagation networks*. University of Cambridge, Department of Engineering, 1990.

[15] L. Deng und X. Li, „Machine learning paradigms for speech recognition: an overview“, *IEEE Transactions on Audio, Speech and Language Processing*, Bd. 21, Nr. 5, S. 1060–1089, 2013.

[16] G. Hinton, S. Osindero und Y. Teh, „A fast learning algorithm for deep belief nets“, *Neural Computation*, Bd. 18, Nr. 7, S. 1527–1554, Juli 2006.

[17] G. E. Dahl, D. Yu, L. Deng und A. Acero, „Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition“, *IEEE Transactions on Audio, Speech and Language Processing*, Bd. 20, Nr. 1, S. 30–42, 2012.

[18] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf und J. Woelfel, „Sphinx-4: A flexible open source framework for speech recognition“, 2004.

[19] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar und A. I. Rudnicky, „Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices“, in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, IEEE, Bd. 1, 2006.

[20] A. Lee, T. Kawahara und K. Shikano, „Julius—an open source real-time large vocabulary recognition engine“, 2001.

[21] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer und K. Vesely, „The kaldi speech recognition toolkit“, in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dez. 2011.

[22] D. Rybach, S. Hahn, P. Lehnen, D. Nolden, M. Sundermeyer, Z. Tüske, S. Wiesler, R. Schlüter und H. Ney, „Rasr-the rwth aachen university open source speech recognition toolkit“, in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

[23] J. Lööf, C. Gollan, S. Hahn, G. Heigold, B. Hoffmeister, C. Plahl, D. Rybach, R. Schlüter und H. Ney, „The rwth 2007 tc-star evaluation system for european english and spanish.“, in *INTERSPEECH*, 2007, S. 2145–2148.

[24] K. Vertanen, „Baseline wsj acoustic models for htk and sphinx: Training recipes and recognition experiments“, Technical report). Cambridge, United Kingdom: Cavendish Laboratory, Techn. Ber., 2006.