Universität Karlsruhe (TH)

Diplomarbeit

Namenserkennung bekannter und unbekannter Namen

Diplomand:	Stefan Ziesemer
Betreuer:	Prof. Dr. A. Waibel
Betreuer:	Dipl.-Inform. H. Holzapfel
Tag der Abgabe:	15.06.2007

Wintersemester 2006/2007

Fakultät für Informatik
Institut für Theoretische Informatik (ITI)

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig verfasst habe und nur die in der Arbeit angegebenen Hilfsmittel und Literaturstellen verwendet habe.

Karlsruhe, den 15. Juni 2007

Stefan Ziesemer

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Der Mensch versucht seit jeher seine Umwelt an sich anzupassen und Arbeiten, die er zu verrichten hat, möglichst effizient zu erledigen. Das führte in der Automobilindustrie von der Entwicklung des Fließbandes (Henry Ford) bis hin zu großen Roboterhallen in denen ein Auto fast vollautomatisch produziert wird. Inzwischen möchten sich die Menschen auch zuhause unliebsame Arbeit wie Staubsaugen, Putzen und Aufräumen von Maschinen abnehmen lassen. Es besteht demnach ein zunehmender Bedarf an Maschinen, die dem Menschen das Leben erleichtern. Erste Maschinen, die diesem Wunsch nachkommen, existieren bereits. So gibt es beispielsweise Staubsaugerroboter, die selbstständig ein Zimmer saugen können. Hielte diese Entwicklung an, würde weiterhin für jede Aufgabe ein spezielles Gerät entwickelt werden. Eine Alternative würde eine Maschine darstellen, die einen ähnlichen Aktionsradius wie ein Mensch aufweist. Dieser Idee nimmt sich der *Sonderforschungsbereich 588 (Humanoide Roboter)* (kurz SFB 588)[1] an der Universität Karlsruhe an. Hier wird ein menschenähnlicher Roboter entwickelt, der einfache Aufgaben im Haushalt übernimmt. Ist es in der Industrie durchaus üblich, und auch machbar, dass ein Robotersystem von ausgebildetem Personal programmiert wird um eine Bestimmte Aufgabe immer wieder präzise durchzuführen, so muss ein Roboter, der in Zukunft in jedem Haushalt zurecht kommen soll und die verschiedensten Aufgaben zu lösen hat, auch von jedem Menschen ohne spezielle Programmierkenntsnisse, bedient und an individuelle Bedrüfnisse angepasst werden können.

Dafür muss also die Art der Programmierung vereinfacht werden. Ein Beispiel dafür ist das *Programmieren durch Vormachen* [EASD00]. Ziel dieses *Programmierparadigmas* ist es, einem System neue Fähigkeiten oder neues Wissen durch Vormachen beizubringen. Dazu muss der Roboter die Aktionen des Menschen erkennen, richtig interpretieren und von ihnen lernen. Üblicherweise werden dazu verschiedene Modalitäten wie Gesten und Sprache verwendet.

[1] `http://www.sfb588.uni-karlsruhe.de/`

Um Gesten erkennen und Sprache verstehen zu können, sind wiederum Systeme nötig, die aus Bildern von Kameras Gesten, Personen und Objekte und aus Schallwellen Sprache erkennen können. Darauf aufbauend wird in beiden Fällen ein System benötigt, das auch *versteht* was das Erkannte bedeutet.

Im Bereich der Spracherkennung bedeutet dies, dass die gesprochene Äußerung eines Bentuzers erkannt und interpretiert werden muss. Gängige Spracherkenner können allerdings nur Wörter erkennen, die bereits in ihrem Wörterbuch enthalten sind. Dazu wird in der Regel das Wort samt seiner Aussprache in Form von Lautfolgen, genauer Phonemen, abgespeichert. Dadurch kann der Erkenner also keine, ihm unbekannten, Wörter erkennen. Um dem Erkenner ein neues Wort *beizubringen*, muss also sein Wörterbuch um das Wort und seine Aussprache erweitert werden. Dies stellt für einen Endandwender eine vergleichbar komplizierte Aufgabe dar, wie beispielsweise die Programmierung eines Videorecorders.

Es wäre also von Vorteil wenn ein möglichst universell einsetzbares Robotersystem neue Wörter durch *vormachen* lernen kann. Der natürlichste Weg wäre ein, möglicherweise mehrfaches, Aussprechen des neuen Wortes oder einer Wortfolge. Besonders häufig treten neue Wörter oder Wortfolgen auf, wenn dem System neue Personen, Orte oder Objekte vorgestellt werden. Dabei kann es sich um Eigennamen wie Orts-, Personen- oder Produktnamen oder eben um Wortfolgen, wie in Buchtiteln, handeln.

Diese Arbeit befasst sich mit dem Problem neue Wörter zu lernen und beschränkt sich dabei beispielhaft auf das Erkennen und Verstehen von bekannten und unbekannten Vornamen. Dabei wird untersucht, wie sich ein großer Namenswortschatz auf das Erkennerverhalten auswirkt. Damit soll eine, bessere Erkennerleistung, durch ein möglichst geschicktes Vorgehen im Umgang mit zusätzlichem Wissen, erzielt werden. Weiterhin werden verschiedene Möglichkeiten zur Generierung bzw. Erkennung der Aussprache eines neuen Namens vorgestellt. Dies hat zum Ziel, herauszufinden welche Möglichkeiten der Aussprachegenerierung bzw. -erfassung sich für das Wiedererkennen eines gelernten Namens eignen.

1.1. Motivation

Im Rahmen des SFB 588 wird ein System entwickelt, das bekannten Personen Auskünfte erteilen und unbekannte Personen kennen lernen kann. Ein erstes System, das dazu in der Lage ist, stellen Holzapfel et al. in „A

Robot learns to know people - First Contacts of a Robot“ vor [HSE+07]. Ein Beispiel Szenario sieht so aus:

Ein Roboter, der auf einem Flur steht, erkennt, ob eine vorbeilaufende Person sich mit ihm unterhalten möchte [Sch05]. Anhand einer Gesichtserkennung [ES05, KHEW07], kann das Gesicht einer Person erkannt und eine bekannte Person identifiziert werden. Ist die Person unbekannt, soll der Roboter diese ansprechen und so die Identität der Person in Erfahrung bringen. Dazu wird der Name der Person erfragt und zusammen mit Bildern der Person gespeichert. Das Vokabular des Erkenners wird außerdem um den Namen der Person erweitert, falls dieser bislang nicht bekannt war.

Ein solcher Kennenlerndialog zwischen einem Roboter (R) und einer Person (P) kann wie folgt ablaufen:

R *Hallo, was kann ich für dich tun?*

P *Ich möchte ein Paket für Herrn Brunn abgeben.*

R *Dazu muss ich wissen wer du bist. Wie heißt du denn?*

P *Ich heiße Adalbert.*

R *Ich habe dich nicht verstanden. Bitte buchstabiere deinen Namen.*

P *A D A L B E R T*

R *Habe ich richtig verstanden, dass du Adalbert heißt?*

P *Ja.*

R *Ok, ich sage dir jetzt wo du Herrn Brunn findest...*

Begegnet die Person dem System eine weiteres Mal, kann der Name direkt verstanden und die Person erkannt werden.

R *Hallo, was kann ich für dich tun?*

P *Ich möchte ein Paket für Frau Ziegler abgeben.*

R *Dazu muss ich wissen wer du bist. Wie heißt du denn?*

P *Ich heiße Adalbert.*

R *Hallo Adalbert. Frau Ziegler findest du ...*

1.2. Anforderungen

In dieser Arbeit soll ein vorhandenes Spracherkennungssystem so erweitert werden, dass es in der Lage ist, sowohl bekannte Namen zu verstehen, als auch unbekannte Namen zu erlernen. Dazu müssen verschiedene Teilsysteme entwickelt werden. Ein grundlegendes Problem beim maschinellen Verstehen von Namen ist die große Anzahl verschiedener Namen. Es ist praktisch nicht möglich, alle Namen der Welt für einen Spracherkenner mit ihrer jeweiligen Aussprache zu kennen. Weiterhin sinkt die Erkennergenauigkeit bei einer zunehmenden Anzahl an bekannten Namen. Deshalb ist es nötig, eine möglichst gute Abwägung zwischen der Wahrscheinlichkeit, dass ein Name dem Erkenner bekannt ist, und der Anzahl der im Wörterbuch vorhandenen Namen zu finden. Ein weiteres Problem, das gelöst werden muss, ist, zu erkennen, wann ein unbekannter Name im Dialog auftritt. Das System muss *bemerken*, dass es mit einem neuen und unbekannten Namen konfrontiert wird. Um nun einen neuen Namen erlernen und wiedererkennen zu können, muss wenigstens seine Aussprache in Form einer Lautfolge ermittelt werden. Besser noch wäre es, wenn das System in der Lage wäre die Schreibweise eines Namens zu erlernen. Weiterhin soll das System sowohl mit einem englischsprachigen, als auch mit einem deutschsprachigen Spracherkenner betrieben werden können. Zusammengefasst sind also folgende Probleme zu lösen:

- Erkennung von (dem System) unbekannten Namen
- Erlernen der Aussprache eines neuen Namens, in Form von Lautfolgen
- Erlernen der Schreibweise eines neuen Namens
- Möglichst gute Wiedererkennung eines gelernten Namens
- Abwägung zwischen Anzahl bekannter Namen und Wahrscheinlichkeit, dass der Name eines Besuchers dem System bekannt ist
- Entwicklung in zwei Sprachen (Deutsch, Englisch)
- Erweiterung des Erkennervokabulars um neu erlernte Namen

1.3. Überblick

In Kapitel 2 wird die für diese Arbeit wichtige Theorie eingeführt. Dazu gehört eine kurze Einführung von linguistischen Begriffen, die in dieser Arbeit verwendet werden, sowie einige Grundlagen über den Aufbau und die Funktionsweise eines Spracherkenners. Weiterhin wird das Vorgehen zur Evaluation von Spracherkennern beschrieben. Anschließend werden verwandte Arbeiten in Kapitel 3 vorgestellt. Kapitel 4 zeigt die Umsetzung der entwickelten Komponenten. Dazu gehört ein Erkenner zur Detektion unbekannter Wörter in Abschnitt 2.6, ein Buchstabiererkenner zur Erfassung der Schreibweise eines Namens in Abschnitt 4.2, ein Phonemerkenner in Abschnitt 4.4 sowie ein Graphem-zu-Phonem-Konverter zur Detektion bzw. Generierung von Phonemsequenzen in Abschnitt 4.3. Weiterhin wird dort ein Entwurf für ein mehrstufiges System vorgestellt, das die Erkennung von bekannten Namen verbessern soll. In Kapitel 5 wird das Vorgehen bei der Aufnahme der Testdaten beschrieben, sowie eine Namensstatistik vorgestellt. Hier wird auch das Erkennerverhalten bei wachsendem Namensvokabular untersucht. Am Ende des Kapitels wird ein Ansatz zur Fusion gleichartiger Eingaben vorgestellt und auf Phonemhypothesen und Buchstabierhypothesen ausgewertet. Aufbauend auf den Ergebnissen aus Kapitel 5 wird in Kapitel 6 ein mehrstufiges System vorgestellt, das die zu erwartende Erkennerleistung optimiert. Dazu wird, in Abhängigkeit der Anzahl an bekannten Namen und der Wahrscheinlichkeit, dass der Namen einer Person tatsächlich bekannt ist, entschieden, mit welcher Konfiguration die Eingabe dekodiert wird. Kapitel 7 beschäftigt sich mit dem Wiedererkennen gelernter Namen. In Kapitel 8 werden die wesentlichen Ergebnisse noch einmal zusammengefasst und mit den anderen Arbeiten verglichen. Kapitel 9 fasst die wesentlichen Punkte dieser Arbeit zusammen und gibt einen Ausblick auf mögliche Verbesserungen des Systems.

2. Terminologie und Grundlagen

Dieses Kapitel führt einige Begriffe aus dem Bereich der Linguistik ein und gibt einen Überblick über den grundlegenden Aufbau eines Spracherkenners. Als weitere wichtige Grundlage wird insbesondere auf die Modellierung von Sprachmodellen eingegangen, denen in dieser Arbeit eine besondere Bedeutung zukommt.

2.1. Verwendete Begriffe aus der Linguistik

In diesem Abschnitt sollen die in dieser Arbeit verwendeten Begriffe aus dem Bereich der Linguistik kurz eingeführt werden.

2.1.1. Dialog und Äußerung

Der Duden [Aub01] definiert einen *Dialog* als „eine von zwei oder mehreren Personen *abwechselnd* geführte Rede". In dieser Arbeit werden ausschließlich Dialoge in Form von abwechselnden Äußerungen zwischen genau *einem* Menschen und *einer* Maschine betrachtet.

Als *Äußerung* wird dabei eine Spracheingabe eines Menschen in Form eines gesprochenen Satzes, eines isoliert gesprochenen Wortes oder einer Buchstabiersequenz oder die Antwort der Maschine bezeichnet.

2.1.2. Grapheme, Morpheme und Phoneme

Der Duden definiert ein *Graphem* als die „kleinste bedeutungstragende Einheit in einem Schriftsystem die ein Phonem bzw. eine Phonemfolge repräsentiert" [Aub01]. Eine etwas andere Definition gibt Bußmann im „Lexikon der Sprachwissenschaft", dort wird ein Graphem wie folgt definiert: „Distinktive Einheit eines Schriftsystems. ... Generell betrachtet man als Graphem nur kleinste distinktive Einheiten eines Schriftsystems. In Alphabetenschriften dienen Grapheme in der Regel der Verschriftung von

phonemischen Objekten, im Idealfall Phonemen ..." [Bus02]. Grapheme werden im Deutschen in Form von lateinischen Schriftzeichen repräsentiert, dabei wird nicht zwischen der Schreibweise eines solchen Zeichens unterschieden.

Als *Morphem* wird die „kleinste bedeutungstragende Einheit in einem Sprachsystem" [Aub01] bezeichnet. Ein Morphem kann also in keine kleineren bedeutungstragenden Einheiten aufgelöst werden.

Als *Phonem* wird hingegen die „kleinste bedeutungsunterscheidende sprachliche Einheit" bezeichnet [Aub01].

Der Duden nennt als Beispiel die Wörter *Bein* und *Pein*. Sie unterscheiden sich in ihrer Aussprache in den Phonemen *b* und *p* und in ihrer Schreibweise in den Graphemen *B* und *P*. Bei beiden Wörtern handelt es sich um eigenständige Morpheme. Das Wort *Beine* besteht hingegen aus zwei Morphemen, nämlich *Bein* und *e*. Dabei ist zu beachten, dass die Schreibweise für die Phoneme und Grapheme in diesem Beispiel beliebig gewählt wurde und nicht der Schreibweise der Wörter entsprechen muss.

2.2. Aufbau eines Spracherkenners

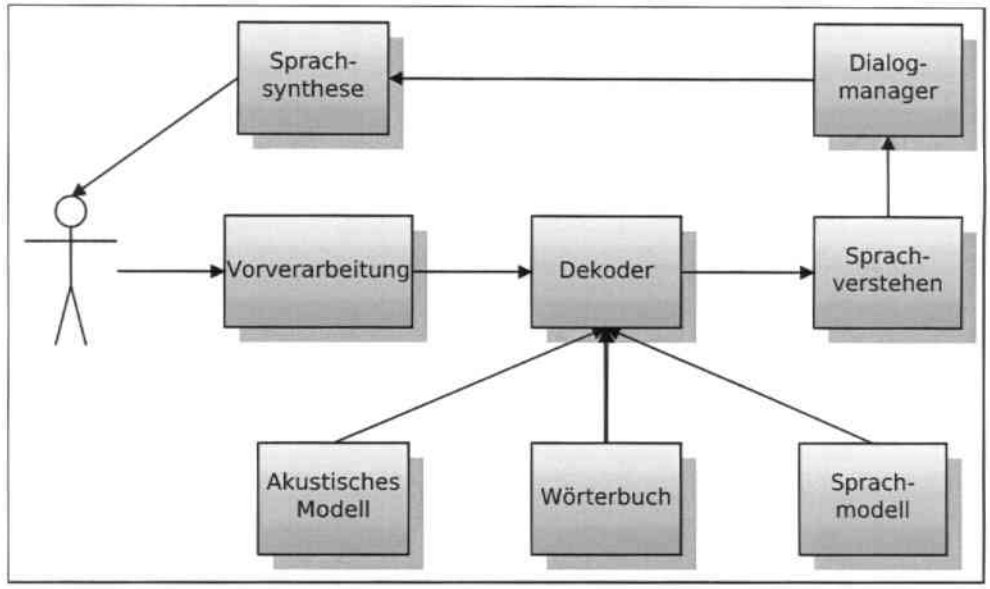

Abbildung 2.1.: Übersicht über den Aufbau eines Spracherkenners (integriert in ein Dialogsystem)

Abbildung 2.1 zeigt den Aufbau eines statistischen Spracherkenners [ST95, Rog05], eingebettet in ein Dialogsystem. Dabei durchläuft eine Eingabe erst die *Vorverarbeitung*. Dort findet eine Normalisierung sowie die Merkmalsextraktion statt. Im Dekoder werden die Merkmale über das akustische Modell und dem Wörterbuch mit dem Sprachmodell (siehe Abschnitt 2.4) verknüpft. Diese Verknüpfung geschieht mit Hilfe der *Fundamentalformel der Spracherkennung* (Abschnitt 2.3). Anschließend kann, je nach verwendetem Sprachmodell, die erkannte Eingabe, auch *Hypothese* genannt, geparst und an eine Sprachverstehenskomponente weitergereicht werden. Ein *Dialogmanager* benutzt diese *verstandene* Sprache, um auf die Eingabe des Benutzers zu reagieren und mit ihm beispielsweise über ein *Sprachsynthesesystem*, das geschriebenen Text aussprechen kann, zu kommunizieren.

2.3. Fundamentalformel der Spracherkennung

Ausgehend vom Bayestheorem (2.1) [Bay] soll die Fundamentalformel der Spracherkennung hergeleitet und erklärt werden. Im Folgenden seien X und W Zufallsvariablen. Dabei steht X für die Beobachtung des Sprachерkenners in Form von Merkmalen, die aus der Vorverarbeitung hervorgehen, und W für eine Hypothese. Gesucht ist nun die Hypothese mit der höchsten Wahrscheinlichkeit:

$$p(W|X) = \frac{p(X|W) \cdot p(W)}{p(X)} \tag{2.1}$$

Dabei bezeichnet $p(X|W)$ die Wahrscheinlichkeit für das Eintreten eines Ereignisses X unter der Bedingung, dass W auftritt. $p(X|W)$ modelliert, also das Zustandekommen von Merkmalen gegeben einem gesprochenen Wort. $p(W)$ ist die Wahrscheinlichkeit dafür, dass ein Wort überhaupt gesprochen wird. Diese Wahrscheinlichkeit wird über ein Sprachmodell geschätzt (Abschnitt 2.4). $p(X)$ bezeichnet die *a priori Wahrscheinlichkeit*, dass eine Beobachtung überhaupt auftritt. $p(W|X)$ bezeichnet die *a posteriori Wahrscheinlichkeit*, für die Hypothese W für eine Beobachtung X. Bei der Spracherkennung wird dasjenige Wort $\hat{W}$ gesucht, dessen

Wahrscheinlichkeit maximal ist (2.2).

$$\hat{W} = \arg\max_W p(W|X) = \arg\max_W \frac{p(X|W) \cdot p(W)}{p(X)} \tag{2.2}$$

$p(X)$ ist konstant für ein gegebenes W, wodurch sich (2.2) vereinfacht zu:

$$\hat{W} = \arg\max_W p(W|X) = \arg\max_W p(X|W) \cdot p(W) \tag{2.3}$$

Der Ausdruck (2.3) maximiert also das Produkt aus der Wahrscheinlichkeit des akustischen Modells $p(X|W)$ und des Sprachmodells $p(W)$. $\hat{W}$ bezeichnet dann diejenige Wortfolge W, die unter der gegebenen Beobachtung X am wahrscheinlichsten ist und somit der wahrscheinlichsten Hypothese für die tatsächliche Wortfolge entspricht.

2.4. Sprachmodelle

Ein Sprachmodell modelliert die „linguistischen" Aspekte einer Sprache. Eine Möglichkeit dieser Modellierung ist aus dem Schulunterricht bekannt: Dort lernt man die Grammatik einer Sprache. Auch bei der maschinellen Spracherkennung können Grammatiken als Sprachmodell eingesetzt werden. Es gibt aber auch weitere Möglichkeiten. In diesem Abschnitt soll erklärt werden, wozu überhaupt ein Sprachmodell nötig ist und welche Sprachmodelle für diese Arbeit von Bedeutung sind.

Ein Sprachmodell liefert zusätzliches Wissen über die Wahrscheinlichkeit einer Aussage in einer gegebenen Sprache. Im Deutschen ist der Satz „Das ist aber ganz schön bunt" wahrscheinlicher zu hören als der Satz „Das ist aber ganz schön Hund". Ein gutes Sprachmodell für die deutsche Sprache würde dementsprechend Ersterem eine höhere Wahrscheinlichkeit zuweisen. Gibt also das akustische Modell eine Bewertung für den Klang eines Wortes ab, so kann mit einem Sprachmodell die Wahrscheinlichkeit des Auftretens eines Wortes oder einer Wortfolge berechnet werden.

2.4.1. Grammatik

Wie bereits erwähnt, kann auch in der automatischen Spracherkennung eine Grammatik verwendet werden. Dies hat den Vorteil, dass die möglichen Hypothesen durch die Grammatik direkt vorgegeben werden, was wiederum die Interpretation des Gesagten erleichtert, da von vornherein

bekannt ist, was der Spracherkenner erkennen kann und was nicht. Gleichzeitig stellt dies auch einen Nachteil dar: Die möglichen Äußerungen eines Benutzers sind festgelegt und müssen beim Entwurf der Grammatik vorhergesehen werden - andernfalls können sie nicht erkannt werden. Ein Vorteil der Grammatiken ist der, dass keine bzw. nur sehr wenige Trainingsdaten vorhanden sein müssen, um ein geeignetes Sprachmodell für ein bestimmtes Anwendungsgebiet zu erstellen. Schöning [Sch01] definiert eine Grammatik wie folgt:

Eine Grammatik ist ein 4-Tupel $G = (V, \Sigma, P, S)$, das folgende Bedingungen erfüllt:

- V ist eine endliche Menge, die Menge der Variablen.
- Σ ist eine endliche Menge, das Terminalalphabet.
- Es muss gelten: $V \cap \Sigma = \emptyset$.
- P ist die endliche Menge der Regeln oder Produktionen. Formal ist P eine endliche Teilmenge von $(V \cup \Sigma)^+ \times (V \cup \Sigma)^*$.
- $S \in V$ ist die Startvariable.

In der automatischen Spracherkennung werden kontextfreie Grammatiken verwendet. Dazu müssen nach Schöning zwei weitere Regeln erfüllt sein. Erstens muss für alle Produktionsregeln $w_1 \rightarrow w_2$ in P gelten, dass $|w_1| \leq |w_2|$ und zweitens $w_1 \in V$ ist. Die erste Regel besagt demnach, dass keine Regel die produzierte Ausgabe verkürzt, die zweite, dass auf der linken Seite der Regel immer genau eine Variable stehen muss.

In dieser Arbeit werden Grammatiken in *SOUP*-Notation verwendet. Abbildung 2.2 zeigt einen Ausschnitt einer Grammatik in SOUP Syntax. Dabei stellen die Ausdrücke in den eckigen Klammern die Variablen, oder auch Nichtterminale genannt, dar. Eine Produktionsregel beginnt immer mit einem Nichtterminal, das auf eine Liste von Terminal- und/oder Nichtterminal-Symbolen verweist. Startsymbole werden mit einem *s* gekennzeichnet. Weitere Details zu SOUP beschreibt Gavaldà in „Soup: a parser for real-world spontaneous speech“ [Gav00].

2.4.2. N-Gramm

Eine weitere Möglichkeit ein Sprachmodell zu modellieren, stellt die Verwendung eines *N-Gramms* dar [CT94, Rog05]. Hier wird anhand großer

```
s[peopleid:informName,VP,_]
    ( [peopleid:informName,V,_] [peopleid:obj_person,NP,_] )

[peopleid:informName,V,_]
    ( mein name ist )
    ( ich bin )

s[peopleid:obj_person,NP,_]
    ( [first_NT] )

[first_NT]
    ( petra )
    ( kai )
    ( rainer )
    ( dirk )
```

Abbildung 2.2.: Ausschnitt einer Grammatik in SOUP Notation

Trainingsdaten eine Statistik über das Auftreten von Worten in einer bestimmten Historie berechnet. Der Parameter N steht für die Größe der Historie, in die zurück geschaut wird. Allgemein schreibt man für ein N-Gramm der Größe k:

$$p(\omega_m|\omega_{m-(k-1)}, \ldots \omega_{m-1}) \tag{2.4}$$

Ein 1-Gramm, bzw. Unigramm, entspricht der Auftretenswahrscheinlichkeit eines Wortes kurz $p(\omega_m)$. Hierbei stellt ω_m das aktuell zu bewertende Wort dar. Ein 2-Gramm, auch Bigramm genannt, entspricht dann $p(\omega_m|\omega_{m-1})$.

Die Häufigkeit des Auftretens einer Wortfolge $\omega_1 \ldots \omega_j$ in den Trainingsdaten wird mit # bezeichnet. Die Wahrscheinlichkeit lässt sich dann aus den Trainingsdaten wie folgt schätzen:

$$p(\omega_m|\omega_{m-(k-1)}, \ldots \omega_{m-1}) = \frac{\#(\omega_{m-(k-1)}, \ldots \omega_{m-1}, \omega_m)}{\#(\omega_{m-(k-1)}, \ldots \omega_{m-1})} \tag{2.5}$$

Da für wachsendes N immer mehr Kombinationsmöglichkeiten entstehen, für eine gute Abschätzung aber möglichst viele Trainingsdaten vorhanden sein müssen, wird in der Praxis nicht nur ein N-Gramm der Größe N sondern alle k-Gramme mit $k \leq N$ verwendet. Um diese verwenden zu können müssen Rückfallwahrscheinlichkeiten berechnet werden. Details hierzu beschreibt Rogina [Rog05].

Vorteil eines N-Gramm-Sprachmodells gegenüber einer Grammatik ist eine allgemeinere Abdeckung der Sprache. Um dies zu gewährleisten sind allerdings große Mengen an Trainingsdaten in Form von Texten nötig.

2.5. Perplexität

Um ein Sprachmodell zu bewerten, können die Ergebnisse eines Spracherkenners unter Verwendung verschiedener Sprachmodelle auf derselben Testdatenmenge verglichen werden. Dabei treten aber verschiedene Probleme auf. Zum einen benötigt ein Erkennerlauf auf einer großen Datenmenge eine gewisse Zeit, zum anderen hängt die Erkennerleistung auch vom Zusammenspiel der verwendeten Akustik und des Dekoders ab. Es ist also wünschenswert ein Sprachmodell auf einem einfacheren Weg evaluieren zu können. Dazu wurde die *Perplexität* (PPL) eingeführt [JMBB77]. Die Perplexität beruht zwar auf der *Entropie*, einem Maß für den mittleren Informationsgehalt eines Zeichens [Sha48], wird aber als aussagekräftiger als diese empfunden. Formel (2.6) zeigt die Definition der Entropie [MS05].

$$H(X) = \mathbb{E}[-\log_2(p(X))] = -\sum_{a \in A} p(a) \log_2(p(a)) \tag{2.6}$$

Dabei ist X eine A-wertige diskrete Zufallsvariable mit Zähldichte p. Die Entropie stellt ein Maß für die Unbestimmtheit eines Ereignisses dar.

Die Perplexität ist definiert als:

$$\mathrm{PPL} = 2^{H(X)} \tag{2.7}$$

In der Spracherkennung wird nun das Sprachmodell als eine Informationsquelle betrachtet [Rog05] und jedes Wort stellt ein Symbol dar, das vom Sprachmodell emittiert wird. Unter der Annahme, dass die Informationsquelle mit einer unendlich langen Symbolfolge komplett beschrieben werden kann, kommt man zu Formel (2.8)

$$H(X) = -\lim_{n \to \infty} \frac{1}{n} \log_2 p(w_1, w_2, \ldots w_n) \tag{2.8}$$

Wobei hier w_i einem Wort entspricht. In der Praxis treten allerdings keine unendlichen Wortfolgen auf, somit wird $H(x)$ angenähert durch (2.9).

$$H(X) \approx -\frac{1}{n} \log_2 p(w_1, w_2, \ldots w_n) \tag{2.9}$$

Eingesetzt in (2.7) ergibt sich nun (2.10) und vereinfacht (2.11).

$$\mathrm{PPL} = 2^{H(X)} = 2^{-\frac{1}{n}\log_2 p(w_1,w_2,\ldots w_n)} \tag{2.10}$$

$$\mathrm{PPL} = p(w_1, w_2, \ldots w_n)^{\frac{1}{n}} \tag{2.11}$$

Anschaulich ist die Perplexität also ein Maß über die zu erwartende Unsicherheit, gegeben eine Historie von Worten. Eine große Perplexität spiegelt also eine hohe Unsicherheit über das zu erwartende Wort wieder. Ein anschauliches Beispiel aus der Spracherkennung stellt der Satz „Mein Name ist ...“ dar. Hier können sehr viele Namen erwartet werden. Es ist demnach schwer vorhersagbar, welcher Name tatsächlich gesprochen wird. Ein anderer Ausdruck wie „Guten ...“ lässt weniger Möglichkeiten zu.

Wie obiges Beispiel schon andeutet, stellt die sehr hohe Perplexität bei der Erkennung von Namen ein inherentes Problem dar. Ein Satz wie „Mein Name ist ...“ lässt hunderttausende Namen zu. Es gilt also die Perplexität möglichst geschickt zu verkleinern. Eine Möglichkeit stellt die Verwendung der X-häufigsten Namen dar, da somit die Wahrscheinlichkeit, dass ein Name erkannt werden kann erhöht wird.

2.6. OOV-Problem

Als OOV (engl. out of vocabulary) werden Wörter bezeichnet, die nicht im Erkennervokabular enthalten sind. Dabei handelt es sich überwiegend um Eigennamen wie Vornamen, Nachnamen oder Straßennamen sowie um Wörter, die sehr selten vorkommen. Weiterhin können, abhängig von dem Anwendungsgebiet, in dem der Spracherkenner verwendet wird, auch neue Wörter entstehen. Dazu können neue Produktbezeichnungen zählen oder, gerade im Deutschen, neue Wortschöpfungen, die sich aus anderen Worten zusammen setzen.

Durch unbekannte Wörter können verschiedene Fehler bei der Erkennung entstehen. So kann das unbekannte Wort zum einen nicht richtig erkannt werden, zum anderen kann es sein, dass durch ein falsch erkanntes Wort ein Folgefehler auftritt. Ein Beispiel: Der Name Sebastian sei nicht im Erkennervokabular enthalten. Die gesprochene Eingabe lautet „Vorname Sebastian“. Der Erkenner erkennt „Mein Name ist Jan“. Durch das unbekannte Wort „Sebastian“ wurde also die komplette Aussage falsch verstanden.

Wünschenswert wäre in einem solchen Fall eine Hypothese der Form „Vorname oov“. Mit diesem Problem beschäftigt sich die Arbeit „Erkennen

und Lernen neuer Wörter“ von Schaaf [Sch04]. Darin wird auch eine Modellierung zur Erkennung von unbekannten Wörtern vorgestellt. Es handelt sich dabei um die Head-Tail-Modellierung (siehe auch Abbildung 2.3). Als Kopf (engl. Head) bezeichnet man eine genaue Modellierung mit Phonemen für den Anfang eines Wortes. Die Auswirkung der verwendeten Anzahl an Phonemen werden ebenfalls von Schaaf untersucht [Sch04]. Der Rumpf (engl. Tail) besteht aus mehreren unscharf modellierten Zuständen, sogenannte *Garbage-Modelle*. Ein solches, unscharfes, akustisches Modell passt auf möglichst viele verschiedene Merkmale. Um unbekannte Wörter

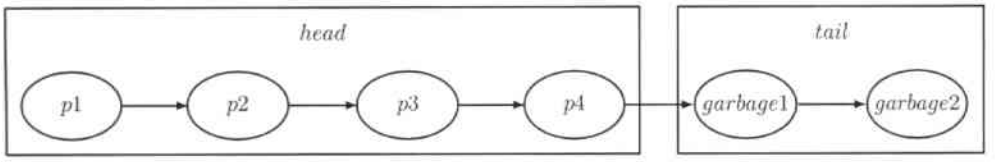

Abbildung 2.3.: Head-Tail-Modell zur OOV-Erkennung

erkennen zu können, werden Wörter nach dem Head-Tail Prinzip generiert und jeweils als ein OOV-Wort in das Wörterbuch aufgenommen. Durch die unscharfe Modellierung am Rumpf können so verschieden lange und auch verschieden klingende Wörter modelliert werden, da lediglich der Anfang eines Wortes, also die ersten vier Phoneme, fest modelliert sind. Weiterhin muss sichergestellt werden, dass ein OOV-Wort eine geringere Auftretenswahrscheinlichkeit hat als ein normales Wort. Dies kann beispielsweise im Sprachmodell berücksichtigt werden. Ebenso ist es möglich, OOVs nur an bestimmten Stellen zu erlauben, dies kann beispielsweise mit einer Grammatik realisiert werden.

2.7. Evaluation von Spracherkennern

Um die Leistung von Spracherkennern vergleichen zu können, wird ein Evaluationsmaß benötigt. Hier sollen die dafür gebräuchlichen Maße definiert werden [MMD+04, Rog05]. Zur Evaluation werden sowohl die *Hypothesen* des Erkenners, als auch die *Referenzen*, also die tatsächliche Äußerung, benötigt. Nun wird jede Hypothese ihrer entsprechenden Referenz gegenüber gestellt und die minimale Editierdistanz [WF74] zwischen beiden berechnet. Dabei werden drei mögliche Fehlerarten definiert und zwar Ersetzungen, Einfügungen und Auslassungen. Tabelle 2.1 zeigt dazu je ein Beispiel.

Fehlerart	Referenz	Hypothese
Ersetzung	Mein Name ist Sebastian	Mein Name ist Jan
Einfügung	Mein Name ist Sebastian	Nein mein Name ist Sebastian
Auslassung	Mein Name ist Sebastian	Name ist Sebastian

Tabelle 2.1.: Übersicht über die drei Fehlerarten mit Beispielen

Unter Verwendung dieser Fehlermessung kann nun die sogenannte Wortfehlerrate, abgekürzt WER (engl. word error rate) definiert werden.

$$\mathrm{WER} = \frac{\#\text{Einfügungen} + \#\text{Ersetzungen} + \#\text{Auslassungen}}{\#\text{Wörter}} \cdot 100\% \quad (2.12)$$

Der Wertebereich der Wortfehlerrate beginnt bei Null (keine Fehler) und ist nach oben nicht begrenzt.

Weiterhin definiert man die Wortakkuratheit, abgekürzt WAcc (engl. word accuracy).

$$\mathrm{WAcc} = \frac{\#\text{Richtig erkannte Wörter}}{\#\text{Wörter}} \cdot 100\% \quad (2.13)$$

Der Wertebereich der Wortakkuratheit liegt immer zwischen 0% und 100%, da maximal soviele Wörter richtig erkannt werden können, wie Wörter in der Referenz vorhanden sind.

Mit Satzakkuratheit, abgekürzt SAcc (engl. sentence accurancy), bezeichnet man das Verhältniss der komplett richtig erkannten Sätze zur Anzahl aller Sätze.

$$\mathrm{SAcc} = \frac{\#\text{Richtig erkannte Sätze}}{\#\text{Sätze}} \cdot 100\% \quad (2.14)$$

Auch hier liegt der Wertebereich zwischen 0% und 100%, da es nicht möglich ist mehr Sätze richtig zu erkennen als vorhanden sind und auch nur maximal alle Sätze falsch erkannt werden können. In dieser Arbeit wird *Satz* und *Wort* im Zusammenhang mit einer Evaluation immer im Sinne obiger Definition verwendet. Ein Satz besteht also aus mehreren Worten. Wird ein Vorname buchstabiert, entspricht der ganze Vorname

Fehlermaß	Wertebereich	Optimalwert
WER	$0\% - \infty\%$	0%
WAcc	$0\% - 100\%$	100%
SAcc	$0\% - 100\%$	100%

Tabelle 2.2.: Übersicht über die drei Evaluationsmaße

einem Satz und die einzelnen Buchstaben einem Wort. Tabelle 2.3 soll dies noch einmal verdeutlichen. Die Bezeichnung *_wb* in der Zeile des Phonemerkenners steht dabei für die Markierung einer Wortgrenze (engl. word boundary) und soll an dieser Stelle nicht weiter betrachtet werden.

Modus	Satz	Das 1. Wort des Satzes
Grammatik	Ich heiße Peter	Ich
Buchstabierung	P E T E R	P
Phonemerkennung	p_wb eh t er2_wb	p_wb

Tabelle 2.3.: Sätze und Wörter in Abhängigkeit des Erkennungsmoduses

2.8. Janus und Tapas

Das in dieser Arbeit entwickelte System verwendet das *JANUS Recognition Toolkit* [FGH+97], kurz Janus, und den darin enthaltenen Ibis Dekoder [SMFW01] als Spracherkennungssystem. Weiterhin wird die von Janus angebotene Möglichkeit, zur Laufzeit zwischen verschiedenen Sprachmodellen zu wechseln, genutzt. Ebenso wird von einer Funktion Gebrauch gemacht, mit der, während der Erkennung verwendeten Grammatikregeln, über einen Dialogmanager verschieden stark gewichtet werden können [FHW04]. Für die schnelle Entwicklung von Prototypen kann Janus über die Skriptsprache Tcl programmiert werden [RT99]. Für die Experimente werden bereits vorhandene akustische Modelle in englischer und

deutscher Sprache verwendet. Diese Modelle wurden auf kontinuierlicher Sprache und verschiedenen Sprechern trainiert, die Trainingsdaten enthalten im Englischen wenige und im Deutschen fast keine Buchstabiersequenzen.

Der Dialogablauf wird von *Tapas* [Hol05], einem Dialogmanager, gesteuert. Dieser nimmt die Hypothesen von Janus entgegen, interpretiert sie und generiert eine passende Antwort, die an ein Sprachsynthese-Programm geschickt wird.

3. Verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über Arbeiten, die sich ebenfalls mit dem Verstehen von Namen beschäftigen. Zu Beginn wird ein Ansatz vorgestellt, der die Erkennerbewertungen gesprochener und buchstabierter bekannter Namen verknüpft, um dadurch eine bessere Hypothese zu erreichen. Eine weitere Arbeit beschreibt einen Ansatz, der mit zusätzlichem linguistischen Wissen in der Lage ist, Namen zu lernen. Dazu kann mit einem Teilsystem aus einer Aussprachehypothese in Form einer Phonemsequenz eine Buchstabierhypothese und aus einer Buchstabenhypothese eine Phonemsequenz gewonnen werden. Eine weitere Arbeit stellt ein System vor, das durch zusätzliches Wissen über das Gesprächsthema gezielt passende Namen in das Vokabular des Erkenners lädt. Dieser Ansatz verfolgt also die Verringerung der Perplexität durch zusätzliches Wissen. Am Ende dieses Kapitels werden zwei Arbeiten vorgestellt, die sich die Redundanz, d. h. das mehrfache Vorkommen ein und der selben Information, zunutze machen um die Erkennungsleistung zu verbessern.

3.1. Erkennung von Eigennamen durch Fusion gesprochener und buchstabierter Eingaben

In der Arbeit „Recognition of Spoken and Spelled Proper Names" beschreiben Meyer und Hild ein Verfahren, das die Erkennung von Namen durch Fusion einer gesprochenen und einer buchstabierten Eingabe verbessert [MH97]. Dabei steht eine Liste von 1337 Nachnamen mitsamt ihrer Aussprache zur Verfügung.

Für große Listen von bekannten Namen wird in einem ersten groben Erkennungslauf eine verkleinerte Liste mit potenziellen Kandidaten ermittelt. Dazu wird davon ausgegangen, dass ein Name erst ausgesprochen und anschließend buchstabiert wird. Die Eingabe liegt also in der Form „Smith S M I T H" vor. Um die Grenzen des ausgesprochenen und buchstabierten Teils zu erkennen, wird ein spezielles Sprachmodell für den Erkenner

verwendet. Dazu wird ein Wörterbuch angelegt, das sowohl die Phoneme, als auch die Buchstaben enthält. Das Sprachmodell erlaubt die in Abbildung 3.1 dargestellten Übergänge. Damit kann eine Phonemsequenz

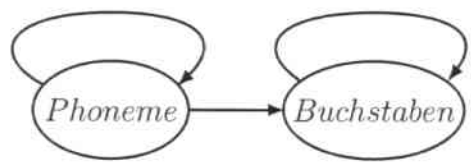

Abbildung 3.1.: Sprachmodell zur Detektion der Aussprache-Buchstabier-Grenzen nach Meyer und Hild [MH97]

gefolgt von einer Buchstabiersequenz erkannt werden. Die Buchstabiersequenzen werden dazu genutzt, die 100 bzw. 1.000 ähnlichsten Namen aus einem Hintergrundwörterbuch zu finden. Diese Namen werden dann für einen zweiten Erkennerlauf in den Spracherkenner aufgenommen. Dadurch werden bessere Grenzen für den gesprochenen Teil und den buchstabierten Teil des Namens gefunden. Anschließend wird auf dem buchstabierten Teil ein Lauf mit einem speziellen Buchstabiererkenner, auf Basis eines MS-TDNN (engl. multistate time delay neural network) [HW93], gestartet. Die Namen mit der besten Bewertung werden für einen weiteren Erkennerlauf mit dem Spracherkenner Janus verwendet.

Daraus resultieren zwei N-Besten-Listen, eine für die Aussprache und eine für die Buchstabierung. Daraus wird die Bewertung für eine neue Liste nach Formel (3.1) berechnet. Der Parameter λ muss durch eine Kreuzvalidierung auf Trainingsdaten gewonnen werden.

$$Y(i) = \lambda \cdot Y_L(i) + (1 - \lambda) \cdot Y_F(i) \tag{3.1}$$

Dabei bezeichnet $Y(i)$ das Ergebnis der Fusion für den Namen i. $Y_L(i)$ steht für die Bewertung des Namens i aus der Buchstabiererkennung. $Y_F(i)$ stellt die Bewertung des Namens i für die Aussprachеerkennung dar. In der Arbeit von Meyer und Hild wird λ mit $\approx 0,96$ als optimal angegeben. Dies erklärt sich durch den verwendeten Buchstabiererkenner, der ohne spezielles Sprachmodell eine Wortakkuratheit von zirka 90% aufweist. Wichtig hierbei ist, dass ein Wort im Sinne der Evaluation einem Buchstaben im Sinne der menschlichen Sprache entspricht. Für die 1337 Namen konnte mit der oben beschriebenen Strategie die Erkennungsleistung gegenüber der Einzelerkennung von 60, 0% für den Phonemerkenner

bzw. 96,5% für den Buchstabiererkenner auf 96,9% für die kombinierte Erkennung verbessert werden.

Damit ein Name bekannt ist, muss sowohl seine Schreibweise, als auch seine Aussprache in Form einer Phonemsequenz bekannt sein. Unbekannte Namen können mit diesem Ansatz also nicht erkannt werden. Zwar kann nach Angaben von Meyer und Hild der MS-TDNN-Buchstabiererkenner eine Liste von bis zu einer Million Namen verwenden, diese muss jedoch erst einmal vorliegen.

Die entwickelte Fusion anhand der N-Bestenliste von Aussprache- und Buchstabiererkenner dürfte ein nützlicher Ansatz zum Auflösen von Mehrdeutigkeiten, auch Disambiguierung genannt, homophoner, also gleich klingender, Wörter sein. Dies trifft vor allem dann zu, wenn durch das Sprachmodell keines der Wörter bevorzugt wird. So wird eine Grammatik, die an einer Stelle einen Namen erlaubt, weder *Stefan* noch *Stephan* bevorzugen. Durch die Buchstabierung unterscheiden sich diese beiden Namen allerdings.

3.2. Erlernen beliebiger Namen

In der Arbeit von Chung et al. [CSW03] wird ein System beschrieben, das gesprochene Namen über eine Telefonverbindung verstehen soll. Auch hier wird von einer Eingabe der Form „Smith S M I T H“ ausgegangen, also ein Name zuerst in gesprochener und dann in buchstabierter Form erwartet. Die Erkennung läuft dann in drei Schritten ab:

- Erkennung der Buchstabierung und erste Buchstabierhypothesen
- Anreicherung der Hypothesen mit linguistischem Wissen
- Erneuter Lauf mit eingeschränktem Suchraum

Im ersten Schritt wird ein OOV-Erkenner in Kombination mit einem Buchstabiererkenner eingesetzt. Dieser gibt eine Hypothese für ein unbekanntes Wort, gefolgt von einer Buchstabierhypothese samt den entsprechenden Zeitgrenzen zurück.

In einem zweiten Schritt wird diese Hypothese mit zusätzlichem Wissen zu Morphemen, Silben, Phonemen und Buchstaben angereichert. Dies geschieht durch ein System namens ANGIE [SLM96]. Es enthält eine Statistik über die Strukturen von Wörtern. Abbildung 3.2 zeigt die zusätzlichen Informationen zu einem Wort. Diese Baumstrukturen werden verwendet,

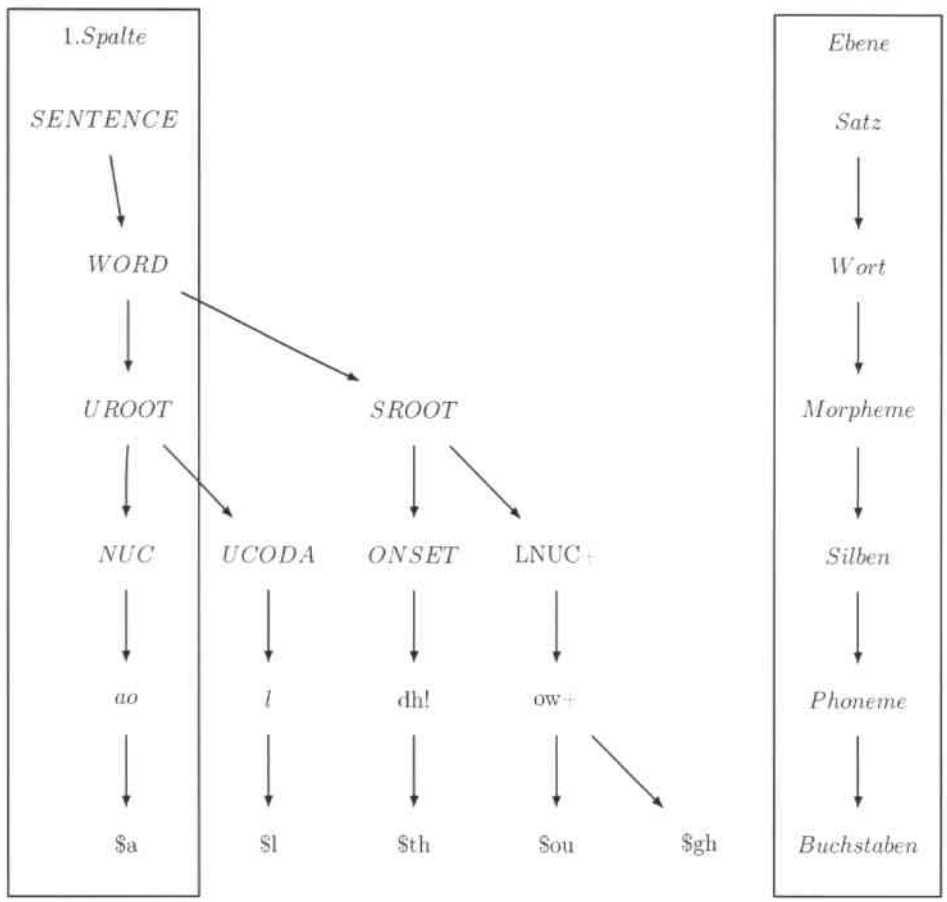

Abbildung 3.2.: Beispielaufbau für das englische Wort *although*[SLM96]

um die Wahrscheinlichkeiten für einen Parser zu trainieren. Dazu kann eine Spalte als eine Einheit angesehen werden. Als Spalte wird dabei ein Weg vom Start, hier *SENTENCE*, zu einem Zielknoten, hier *$a*, bezeichnet. Der Vorteil dieses Aufbaus liegt in der Möglichkeit, auf einer beliebigen Höhe ähnliche Eingaben verknüpfen zu können. Ist beispielsweise ein Wort samt Schreibweise bekannt, steht durch diesen Baum gleichzeitig eine Phonemsequenz zur Verfügung. Ist bei einer weiteren Hypothese dagegen nur die Phonemsequenz bekannt, können diese auf der Höhe der Phonemsequenzen miteinander kombiniert werden. Weiterhin kann ein Bigramm über die Spalten berechnet werden, welches nicht nur das Auftreten eines Buchstabens bezogen auf die Buchstabenhistorie betrachtet, sondern zusätzlich die erwähnten Merkmale wie Phoneme, Silben und Morpheme. Gleichzeitig existiert so eine Möglichkeit, Buchstaben und Phoneme ineinander zu überführen.

Im letzten Schritt wird ein erneuter Lauf mit reduziertem Suchraum gestartet. Am Ende werden die Bewertungen aller Läufe miteinander verrechnet, die beste Hypothese ausgewählt und dem Benutzer über eine Sprachsynthese-System mitgeteilt. Wird diese Hypothese bestätigt, so wird durch einen weiteren Lauf die Phonemschreibweise ermittelt und der neue Name in das System eingetragen. Lehnt der Benutzer die Hypothese ab, so kann er den Namen mit Hilfe der Telefontasten eingeben.

Das System erreicht auf einem Testset von 416 Namen, die bereits im Training vorkamen, eine Buchstabierfehlerrate von $8,4\%$ und eine Wortfehlerrate von $27,4\%$. In einem weiteren Testset mit 219 Namen, die dem System unbekannt sind, wird eine Buchstabierfehlerrate von $12,4\%$ und eine Wortfehlerrate von $46,1\%$ erreicht.

3.3. Anpassen des Sprachmodells an Äußerungen im Dialog

Gruenstein et al. stellen ein Multimodales System vor, das Auskünfte über Restaurants erteilen kann [GSW06]. Als Eingabemodalitäten werden Tastatureingaben, Spracheingaben und Gesten über einen Stift oder eine Maus unterstützt. Bei den Gesten handelt es sich um Linien, Klicks und Kreise auf einer Karte. Damit ist es möglich, bestimmte Orte oder Straßen auf einer Karte zu markieren und diese dem System mitzuteilen. Das System erlaubt auch das Setzen von Landmarken. Dazu kann mit der Maus

ein Punkt angeklickt werden und ein Text für diese Landmarke über die Tastatur eingegeben werden.

Um die möglichen Namen in einem Dialogschritt gering zu halten, werden diese entsprechend ein- und ausgelagert. Dazu werden die Namen in verschiedene Klassen, für Städte, Straßen, Restaurants und Landmarken eingeteilt. Weiterhin stehen in einer Datenbank Informationen über Orte, deren Straßen und die Restaurants in den Straßen zur Verfügung. Wird nun ein spezieller Ort ausgewählt, so werden dessen Straßen in das Sprachmodell aufgenommen. Weiterhin werden die Straßen noch nach der Anzahl der dort ansässigen Restaurants gewichtet. Wird ein Satz gesprochen und gleichzeitig ein neuer Ort auf der Karte markiert, so wird das Sprachmodell nach dem ersten Erkennerlauf verändert und auf den markierten Ort angepasst. Nun wird mit der neuen Konfiguration ein zweiter Lauf auf der alten Eingabe gestartet, um so ein besseres Ergebnis zu erhalten. Die Hintergrunddatenbank wird aus Google-Maps[1], einem Internet-Kartendienst, erstellt. Die Ausgabe des Systems erfolgt über eine Text- und eine Sprachausgabe.

Interessant an diesem System ist die Möglichkeit des dynamischen Veränderns des Sprachmodells in Abhängigkeit von Äußerungen oder Gesten im Dialog. Dazu ist aber Hintergrundwissen, wie vorhandene Straßennamen in einem bestimmten Ort, nötig, das nicht in jedem Anwendungsgebiet bereits zur Verfügung steht oder auch nicht zur Verfügung stehen kann.

3.4. Automatisches Ballen von ähnlich klingenden Äußerungsfragmenten

Park und Glass [PG06] stellen in ihrer Arbeit ein Verfahren vor, mit dem automatisch und unüberwacht Wörter in großen gesprochenen Texten, wie Vorträgen oder Vorlesungen, erkannt werden können. Dem zu Grunde liegt die Feststellung von Park und Glass, dass in einem Vortrag zwar oft nur ein kleines Vokabular, dafür aber häufig Fachbegriffe vorkommen, die im üblichen Sprachgebrauch kaum Verwendung finden. Ihr Verfahren setzt deshalb auf der akustischen Ähnlichkeit einer mehrfach gesprochenen Äußerung an. Dazu wird der Audiodatenstrom einer Vorlesung, anhand von Gesprächspausen, in kleinere Teile zerlegt. Zwischen diesen Teilen wird dann mittels des DTW-Algorithmus (engl. dynamic time war-

[1] `http://maps.google.com`

ping) [ST95, Rog05], vergleichbar mit der minimalen Editierdistanz, eine Ähnlichkeit bestimmt. Alle ähnlichen Teile werden dann mit einem Ballungsverfahren zu einer Klasse zusammengefasst. Um nun das gesprochene Wort erkennen zu können, wird auf den gefundenen Repräsentanten einer Klasse eine Phonemerkennung durchgeführt. Die resultierenden Phonemsequenzen können dann mit einem sehr großen Hintergrundwörterbuch (ca. 150.000 Wörter) verglichen werden. Die ähnlichsten Wörter aus dem Hintergrundwörterbuch gelten als die wahrscheinlichsten.

Park und Glass geben an, dass sich dieses Verfahren beispielsweise zur Informationswiedergewinnung eignet. Unter der Annahme, dass in einem Vortrag wichtige Wörter häufig wiederholt werden, können genau diese Wörter durch die beschriebene Form der Ballung ermittelt und dank des sehr großen Hintergrundwörterbuches erkannt werden.

Ob sich dieser durchaus interessante Ansatz, auch für die Erkennung von vereinzelt auftauchenden unbekannten Wörtern in einem Online-System eignet ist fraglich. Da gerade bei dem Problem des Kennenlernens umgangssprachliche Ausdrücke wie „Ich bin ..." oder „Ich heiße ..." verwendet werden und sich mit Hilfe einer grammatikbasierten Erkennung in Kombination mit einem OOV-Erkenner verhältnismäßig einfach das Auftauchen eines unbekannten Namen erkennen lässt. Das Problem des Verstehens löst der oben beschriebene Ansatz durch ein sehr großes Hintergrundwörterbuch, was für die schier unbegrenzte Anzahl an möglichen Namen keine gangbare Lösung darstellt. Denkbar ist aber, dass sich mit dem Verfahren die zeitlichen Grenzen von Namen besser bestimmen lassen. Ein Beispiel: Gesagt wurde „Name Sebastian", die Hypothese des Erkenners ist „Mein Name ist Jan". Fordert nun das Dialogsystem den Benutzer dazu auf, seinen Namen isoliert zu wiederholen, dann wird nur der Name erwartet, in dem Fall also „Sebastian". Jetzt ist dem System also bekannt wie die akustischen Merkmale des Namens in etwa „aussehen", und es kann in der ersten Aussage nach einem ähnlichen Muster suchen und so die Grenze des Namens besser bestimmen um eine bessere Hypothese zu liefern.

3.5. Nutzung gleicher Informationen verschiedener Modalitäten

Einen weiteren Ansatz, der die Redundanz zur besseren Erkennung nutzt, stellt Kaiser vor [Kai06]. In seiner Arbeit stellt er ein System vor, das

durch Fusion von Sprach- und Handschriftenerkennung den Inhalt eines sogenannten Gantt-Diagramms in Textform umwandeln kann. Dazu wird die Erstellung des Diagramms während eines Meetings aufgezeichnet und anschließend offline verarbeitet.

Für die Erkennung werden ein Handschriftenerkenner, sowie drei spezialisierte Spracherkenner verwendet. Ein Spracherkenner besitzt ein großes Vokabular und wird zur Erkennung kontinuierlich gesprochener Sprache verwendet. Ein weiterer Erkenner dient zur Erkennung von Phonemsequenzen und ein letzter zur Erkennung von gelernten Wörtern oder Phrasen.

Um die Hypothesen der Handschriftenerkennung mit den Hypothesen der Spracherkennung verknüpfen zu können, wird ein Graphem-zu-Phonem-Konverter verwendet, der die Buchstabenhypothesen des Handschriftenerkenners in Phonemsequenzen umwandelt. Werden während der Diagrammerstellung neue Begriffe eingeführt, versucht das System diese zu lernen und für den weiteren Verlauf vorzuhalten. Als Beispiel wird der Begriff *Open Source* sowohl geschrieben als auch gesprochen eingeführt. Im weiteren Verlauf werden die Abkürzungen *OSI* und *OSDL* eingeführt. Dabei werden nur die Abkürzungen aufgeschrieben, gesprochen wird dagegen *O S I Open Source Initiative* bzw. *O S D L Open Source Development Labs.* Wurde nun das erste Vorkommen von *Open Source* richtig erkannt und in das Vokabular aufgenommen, so können die folgenden Abkürzungen auch besser verstanden werden.

Dieses System ist insofern interessant, da es geschickt die in verschiedenen Modalitäten auftretende Redundanz nutzt um eine bessere Erkennungsgenauigkeit zu erreichen. Dabei dient die Handschrifterkennung als eine Art Ersatz für eine Tastatur oder Buchstabiererkennung. Weiterhin werden neu auftauchende Begriffe eingelernt um sie im späteren Verlauf für die Erkennung wiederverwenden zu können.

4. Konzepte und deren Umsetzung

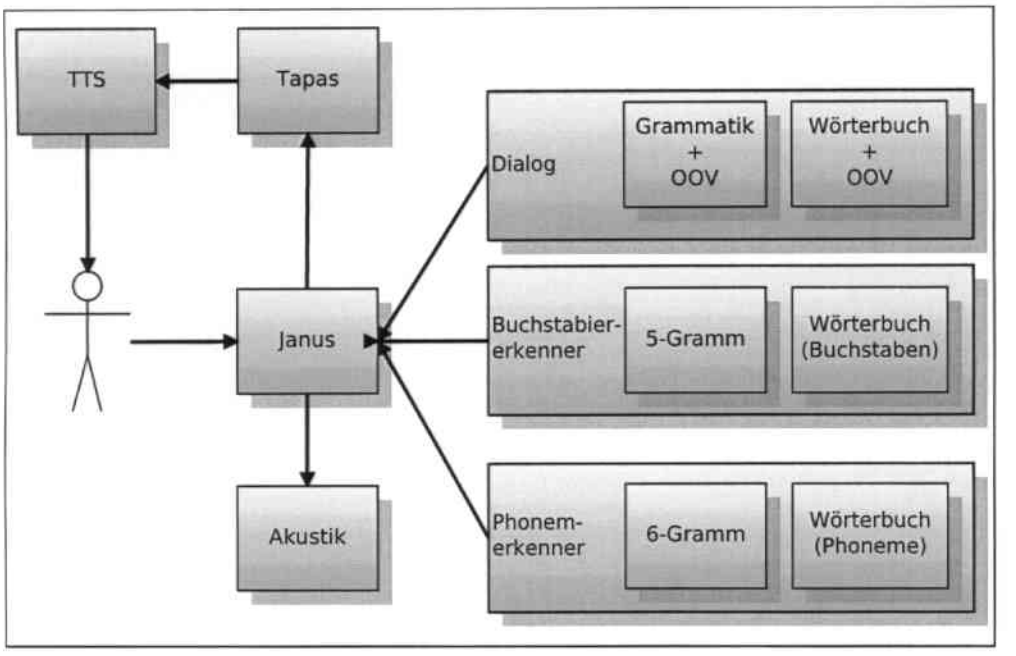

Abbildung 4.1.: Überblick über die Systemerweiterung zur Buchstabier- und Phonemerkennung

In diesem Kapitel sollen die benötigten Erweiterungen, die zur Erkennung unbekannter Wörter sowie zur Buchstabier- und Phonemerkennung notwendig sind, vorgestellt werden. Abbildung 4.1 zeigt das um diese Komponenten erweiterte Spracherkennungssystem (vgl. Abbildung 2.1). Im Folgenden soll die Erstellung der abgebildeten Komponenten besprochen werden. Dazu gehören ein OOV-Erkenner, ein Buchstabiererkenner und ein Phonemerkenner. Nicht abgebildet ist ein Graphem-zu-Phonem-Konverter, mit dessen Hilfe aus einem geschriebenen Wort eine Phonemsequenz für dessen Aussprache generiert werden kann.

4.1. Erkennung unbekannter Wörter (OOV-Modell)

Um in einer Äußerung einen unbekannten Namen erkennen zu können, wird ein OOV-Erkenner benötigt. Weiterhin muss die Grammatik um die Option ein OOV erkennen zu können erweitert werden. Abbildung 4.2 zeigt den Ausschnitt einer Grammatik, mit der an Stelle eines Namens auch die Erkennung eines OOVs erlaubt wird.

```
s[peopleid:informName,VP,_]
  ( [peopleid:informName,V,_] [peopleid:obj_person,NP,_] )

[peopleid:informName,V,_]
  ( mein vorname ist )
  ( ich bin )
  ( ich hei~se )

s[peopleid:obj_person,NP,_]
  ( [first_NT] )

[first_NT]
  ( oov )
  ( thomas )
  ( martin )
  ( markus )
  ( matthias )
  ( alexander )
```

Abbildung 4.2.: Grammatik mit OOV Erkennung in SOUP Notation

Für den englischsprachigen Erkenner wird ein vorhandenes OOV-Modell verwendet. Für das deutschsprachige System muss dagegen ein OOV-Modell erstellt werden. Dazu wird für alle Einträge aus einer Liste mit deutschen Vornamen[1] mit dem Graphem-zu-Phonem-Konverter eine Aussprachelist der Namen erstellt. Daraus werden alle Wörter mit mindestens vier Phonemen herausgesucht und die ersten vier Phoneme als Head-Modell verwendet. Als Tail-Modell werden zwei allgemeinere Modelle verwendet.

Daraus resultiert eine Liste mit 511 OOV-Modellen, die in das Wörterbuch aufgenommen werden. Abbildung 4.3 zeigt einige dieser Modelle. Auf der linken Seite befinden sich die Namen der jeweiligen Modelle. Auf

[1] Freundlicherweise erhalten von `http://www.behindthename.com`

der rechten befinden sich die Phonemsequenzen bestehend aus jeweils vier Phonemen gefolgt von zwei mit +QK bezeichneten, allgemeineren Modellen. Die Markierung WB steht für eine Wortgrenze und ist ein zusätzliches Attribut, das für das Verständniss des OOV-Erkenners ohne Bedeutung ist. Die Leistung des entwickelten OOV-Erkenners ist abhängig von der

```
{OOV:MO(A/B/I/G/+QK/+QK)}     {{A WB} B I G +QK {+QK WB}}
{OOV:MO(B/EH/AH/T/+QK/+QK)}   {{B WB} EH AH T +QK {+QK WB}}
{OOV:MO(D/IE/AH/N/+QK/+QK)}   {{D WB} IE AH N +QK {+QK WB}}
```

Abbildung 4.3.: Einige OOV-Modelle

Anzahl an zusätzlich bekannten Namen und wird deshalb im Abschnitt 5.3 besprochen. Dort wird das Erkennerverhalten mit einer unterschiedlichen Anzahl an Namen im Detail untersucht.

4.2. Erstellung der Buchstabiererkenner

Um die Schreibweise eines Namens erlernen zu können, bietet sich eine Eingabe per Buchstabierung an. In diesem Abschnitt sollen zwei Arten einen Buchstabiererkenner zu entwickeln vorgestellt und evaluiert werden.

Eine Möglichkeit besteht darin, einen grammatikbasierten Erkenner zu erstellen, eine andere ein N-Gramm zu verwenden. Der grammatikbasierte Ansatz lässt nur Buchstabiersequenzen als mögliche Hypothesen zu, die in der Grammatik definiert wurden, während ein N-Gramm Erkenner, unter Berücksichtigung von Auftretenswahrscheinlichkeiten, eine allgemeinere Buchstabierung erlaubt.

4.2.1. Grammatikbasierter Buchstabiererkenner

Für die grammatikbasierte Buchstabiererkennung sind zwei Vorgehensweisen möglich. Die erste bewertet jede Buchstabenfolge gleich wahrscheinlich, das bedeutet, dass die Hypothesen für die Buchstabierung allein durch die Bewertung des akustischen Modells entstehen. Die Zweite lässt nur Buchstabierhypothesen bekannter Wörter zu. In beiden Fällen werden einzelne Buchstaben als Wörter in die Grammatik eingetragen. Weiterhin wird für jeden Buchstaben ein Eintrag im Wörterbuch benötigt. Der Unterschied besteht in der stark eingeschränkten Perplexität, wenn nur wenige bekannte Wörter als mögliche Hypothesen erlaubt sind, gegenüber

einem allgemeinen Modell, das alle Kombinationen zulässt. Dies wirkt sich negativ auf die Erkennungsgenauigkeit und vor allem auf die Erkennungsgeschwindigkeit aus. Allerdings können mit dem reduzierten Modell keine neuen Wörter verstanden werden, was für das Erlernen neuer Wörter denkbar ungeeignet ist.

4.2.1.1. Allgemeines Modell

Ein Versuch mit einer allgemeinen Grammatik, die innerhalb eines normalen Satzes eine Buchstabiersequenz zuließ, stellte sich als ungeeignet heraus. Dabei wurden alle Buchstaben innerhalb einer erlaubten Buchstabiersequenz als gleich wahrscheinlich behandelt. Da dieses Sprachmodell keinerlei Einschränkung bezüglich möglicher Buchstabenfolgen bietet, sind die entstehenden Hypothesen kaum zu gebrauchen. Desweiteren verschlechtert sich Laufzeit des Erkenners bei manchen Buchstabiersequenzen so stark, dass ein solches Modell nicht für den Online-Betrieb geeignet ist. Abbildung 4.4 zeigt einen Ausschnitt einer solchen Grammatik.

```
s[peopleid:informNameSpelling,NP,_]
    ( [letters_NT] )
    ( *ja das w~are [letters_NT] )
    ( *ja das ist [letters_NT] )

[letters_NT]
    ( [letter_NT] )
    ( [letter_NT] [letters_NT] )

[letter_NT]
    ( a )
    ( b )
    ( c )
```

Abbildung 4.4.: Grammatik mit integrierter Buchstabiermöglichkeit in SOUP Notation

4.2.1.2. Erkennung von bekannten Namen

Die zweite Möglichkeit sieht die Erkennung von isoliert gesprochenen und dem System bereits bekannten Namen vor. Dazu wird eine Grammatik so aufgebaut, dass sie nur Buchstabenfolgen der bekannten Wörter akzeptiert. Abbildung 4.5 zeigt eine solche vollständige Grammatik für sechs unterschiedliche Namen in SOUP Notation. Die Erkennungsleistung eines

```
s[first_NT]
 ( m i c h a e l )
 ( m a x )
 ( s e b a s t i a n)
 ( u l r i k e )
 ( m o r i t z )
 ( p e t e r )
```

Abbildung 4.5.: Grammatik zur Erkennung isoliert buchstabierter und bekannter Vornamen in SOUP Notation

solchen Erkenners hängt deutlich von der Anzahl der eingetragenen Namen ab. Einerseits können bei wachsender Anzahl mehr Namen erkannt werden, andererseits nimmt die Erkennergenauigkeit durch die höhere Perplexität ab.

Dieses Verhalten wird in Abschnitt 5.3 genauer untersucht.

4.2.2. N-Gramm-basierter Buchstabiererkenner

Um den Problemen, die Grammatiken bei der Buchstabiererkennung mit sich bringen, aus dem Weg zu gehen, bietet sich ein N-Gramm an. Hier kann anhand von Beispieldaten eine Statistik über das Auftreten verschiedener Buchstabenkombinationen erstellt werden. Dadurch werden unmögliche oder sehr seltene Buchstabenfolgen mit einer geringen Wahrscheinlichkeit belegt.

4.2.2.1. Erstellung der Sprachmodelle

Für die Erstellung der N-Gramm Buchstabier-Sprachmodelle wird jeweils ein N-Gramm auf einem Textkorpus der Süddeutschen Zeitung und auf der Namensliste der Universitätsbibliothek Karlsruhe (siehe dazu auch Abschnitt 5.2) mit Hilfe des SRI-Toolkits [Sto02] berechnet. Um ein Buchstabiermodell für Vornamen zu generieren, werden die jeweils erstellten Sprachmodelle gemischt. Dazu wird für jedes Sprachmodell auf einem Evaluationsdatensatz, ebenfalls mit Hilfe des SRI-Toolkits, die Perplexität berechnet. Der Evaluationsdatensatz besteht aus 500 Vornamen, die anhand der Verteilung der Namen aus der Namensliste gezogen wurden. Es treten also häufige Namen mehrmals auf. Abbildung 4.6 verdeutlicht das Vorgehen. Durch die berechneten Perplexitäten kann eine Gewichtung λ berechnet werden, mit der die beiden Modelle gemischt werden. Dabei

wirk sich die Namensliste deutlich stärker auf das resultierende N-Gramm aus, als der SZ-Korpus.

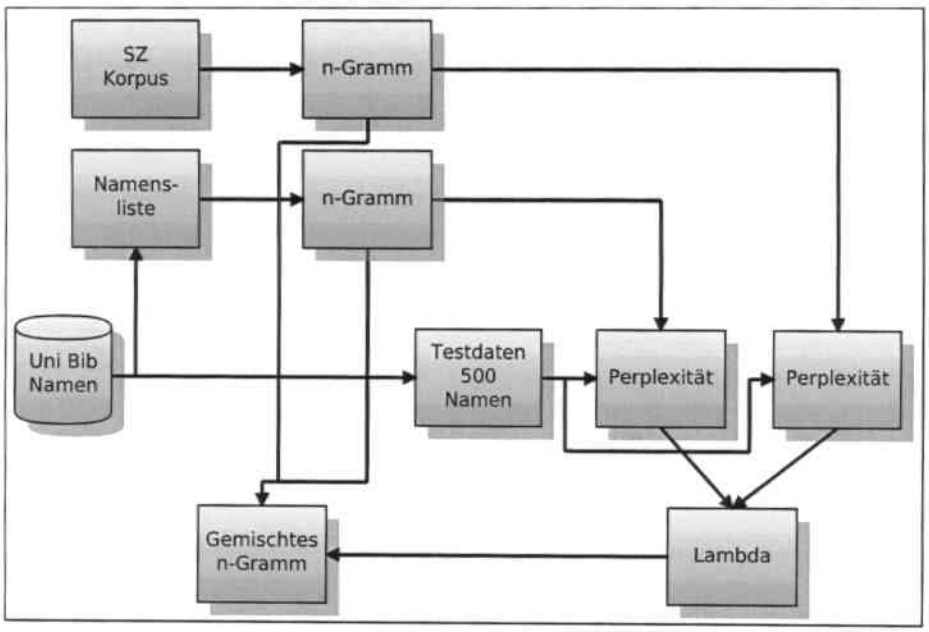

Abbildung 4.6.: Ablauf der Buchstabiersprachmodell-Generierung

Mit dem gleichen Verfahren können auch andere spezialisierte Sprachmodelle, beispielsweise für Nachnamen, Produktnamen oder Ortsnamen, erstellt werden.

Insgesamt werden auf diese Art N-Gramme der Größen drei bis sechs erstellt und anschließend auf dem Trainingsdatensatz (siehe dazu Abschnitt 5.1) evaluiert.

4.2.2.2. Evaluation

Abbildung 4.7 zeigt die Wortakkuratheit der einzelnen Modelle. Dabei schneiden das 4-Gramm und das 5-Gramm mit 83,3% gleich gut ab. Bei der Satzakkuratheit (Abbildung 4.9) erreicht das 5-Gramm mit 51,77% geringfügig bessere Werte als das 4-Gramm mit 50,59%. Bei der Wortfehlerrate (Abbildung 4.8) liegen wieder beide gleich mit je 30,71%. Das 5-Gramm ist also geringfügig besser als das 4-Gramm. Beide schneiden aber besser als das 3-Gramm und das 6-Gramm ab.

Verglichen mit anderen Buchstabiererkennern, fällt die Wortakkuratheit etwas schlechter aus, so erreicht Hild mit einem speziellen Buchsta-

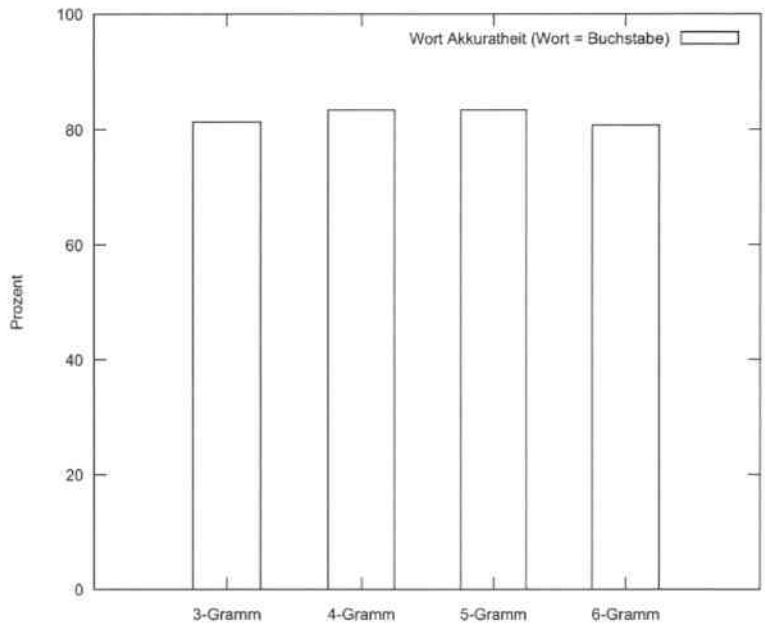

Abbildung 4.7.: Wortakkuratheit N-Gramm (Buchstabiererkenner)

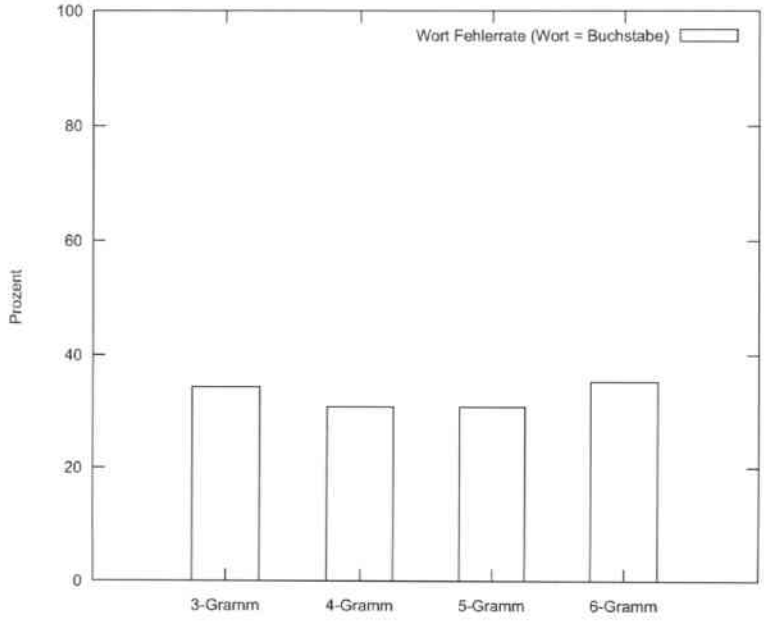

Abbildung 4.8.: Wortfehlerrate N-Gramm (Buchstabiererkenner)

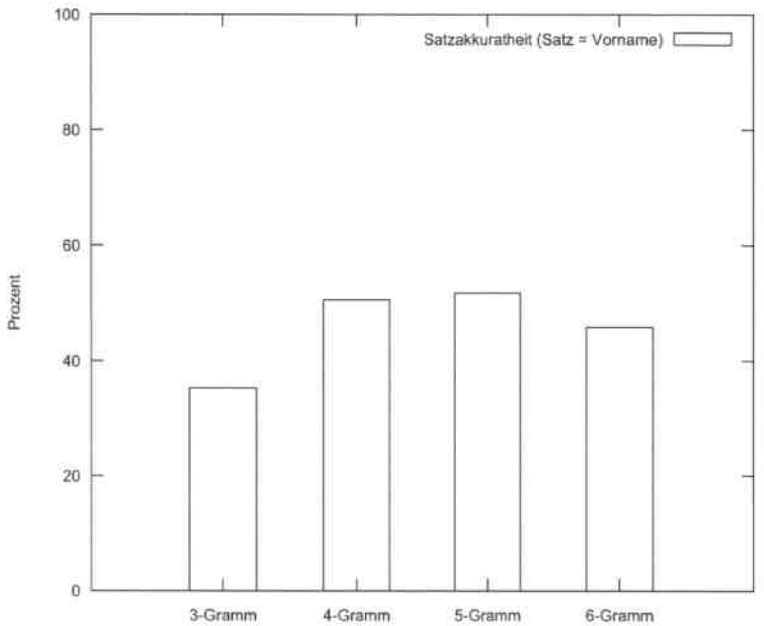

Abbildung 4.9.: Satzakkuratheit N-Gramm (Buchstabiererkenner)

biererkenner auf Basis eines MS-TDNN eine Wortakkuratheit von 92% auf sprecherunabhängigen, englischen Testdaten und 90% auf deutschen Daten [HW93]. Die Satzakkuratheit liegt von aktuellen Buchstbaiererkennern, ohne Verwendung zusätzlicher Modelle, die die Hypothesen begrenzen, bei etwa 60% [Hil97, Sch04, HSE+07].

Unter der Berücksichtigung, dass das verwendete akustische Modell kaum mit Buchstabierdaten trainiert wurde, und es sich bei den Testdaten um spontansprachliche Äußerungen handelte, erreicht der Erkenner eine akzeptable Leistung.

4.3. Graphem-zu-Phonem-Konverter

Um die Aussprache in Form einer Phonemsequenz eines buchstabierten Namens zu erhalten, wird ein Graphem-zu-Phonem-Konverter entwickelt. Weiterhin ist er nötig, um für eine sehr große Liste von Namen Aussprachen zu generieren.

In diesem Abschnitt soll der Aufbau eines solchen Konverters für die Sprachen Deutsch und Englisch beschrieben werden. Abschließend werden beide Konverter evaluiert.

Der Konverter basiert auf dem Sprachsyntheseprogramm Cepstral[2]. Dieses Programm bietet die Möglichkeit über eine Schnittstelle die generierten Phonemsequenzen eines Wortes abzufragen. Um diese Phoneme jedoch in Janus nutzen zu können, müssen sie auf dessen Phoneme abgebildet werden. Die Abbildung erfolgt mit Hilfe einer von Hand erstellten Abbildungstabelle.

Es gibt zwei mögliche Fehlerquellen, die bei der Umsetzung eines geschriebenen Wortes in eine Phonemsequenz auftreten können:

- Fehler bei der Phonemsequenzgenerierung im verwendeten Sprachsynthesesystem
- Fehler durch die Abbildung der Phonemsequenzen

Um ein Maß für die Güte des Konverters zu haben, wird eine Evaluation durchgeführt. Dazu wird das vorhandene Aussprachewörterbuch des Spracherkenners verwendet. Zuerst wird dieses Wörterbuch bereinigt, da es beispielsweise Aussprachen von Zahlen in der Form „Neunzehnhundertneunzig“ und in der Form „Eintausendneunhundertneunzig“ enthält. Weiterhin werden Aussprachealternativen von Wörtern entfernt. Die übrigen Wörter (35.535 im deutschsprachigen System und 55.411 im englischsprachigen) werden dann mit dem Sprachsyntheseprogramm in Phonemsequenzen umgewandelt. Diese generierten Sequenzen können dann mit den Referenzen aus dem Wörterbuch verglichen werden.

Die Wortakkuratheit (WAcc) des deutschsprachigen Graphem-zu-Phonem-Konverters liegt bei $89,86\%$, die des englischen bei $80,79\%$. Die Wortfehlerrate (WER) und die Satzakkuratheit (SAcc) liegt beim deutschen System bei $13,27\%$ und $46,2\%$, beim englischen bei $21,54\%$ bzw. $24,07\%$. Tabelle 4.1 fasst diese Ergebnisse noch einmal zusammen. Es bleibt zu erwähnen, dass die Referenzen nicht immer eine eindeutige Phonemschreibweise haben, wie das bereits oben erwähnte Beispiel mit den „Jahres“-Zahlen zeigt. Weiterhin stehen in dem vorhandenen Wörterbuch falsch geschriebene Wörter und auch Abkürzungen. Es kann aber vorkommen, dass Cepstral eine Abkürzung ausspricht oder ein Wort buchstabiert. Dies kann zu sehr unterschiedlichen Phonemsequenzen führen, was sich in schlechteren Ergebnissen bei obiger Evaluation widerspiegelt. Weiterhin

[2] `http://www.cepstral.com/`

	Englisch	Deutsch
Wörter im Test	55.411	35.535
WAcc	80, 79%	89, 86%
WER	21, 54%	13, 27%
SAcc	24, 07%	46, 2%

Tabelle 4.1.: Evaluationsergebnisse der beiden Graphem-zu-Phonem-Konverter

stehen im englischen Wörterbuch deutsche Straßennamen, die nicht selten Umlaute enthalten. Die englischsprachige Version des Sprachsyntheseprogramms hat allerdings Probleme diese Umlaute auszusprechen, dadurch werden unbrauchbare Phonemsequenzen erzeugt.

4.4. Phonemerkenner

Eine weitere Möglichkeit, die Aussprache in Form einer Phonemsequenz für einen neuen Namen zu ermitteln, stellt ein Phonemerkenner dar. In diesem Abschnitt soll sowohl die Erstellung als auch eine Evaluation eines Phonemerkenners erläutert werden.

Für die Erstellung eines Phonemerkenners wird ähnlich vorgegangen wie für die Erstellung des N-Gramm-basierten Buchstabiererkenners in Abschnitt 4.2.2. Dazu muss jedes Wort, das im Korpus der Süddeutschen Zeitung enthalten ist, in eine Phonemsequenz umgewandelt werden. Dies geschieht mit Hilfe eines bereits vorhandenen Spracherkennerwörterbuchs. Jedes Wort, das dort nachgeschlagen werden kann, wird in eine Trainingsdatei geschrieben. Diese dient dann zur Generierung der N-Gramme. Für die Namen aus der Namensliste wird die Phonemschreibweise mit Hilfe des in Abschnitt 4.3 beschriebenen Graphem-zu-Phonem-Konverters erzeugt. Für die Berechnung der Perplexität wird wiederum ein Testdatensatz benötigt. Dazu werden die Namen, die in der Namensliste stehen, in dem vorhandenen Wörterbuch nachgeschlagen und die dort gefundenen Phoneme verwendet. Dieses Vorgehen soll den Einfluss möglicher Fehler im Graphem-zu-Phonem-Konverter mildern. Der Testdatensatz besteht aus 449 Namen. Anhand der berechneten Perplexitäten kann wieder ein ge-

mischtes Sprachmodell erstellt werden. Für die Phonemerkennung werden N-Gramme der Größen drei bis sieben erstellt und anschließend evaluiert.

Das Wörterbuch für diesen Erkenner enthält eine Eins-zu-Eins Abbildung von dem Phonemsatz des Spracherkenners auf die Wörter des Phonemerkenners.

Um die Leistung des Phonemerkenners zu evaluieren, wird eine Grammatik erstellt, die genau die Namen enthält, die auch in den Testdaten vorkommen. Damit soll eine möglichst gute Erkennung der Namen und somit ihrer Framegrenzen sichergestellt werden. Wird nun ein Name oder ein OOV erkannt, wird auf den Framegrenzen, die der Erkenner liefert, ein neuer Erkennungslauf mit dem Phonemerkenner gestartet. Die Hypothese wird dann mit der Phonemsequenz, die der Graphem-zu-Phonem-Konverter anhand des tatsächlichen Namens liefert, verglichen. Abbildung 4.12 zeigt die Satzakkuratheit. Hier entspricht ein Satz einem Vornamen. Wäre also die Satzakkuratheit bei 100%, würde der Phonemerkenner bei jeder Eingabe genau die Phonemsequenz ausgeben, die auch der Graphem-zu-Phonem-Konverter ausgeben würde. Die Wortakkuratheit und die Wortfehlerrate der verschiedenen Erkenner sind in Abbildung 4.10 bzw. Abbildung 4.11 aufgetragen.

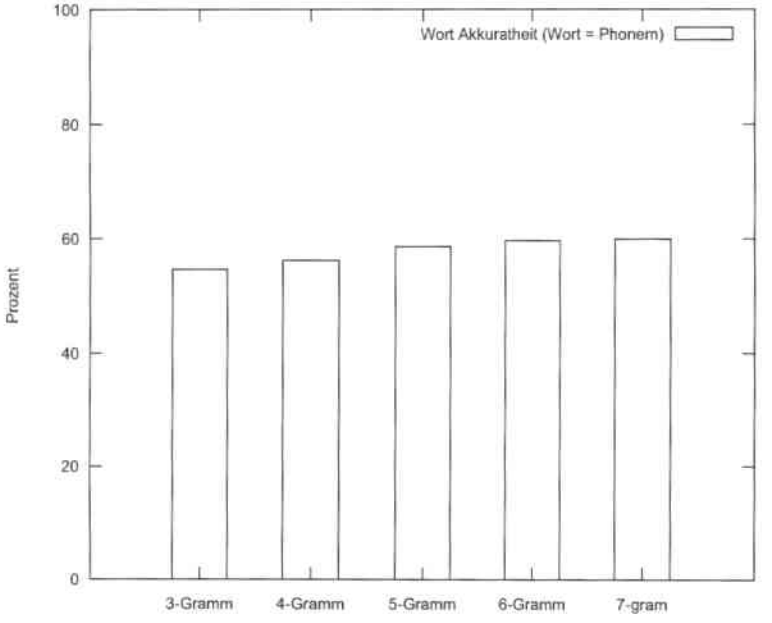

Abbildung 4.10.: Wortakkuratheit N-Gramm (Phonemerkenner)

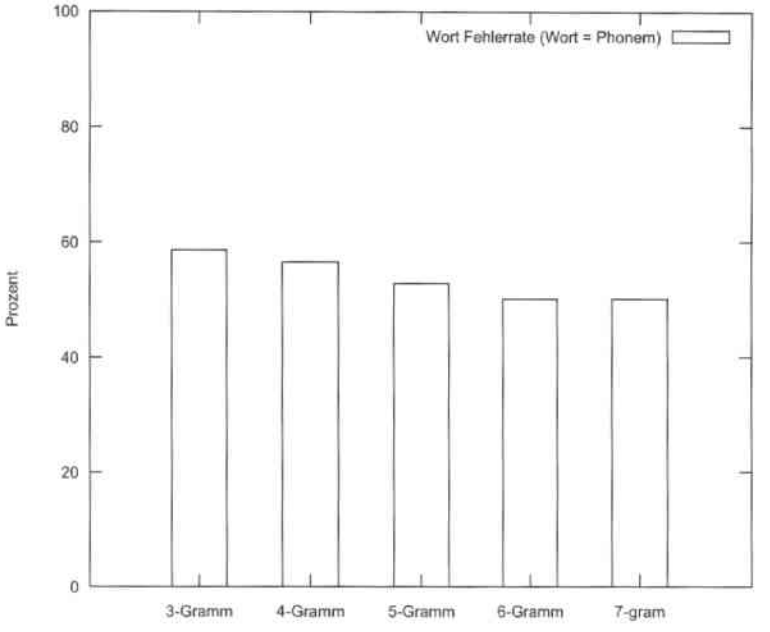

Abbildung 4.11.: Wortfehlerrate N-Gramm (Phonemerkenner)

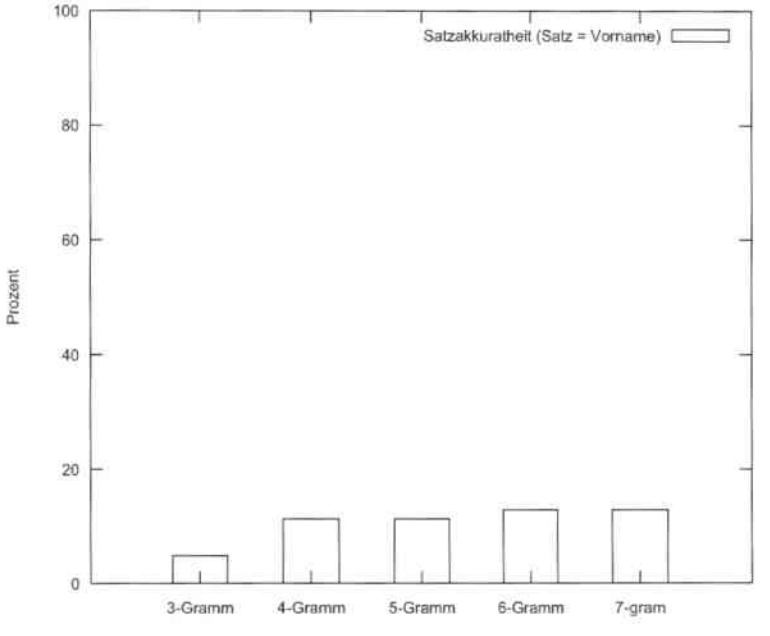

Abbildung 4.12.: Satzakkuratheit N-Gramm (Phonemerkenner)

Es zeigt sich, dass das 7-Gramm gegenüber dem 6-Gramm eine geringe Verbesserung in der Wortakkuratheit mit sich bringt (59, 95% zu 59, 68%). In den Werten für die Satzakkuratheit (12, 9%) und für die Wortfehlerrate (50, 13%) unterscheiden sich die beiden Modelle nicht. Der geringe Unterschied ist auch nicht weiter verwunderlich, da die durchschnittliche Anzahl an Phonemen für die Aussprache eines Namens – aus der vorhandenen Namensliste – bei 6, 1 Phonemen liegt und das Sprachmodell nicht über einen Namen hinausreicht. Das bedeutet, dass bereits die Aussprache häufig vorkommender Namen im 6-Gramm Sprachmodell steht. Die durchschnittliche Anzahl an Phonemen der Wörter aus dem SZ-Korpus beträgt sogar nur 4, 7 Phoneme pro Wort.

Mit knapp 60% Wortakkuratheit, ist dieser Phonemerkenner in etwa mit aktuellen Phonemerkennern zu vergleichen. So schreiben Holzapfel et al. von einer Wortakkuratheit von 65% [HSE+07]. Die Wortfehlerrate fällt dagegen, verglichen mit einem Phonemerkenner den Schaaf erwähnt (32, 8%) schlechter aus [Sch04]. Es sei an dieser Stelle allerdings noch einmal erwähnt, dass die Erkennung auf automatisch segmentierten Grenzen, der erkannten Namen, durchgeführt wurde und als Referenz die Phonemsequenzen des Graphem-zu-Phonem-Konverters verwendet wurden. Fehler die durch diese Segmentierung oder den Graphem-zu-Phonem-Konverter entstanden sind, spiegeln sich also auch in obigen Werten wieder.

4.5. Entwurf

Abbildung 4.13 zeigt den Entwurf für das Gesamtsystem. Dazu sollen mehrere Hypothesen gleichartiger Eingaben, also beispielsweise Phonemhypothesen für einen mehrfach gesprochenen Namen, zu einer besseren Hypothese fusioniert werden. Über ein mehrstufiges System soll die zu erwartende Erkennerleistung für gesprochene und bekannte Namen verbessert werden. Weiterhin soll Tapas eine Schnittstelle zur Verfügung gestellt werden, mit der neu gelernte Namen in das Spracherkennungssystem eingetragen werden können.

Es ist zu erwarten, dass die redundate Information, die in mehreren gleichartigen Eingaben steckt, genutzt werden kann, um eine bessere Hypothese zu erhalten. Eine Möglichkeit einer solchen Fusion stellt Abschnitt 5.4 im folgenden Kapitel vor.

Abbildung 4.14 zeigt den Aufbau des mehrstufigen Dekoders. Dieser soll durch mehrmaliges Dekodieren der gleichen Eingabe, unter Verwendung verschiedener Konfigurationen, die Wahrscheinlichkeit erhöhen, dass

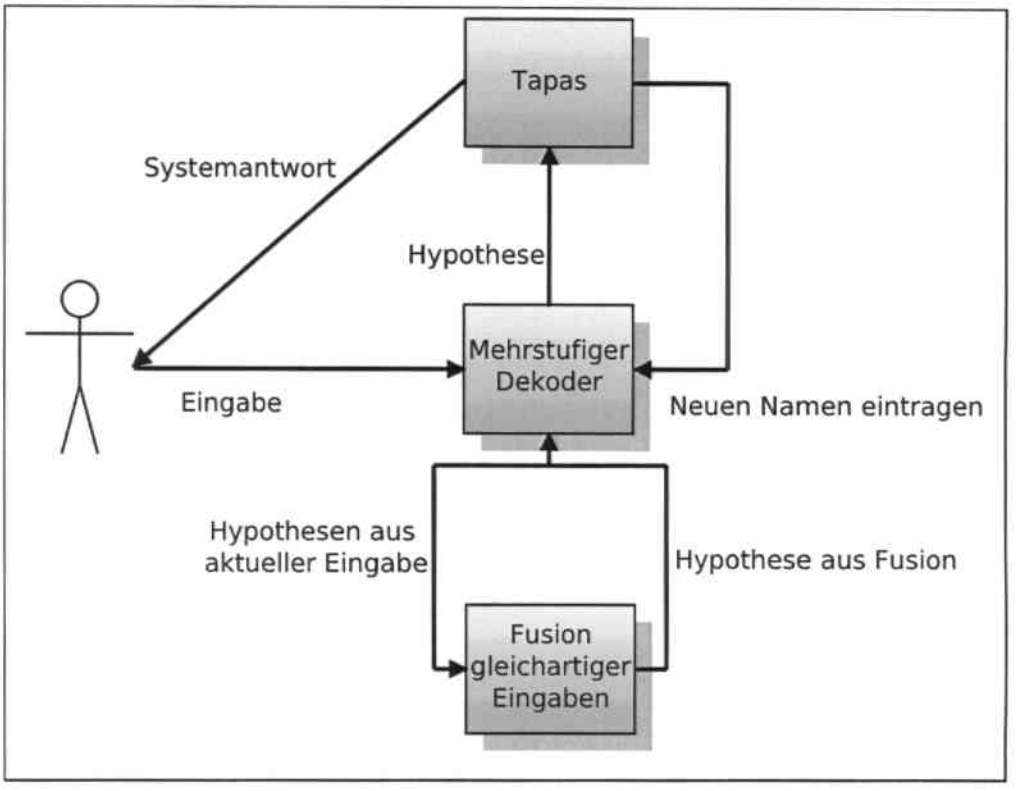

Abbildung 4.13.: Entwurf des Gesamtsystems zur Namenserkennung

ein gesprochener Name richtig erkannt wird. Damit ein gesprochener Name richtig erkannt werden kann, müssen zwei Parameter gegeneinander abgewägt werden:

- Die zu erwartende Erkennergenauigkeit (steigt, wenn wenige Namen bekannt sind)
- Die Wahrscheinlichkeit, dass der Name im Vokabular steht (steigt, bei einem großen Vokabular)

Ist beispielsweise absolut sicher, dass ein Name in einer sehr kleinen Benutzerdatenbank steht, so ist es sinnvoll, auch nur diese Namen zur Dekodierung zu verwenden und keine Liste mit beispielsweise 11.000 Namen.

Weiterhin macht es Sinn, bei einem erkannten OOV mit einem erweiterten Wortschatz, den ein Hintergrundwörterbuch zur Verfügung stellt, noch einmal zu dekodieren. Durch die größere Anzahl an Namen wird schließ-

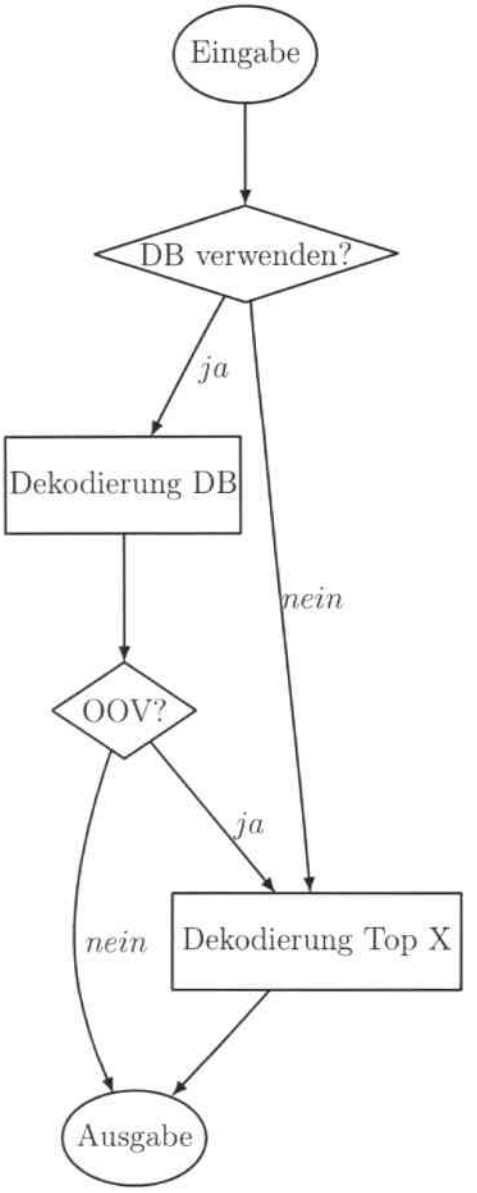

Abbildung 4.14.: Entwurf einer mehrstufigen Dekodierung für eine Spracheingabe

lich die Wahrscheinlichkeit erhöht, dass der Name dem System bekannt ist.

Wird ein Name erkannt, so wird dieser an das Dialogsystem Tapas weitergeleitet. Somit kann der Benutzer gefragt werden, ob der Name richtig verstanden wurde.

Um obigen Entwurf umsetzen zu können, sind also die folgenden Punkte zu untersuchen:

- Konfidenz des OOV-Erkenners: Ist ein erkanntes OOV auch ein OOV?
- Welche Größe einer Top-X-Namensliste bietet die beste Abwägung zwischen der Wahrscheinlichkeit, dass ein Name bekannt ist und der zu erwartenden Erkennerleistung?
- Ab welcher Wahrscheinlichkeit, dass ein Name dem System bereits bekannt ist, ist es besser zuerst auf der kleineren Benutzerdatenbank zu dekodieren?

Diese Punkte sollen in Kapitel 6 detailliert untersucht werden.

Das Eintragen eines neu gelernten Namens in den Erkenner kann durch Aufrufe bereits vorhandener Funktionen von Janus realisiert werden. Um die Aufgabe für Tapas zu erleichtern, werden diese Aufrufe in einer neuen Methode zusammengefasst. Die neue Methode benötigt dann lediglich zwei Parameter:

- Das Nichtterminal, in dem das neue Wort eingefügt werden soll, hier immer *first_NT*
- Den Namen, der gelernt wird

Die Aussprache des Namens wird dann über den Graphem-zu-Phoneme-Konverter generiert. Danach kann der Name sowohl in die Grammatik, also dem Sprachmodell, als auch in das benötigte Wörterbuch eingetragen werden. Die dazu verwendeten Methoden bieten dies sogar zur Laufzeit an, somit kann ein gelernter Name ohne Neustart des Systems für die Erkennung hinzugefügt werden. Alternativ zum Graphem-zu-Phonem-Konverter, kann auch eine Phonemsequenz mit Hilfe eines Phonemerkenners erzeugt werden.

Wie gut sich die verschiedenen Möglichkeiten der „Phonemsequenzgewinnung“ für die Wiedererkennung gelernter Namen eignen wird in Kapitel 7 untersucht.

5. Experimente

In diesem Kapitel werden, die für weitere Experimente benötigten, Testdaten beschrieben. Dazu gehört die Aufnahme und Auswahl von Sprachdaten und eine Namensstatistik. Weiterhin werden zwei Experimente durchgeführt. Das erste Experiment untersucht detailliert das Erkennerverhalten bei einer unterschiedlich großen Anzahl von bekannten Namen im Erkennervokabular. Dabei wird auch die Leistung des OOV-Erkenners betrachtet sowie die Leistung der beiden Buchstabiererkenner verglichen. Das zweite Experiment stellt ein Verfahren vor, mit dem gleichartiger Spracheingaben fusioniert werden können. Abschließend wird diese Fusion mit den Daten des Phonemerkenners sowie den Daten des Buchstabiererkenners evaluiert.

5.1. Aufnahme von Testdaten

Im Rahmen eines Dialogexperiments – in Zusammenarbeit des Lehrstuhls Waibel und dem Institut für Soziologie der Universität Karlsruhe – wurden auch Kennenlerndialoge durchgeführt, die für diese Arbeit als Trainingsdaten benutzt werden. In diesem Abschnitt soll der Ablauf des Experiments aus zwei Gründen erklärt werden. Zum einen wird dadurch aufgezeigt, wie die Testdaten zustande kamen, zum anderen dient das Experiment als Beispielszenario für das in dieser Arbeit vorgestellte System.

5.1.1. Aufbau

Für das Experiment wurde ein Roboter mit drehbarem Stereokamerakopf und einem daran befestigten Fernbesprechungsmikrofon auf dem Flur des Laborgebäudes aufgestellt. Außerdem stand ein Nahbesprechungsmikrofon bereit, das im späteren Ablauf des Experiments als Eingabemikrofon diente. Die Aufzeichnung der Daten erfolgte auf beiden Mikrofonen. Die Aufgabe der Probanden war es, einen Dialog mit dem System zu führen. Für die Probanden war allerdings nicht bekannt, dass die Verarbeitung

nicht ein Computersystem übernahm, sondern ein Mensch. Dieses Vorgehen wird auch als Wizard-of-Oz Experiment bezeichnet [Kel83]. Dadurch können Daten für ein neues Anwendungsgebiet gesammelt werden, um damit ein neues System zu entwerfen.

5.1.2. Ablauf

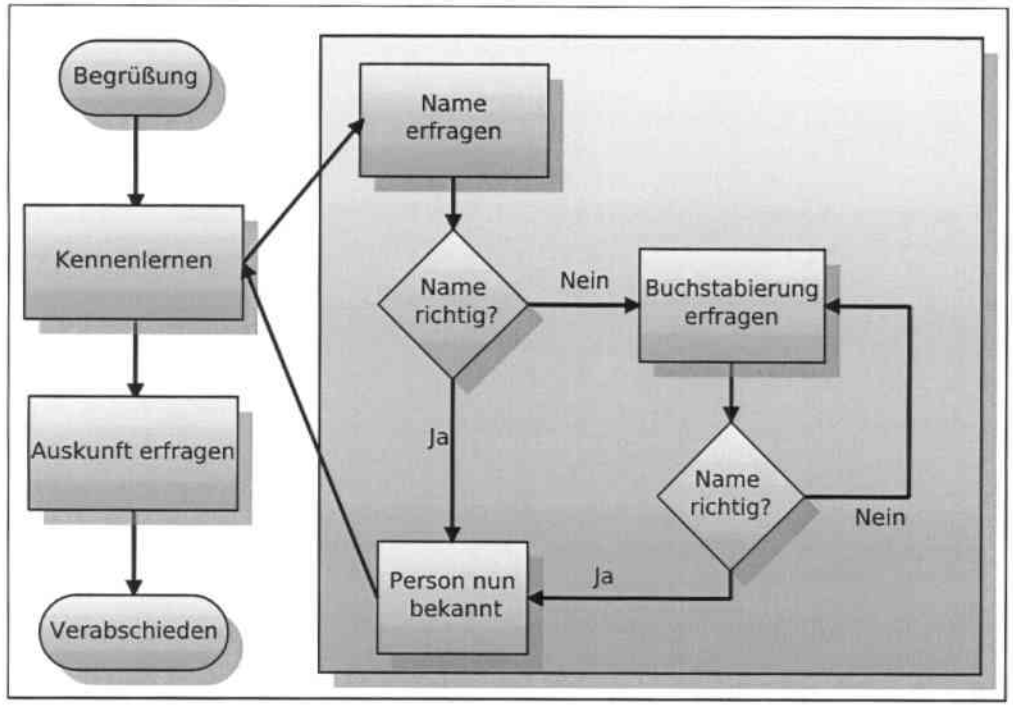

Abbildung 5.1.: Skizze des Dialogablauf

Die Versuchsteilnehmer hatten die Aufgabe die Zimmernummer einer Person bei dem System zu erfragen, um dort ein Paket abliefern zu können. Um die Zimmernummer zu erhalten, musste der Proband sich dem System erst vorstellen. Erst wenn das System den Namen des Probanden verstanden hatte, wurde ihm die Zimmernummer mitgeteilt. Abbildung 5.1 verdeutlicht den groben Ablauf des Dialogs. Weiterhin wurde das Experiment an den folgenden zwei Tagen jeweils mit denselben Probanden wiederholt.

5.1.3. Für diese Arbeit verwendete Daten

Für diese Arbeit werden nur die Benutzeräußerungen verwendet, die für das Verstehen und Erlernen von Namen wichtig sind. Dazu gehört der gesprochene Name eines Probanden, sowie die Buchstabierung des Namens. Tabelle 5.1 gibt einen Überblick über die pro Name angefallenen Sprachdaten. Dabei bezeichnet die Spalte *buchstabiert* die Anzahl der richtig und isoliert buchstabierten Namen. Die Spalte *gesprochen* gibt die Anzahl der gesprochenen Aufnahmen wieder. Dabei kann es sich um Äußerungen wie „Mein Name ist ...“ oder um isoliert gesprochene Namen handeln. Die Daten wurden von Hand segmentiert und transkribiert. Die letzte Zeile gibt die durchschnittliche Anzahl der verwendeten Aufnahmen pro Dialog an.

5.2. Namensstatistik

Aus der Benutzerdatenbank der Universitätsbibliothek Karlsruhe wurde eine Verteilung der Vornamen gewonnen. Dazu mussten die Rohdaten von Falscheinträgen, akademischen Titeln, Namenszusätzen und sonstigen ungewollten Einträgen bereinigt werden. Übrig blieb eine Datenbasis die 11.097 verschiedene Vornamen von 77.106 Personen enthält. Darunter sind auch Doppelnamen, möglicherweise falsch geschriebene Namen, sowie Falscheinträge, die nicht beim Aufräumen der Datenbank gefunden wurden. In Abbildung 5.2 ist zu sehen, wie häufig verschiedene Vornamen vorkommen. Dabei ist zu beachten, dass die X-Achse logarithmisch skaliert ist.

Aus dieser Verteilung kann nun die Dichte berechnet werden. Mit ihrer Hilfe kann die Wahrscheinlichkeit abgeschätzt werden, einen beliebigen Namen in einer Liste der X häufigsten Namen vorzufinden. Dabei wird angenommen, dass die Verteilung eine gute Schätzung für die tatsächliche Verteilung der Vornamen ist. Formel (5.1) zeigt wie diese Dichte berechnet wird, Abbildung 5.3 zeigt die Dichte.

$$p(\text{Person in Top X}) = \frac{\#\text{ Personen in Top X}}{\#\text{Personen}} \tag{5.1}$$

Dabei ist interessant, dass eine Liste mit den 1000 häufigsten Namen, den Namen einer beliebigen Person mit einer Wahrscheinlichkeit von $0,8$ enthält.

Name	**buchstabiert**	**gesprochen**
Andreas	1	1
Christian	4	2
Clemens	7	5
Dennis	9	5
Emanuel	5	4
Evelin	6	2
Fabian	7	7
Lena	1	2
Lisa	13	6
Michael	5	4
Philipp	7	5
Sebastian (zwei Sprecher)	5	6
Stefan	4	5
Ulrike	8	4
Verena	3	5
$\sum$	85	63
⌀ Aufnahmen pro Dialog	3,03	1,75

Tabelle 5.1.: Übersicht der verwendeten Testdaten

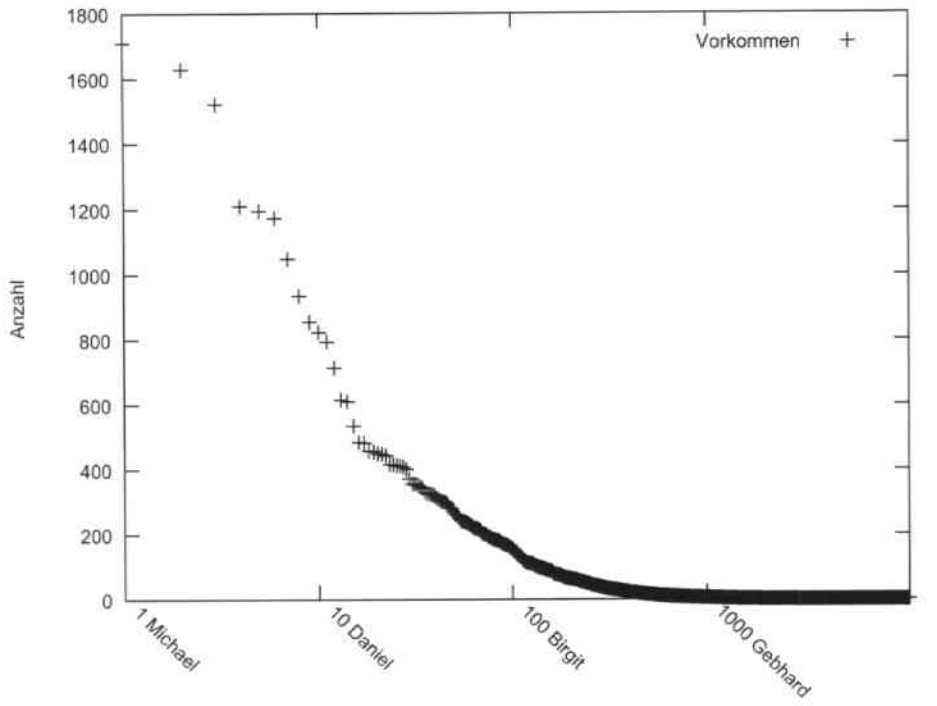

Abbildung 5.2.: Verteilung der Vornamen an der Universitätsbibliothek Karlsruhe (Stand Dezember 2006)

Weiterhin kann aus den Daten die Wahrscheinlichkeit geschätzt werden, die angibt, wie wahrscheinlich der Name einer Person, die an einem System, das eine Datenbank der Größe X verwendet, vorbei kommt, bereits in dieser Datenbank steht. Dabei wird davon ausgegangen, dass sowohl der Name der Person als auch das Zustandekommen der Datenbank obiger Verteilung folgt. Es wird also vorausgesetzt, dass das Auftreten von Personennamen unabhängig vom Inhalt der Datenbank ist. Diese Annahme trifft in der Realität oft nicht zu, da ein System meist an einem festen Ort, wie hier in einem Flur an einer Arbeitsstelle mit Büroräumen, aufgestellt wird, an dem bestimmte Personen besonders oft vorbeikommen. Diese stehen deshalb mit höherer Wahrscheinlichkeit bereits in der Datenbank. Für ein System, das an einem beliebigen Ort aufgestellt wird, kann dieser Wert aber als eine grobe, initiale Schätzung dienen. Formel (5.2)

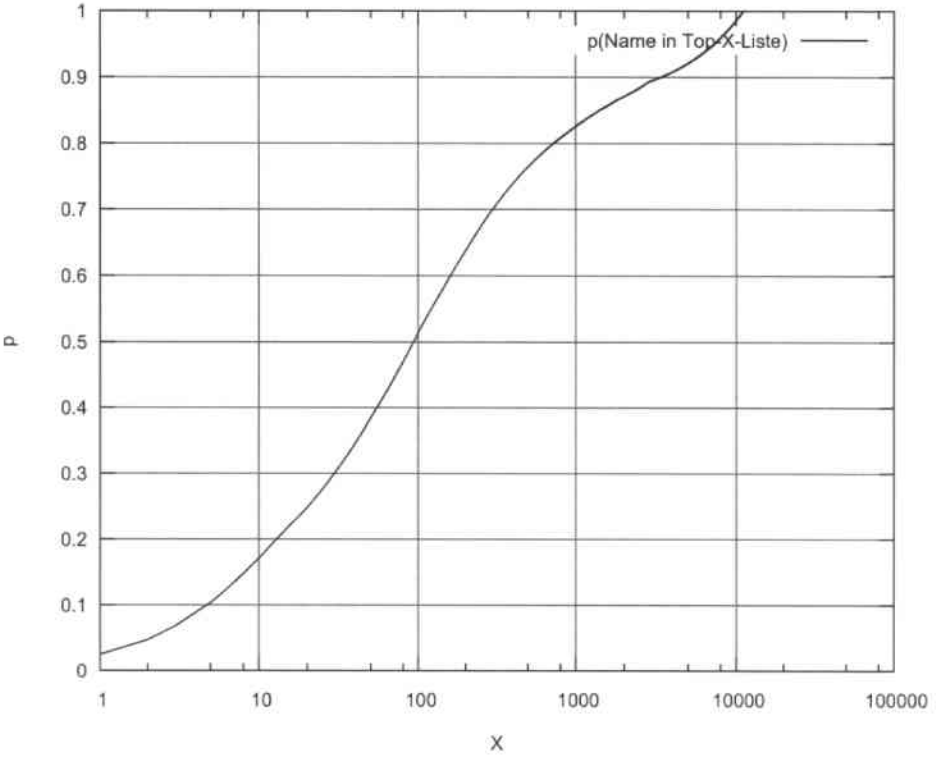

Abbildung 5.3.: Wahrscheinlichkeit, dass ein Name in einer Top-X Liste auftaucht

zeigt wie diese Schätzung berechnet wird.

$$p(\text{Name in DB}) \approx \sum_{k=0}^{n} p(N_k) \cdot (1 - (1 - p(N_k))^X) \tag{5.2}$$

Dabei bezeichnet $p(N_k)$ die Wahrscheinlichkeit, dass der Name N_k aus den insgesamt n verschiedenen Namen gezogen wird. X ist die Größe der verwendeten Datenbank. Dabei ist zu beachten, dass die Datenbank in diesem Fall nicht zwangsläufig ausschließlich verschiedene Namen enthält. Es werden also insgesamt X Personen mit zurücklegen gezogen, unter denen sich $d \leq X$ verschiedene Personennamen befinden. $1 - p(N_k)$ entspricht der Wahrscheinlichkeit, dass ein Name nicht gezogen wird, $(1 - p(N_k))^X$ gibt demnach die Wahrscheinlichkeit wieder, dass ein Name X mal *nicht* genzogen wird. $(1 - (1 - p(N_k))^X)$ gibt dann wie Wahrscheinlich an, dass

ein Name wenigstens einmal gezogen wird, wenn insgesamt X mal gezogen wird.

Das Ergebnis dieser Berechnung ist in Abbildung 5.4 zu sehen.

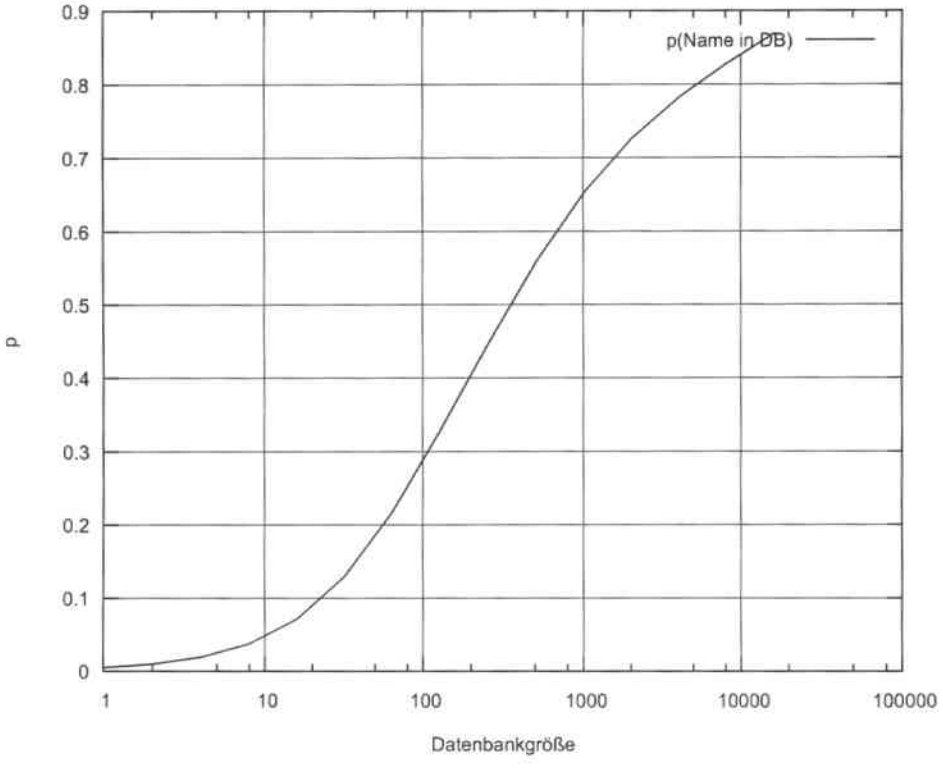

Abbildung 5.4.: Wahrscheinlichkeit, dass ein Name in einer Datenbank der Größe X steht

5.3. Erkennerverhalten bei wachsendem Namensvokabular

Ziel des Experiments ist, die Erkennungsleistung des Spracherkenners bei steigender Perplexität zu untersuchen. Zu erwarten wäre, dass bei einer steigenden Anzahl an erlaubten Namen in der Grammatik die Laufzeit steigt, sowie die Erkennungsgenauigkeit von Namen und OOVs sinkt.

5.3.1. Aufbau

Für die Erkennung der Aussprache und der Buchstabierung wird sowohl für die Aussprache als auch die Buchstabierung eine Grammatik verwendet. Die Aussprachegrammatik erlaubt verschiedene Sätze wie „Ich bin *Vorname*“ oder „Mein Name ist *Vorname*“. Für die Buchstabiererkennung wird lediglich der Name verwendet, da die Buchstabierung in den Testdaten nur isoliert gesprochen vorkommt. Dabei wird pro Erkennerlauf die gleiche Menge an Namen für die Aussprachegrammatik als auch für die Buchstabiergrammatik verwendet, wobei die Aussprachegrammatik zusätzlich ein OOV-Modell enthält, um unbekannte Wörter zu erkennen. Passend zur Grammatik wird jeweils ein Aussprachewörterbuch benötigt. Das Wörterbuch für die Aussprache wird mit Hilfe des in Abschnitt 4.3 beschriebenen, Graphem-zu-Phonem-Konverters erzeugt. Das Aussprachewörterbuch für die Buchstabierung enthält dagegen alle deutschen Buchstaben in ihrer Janus-spezifischen Phonemschreibweise.

In diesem Experiment werden zwei Parameter verändert:

- die Anzahl der bekannten Namen in der Grammatik, um die OOV-Erkennung evaluieren zu können
- die Anzahl an zusätzlichen Namen (Dummynamen), um die Perplexität des Erkenners zu erhöhen

Dabei werden die Dummynamen nach ihrer Auftretenswahrscheinlichkeit, bezüglich der berechneten Verteilung aus Abschnitt 5.2 gewählt. Allerdings werden die Namen, die in den Testdaten vorkommen, nicht in die Liste mit den Dummynamen aufgenommen. Somit wird sichergestellt, dass es für jede Größe an Dummyeinträgen auch unbekannte Namen gibt, die nicht richtig erkannt werden können.

Für dieses Experiment werden die Aufnahmen des Nahbesprechungskanals der Daten aus Abschnitt 5.1 verwendet.

5.3.2. Ablauf

Der Pseudocode in Abbildung 5.5 zeigt den Ablauf des Experiments.

Eine Besonderheit liegt in den Konfigurationen, in denen nur fünf bzw. zehn der insgesamt 15 in den Audiodateien vorkommenden Namen verwendet werden. Hier wird das Round-Robin Verfahren eingesetzt, um keine Konfiguration zu bevorzugen. Das bedeutet, dass immer 15 verschiedene Konfigurationen erstellt werden, auf denen jeweils ein Erkennungslauf

```
foreach dummySize do
        foreach windowSize do
                foreach roundRobin do
                        generate grammars
                        generate dictionary
                        decode audio
                done
        done
done
```

Abbildung 5.5.: Ablauf des Experiments in Pseudocode

durchgeführt wird. Anschaulich kann man sich dieses Verfahren so vorstellen, dass ein Fenster der Größe fünf oder zehn zyklisch über die ganze Namensliste geschoben wird, bis es nach einer Runde wieder den Anfang erreicht. Dies ergibt pro Fenstergröße jeweils 15 verschiedene Konfigurationen.

Vor jedem neuen Erkennerlauf wird also eine neue Namensliste entsprechend dem oben erwähnten Schema generiert und anschließend in die Grammatik und das Wörterbuch eingetragen. Danach wird mit dieser Konfiguration ein Erkennerlauf über alle Testdaten durchgeführt.

5.3.3. Auswertung

Für die Auswertung werden die Hypothesen des Erkenners wie in Abschnitt 2.7 beschrieben evaluiert. Für die Fenstergrößen fünf und zehn werden aus den Werten der verschiedenen Round-Robin-Konfigurationen die Mittelwerte gebildet, um Erkennerschwankungen gegenüber verschiedenen Namen auszugleichen. Weiterhin werden Namen, die zwar unterschiedlich geschrieben aber im Aussprachewörterbuch die gleiche Aussprache haben, als richtig erkannt angesehen. Dadurch wird zwar die Erkennungsleistung bezüglich der Schreibweise unter Umständen überschätzt, die Daten bleiben aber besser vergleichbar. So haben die Namen „Clemens“ und „Klemens“ die gleiche Aussprache und somit auch die gleiche Phonemfolge „{{K WB} L EH M E N {S WB}}“. Der Spracherkenner bevorzugt unter Verwendung einer Grammatik keinen der beiden Namen, es liegt also an der Implementierung welcher der beiden Namen erkannt wird.

Der Graph in Abbildung 5.6 zeigt die Wort- und Satzakkuratheit der Grammatikerkennung. In Abbildung 5.7 ist ergänzend die Wortfehlerrate eingetragen.

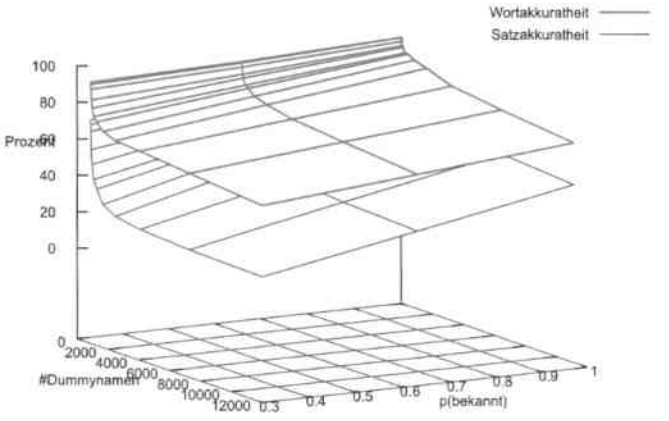

Abbildung 5.6.: Wort- und Satzakkuratheit bei der Grammatikerkennung in Abhängigkeit der Konfiguration

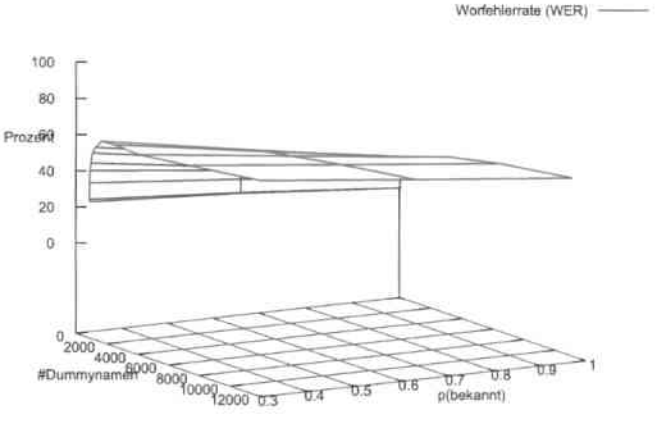

Abbildung 5.7.: Wortfehlerrate in Abhängigkeit der Konfiguration

Der Graph in Abbildung 5.8 gibt die Wort- und Satzakkuratheit für die grammatikbasierte Buchstabiererkennung an. Zur Erinnerung: Ein Wort entspricht hier einem Buchstaben und ein Satz einem Vornamen. Die Satzakkuratheit gibt also an wie oft ein buchstabierter Name komplett richtig erkannt wird. In Abbildung 5.9 wird wiederum ergänzend die Wortfehlerrate angegeben.

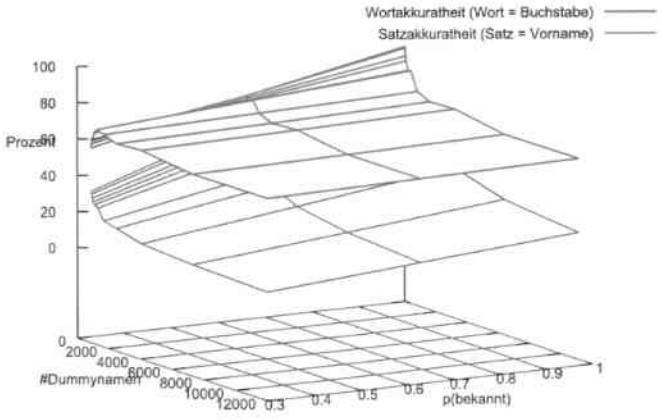

Abbildung 5.8.: Wort- und Satzakkuratheit bei der Buchstabiererkennung in Abhängigkeit der Konfiguration

In den Abbildungen 5.8 und 5.9 ist deutlich zu sehen, dass die Leistung des auf der Grammatik basierenden Buchstabiererkenners sehr stark von zwei Faktoren abhängt:

- Der Bekanntheit der zu erkennenden Namen
- Der Anzahl der Namen im Erkennervokabular

Es zeigt sich, dass bei einem sehr kleinen Erkennervokabular die bekannten Namen sehr zuverlässig und richtig erkannt werden. Unbekannte Namen können dagegen überhaupt nicht erkannt werden. Der Unterschied der Wortakkuratheit, bezogen auf die Bekanntheit eines Testnamens, fällt ebenfalls mit kleinerer Vokabulargröße. Wird die Vokabulargröße allerdings größer, gleicht sich die Wortakkuratheit an. Dies liegt an einer Zunahme von – den unbekannten Namen ähnlichen – Namen im Vokabular.

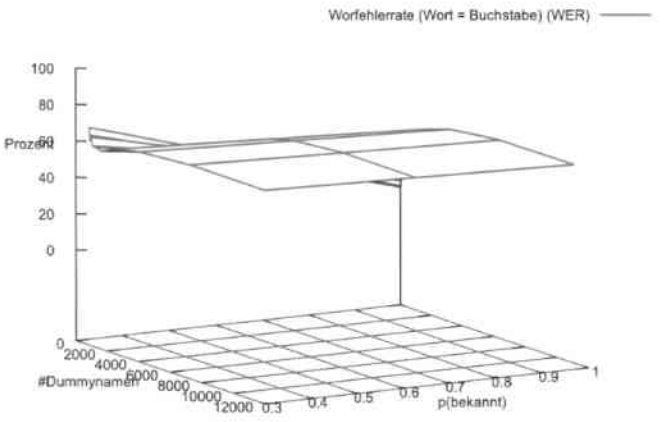

Abbildung 5.9.: Wortfehlerrate der Buchstabierung in Abhängigkeit der Konfiguration

Somit wird häufiger ein ähnlicher Name erkannt, was die Wortakkuratheit verbessert.

5.3.3.1. Vergleich N-Gramm Buchstabiererkennung mit grammatikbasierter Buchstabiererkennung

Hier soll nun der grammatikbasierte Buchstabiererkenner mit dem N-Gramm-Buchstabiererkenner verglichen werden.

Abbildung 5.10 zeigt die Wortakkuratheit der beiden Buchstabiererkenner. Da der N-Gramm-Erkenner aufgrund seines allgemeinen Aufbaus nicht von der Anzahl bekannter Namen abhängt, sind seine Werte als Ebene eingezeichnet. Schon bei einer Wortliste, die alle Namen aus den Testdaten plus weiteren 100 Dummynamen enthält, schneidet der grammatikbasierte Buchstabiererkenner schlechter ab. Fehlen hingegen bereits Namen aus den Testdaten in der Grammatik, so schneidet der N-Gramm-Erkenner grundsätzlich besser ab.

Bei der Satzakkuratheit schneidet der grammatikbasierte Erkenner immerhin bis zu einer Namensliste, die alle Namen aus dem Testset plus 400 Dummynamen enthält, besser als der N-Gramm-Erkenner ab (Abbildung 5.12). Abbildung 5.11 zeigt ergänzend die Wortfehlerrate. Tabelle 5.2

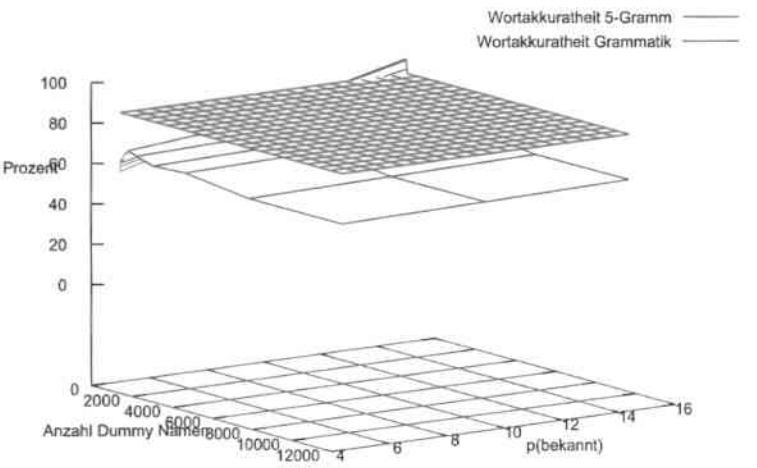

Abbildung 5.10.: Vergleich der beiden Buchstabiererkenner (Wortakkuratheit)

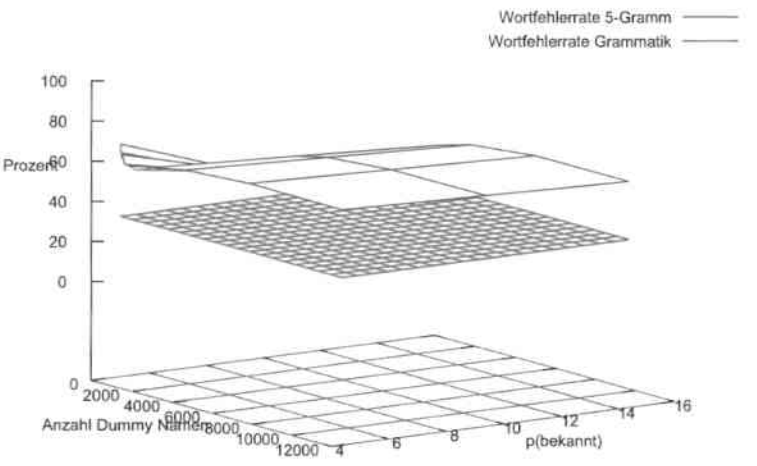

Abbildung 5.11.: Vergleich der beiden Buchstabiererkenner (Wortfehlerrate)

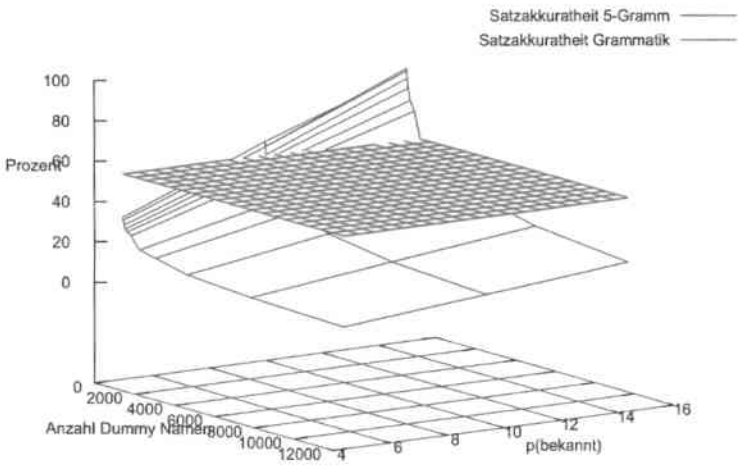

Abbildung 5.12.: Vergleich der beiden Buchstabiererkenner (Satzakkuratheit)

fasst die wichtigsten Daten zusammen. Darin sind die Satzakkuratheit (SAcc), Wortakkuratheit (WAcc) und die Wortfehlerrate (WER) für das 5-Gramm sowie für die grammatikbasierte Buchstabiererkennung aufgetragen. Für die grammatikbasierte Erkennung sind die Werte für 100 und 400 zusätzliche Namen im Vokabular eingetragen, dazu jeweils eine Konfiguration, bei der 33%, 66% bzw. 100% der Namen aus den Testdaten im Vokabular enthalten sind.

	5-Gramm	100			400		
		33%	66%	100%	33%	66%	100%
SAcc	51,77	24,78	49,65	75,29	22,28	44,00	64,71
WAcc	83,33	59,15	71,19	83,77	65,62	72,01	78,28
WER	30,71	61,65	42,53	23,78	56,93	44,12	31,27

Tabelle 5.2.: Vergleich der Erkennerdaten (Angaben in Prozent)

5.3.3.2. Detaillierte Auswertung der Erkennungsleistung

Die Erkennungsleistung des normalen Grammatikerkenners kann noch genauer ausgewertet werden. Dazu wird untersucht, ob ein erkannter Name mit dem tatsächlich gesprochenen übereinstimmt. Satzteile wie „Mein Name ist ..." werden dabei nicht berücksichtigt, da es für die Erkennung des Namens bedeutungslos ist, ob „Ich bin Sebastian" oder „Mein Name ist Sebastian" erkannt wurde. Weiterhin soll die Erkennerleistung in Abhängigkeit davon, ob der Name dem System bekannt war oder nicht, unterschieden werden. Ist der Name bekannt, sind folgende Hypothesenklassen möglich:

- Name wird richtig erkannt
- ein anderer (falscher) Name wird erkannt
- ein OOV wird fälschlicherweise erkannt
- es wird weder ein Name noch ein OOV erkannt

Ist der Name dem System dagegen unbekannt, ergeben sich die folgenden möglichen Hypothesenklassen:

- OOV wird erkannt
- ein anderer (falscher) Name wird erkannt
- es wird weder ein Name noch ein OOV erkannt

Für diese Werte kann nun eine Statistik aus den Trainingsdaten erstellt werden. Daraus lässt sich dann eine Näherung für die bedingten Wahrscheinlichkeiten der einzelnen Hypothesenklassen (HK), unter der Bedingung, dass der Name dem System bekannt $p(\text{HK}|\text{bekannt})$ bzw. unbekannt $p(\text{HK}|\text{unbekannt})$ ist, schätzen.

Die Schreibweise für die so bestimmten Wahrscheinlichkeiten zeigt Tabelle 5.3. Der Übersichtlichkeit halber sind die entsprechenden Graphen im Anhang A abgebildet.

OOV Erkennung Es zeigt sich, dass das verwendete OOV-Modell gerade bei einer kleinen Anzahl an bekannten Namen (bis ca. 20) über 60% der tatsächlichen OOV Vorkommen erkennt. Allerdings werden auch falsche Positive erkannt. Das bedeutet, es wird ein OOV an Stelle eines Wortes, das eigentlich bekannt wäre, erkannt. Dieser Wert liegt allerdings bei etwa

Bezeichnung	Erklärung
p(richtig|bekannt)	Name richtig erkannt
p(falsch|bekannt)	Falschen Namen erkannt
p(oov|bekannt)	OOV statt Namen erkannt
p(kein Name|bekannt)	Weder Name noch OOV erkannt
p(oov|unbekannt)	OOV richtig erkannt
p(falsch|unbekannt)	Namen statt OOV erkannt
p(kein Name|unbekannt)	Weder Name noch OOV erkannt

Tabelle 5.3.: Übersicht über die ermittelten Wahrscheinlichkeiten

zwei Prozent und geht bei einer Datenbankgröße von ca. 200 auf Null zurück. Das bedeutet, dass ein erkanntes OOV ab einer Datenbankgröße von ca. 200 mit sehr hoher Wahrscheinlichkeit auch ein unbekanntes Wort ist. Dabei werden allerdings nur Trainingsdaten verwendet, in denen ein Name auftaucht. Der Fall, dass bei einer Aussage des Benutzers, die keinen Namen enthält, dennoch ein Satz wie „Mein Name ist ...“ erkannt wird, bleibt in dieser Arbeit unberücksichtigt.

Erkennung von gesprochenen Namen Sind dem System alle Namen aus dem Trainingsdatensatz und weitere etwa 11.000 verschiedene Namen bekannt, wird zu etwa 51% Wahrscheinlichkeit der richtige Name und entsprechend mit 49% ein falscher Name erkannt. Sind Namen unbekannt, steigt die Erkennung eines falschen Names an Stelle eines OOVs stark an. Ab einer Größe von ca. 400 bekannten Namen gilt in etwa:

$$1 = p(\text{falsch}|\text{unbekannt}) + p(\text{oov}|\text{unbekannt}) \tag{5.3}$$

Der Fehler, dass weder ein OOV noch ein Name erkannt wird, tritt interessanterweise nur auf, wenn der gesprochene Name unbekannt ist und die Anzahl der bekannten Namen im Vokabular maximal 400 Wörter beträgt. Dies liegt daran, dass in dem größeren Vokabular ein Name enthalten sein

muss, der dem unbekannte Namen ähnlich ist, dies führt dann dazu, dass tatsächlich ein Name, wenn auch ein falscher, erkannt wird und nicht ein anderer Satz aus der Grammatik.

5.3.4. Zusammenfassung

In diesem Abschnitt wurde das Erkennerverhalten bei steigender Perplexität, also in Abhängigkeit von einer unterschiedlichen Anzahl an Namen im Vokabular, untersucht. Aus den Messwerten lässt sich nun das Erkennerverhalten bezogen auf eine Datenbankgröße schätzen. Im weiteren Verlauf dieser Arbeit, wird auf diese Werte zurückgegriffen.

5.4. Fusion mehrerer Spracheingaben

In diesem Abschnitt soll untersucht werden, wie Hypothesen aus mehreren Eingaben, also beispielsweise mehrere Phonemhypothesen, zu einer besseren Hypothese fusioniert werden können. Dazu soll eine Fusion nach Bayesschem Ansatz durchgeführt werden [KHDM98, BS06]. Dabei soll, wie auch bei den Arbeiten von Park und Glass [PG06] und Kaiser [Kai06] (vorgestellt am Ende von Kapitel 3), eine vorhandene Redundanz genutzt werden, um eine bessere Hypothese zu erhalten. Hier tritt die Redundanz allerdings nicht durch verschiedene Eingabemodalitäten auf, sondern durch wiederholtes aussprechen des gleichen Namens.

5.4.1. Definitionen

Durch das mehrfache Aussprechen eines Namens oder einer Buchstabierung sind also j-Äußerungen a_j einer Wortfolge ω eines Sprechers geben. Gesucht sei nun ebendiese Wortfolge ω. Für jede geäußerte Wortfolge liefert der Spracherkenner eine N-Bestenliste mit maximal N Hypothesen für die gesuchte Wortfolge ω. Die Hypothesen aller Äußerungen werden zu einer Liste zusammengefasst. Es ergibt sich also eine Menge mit möglichen Hypothesen mit der Größe n. Eine einzelne Hypothese der Menge wird mit ω_i bezeichnet.

5.4.2. Fusion

Gesucht ist nun die Wortfolge ω_i, die, gegeben die Äußerungen a_j, am wahrscheinlichsten ist. Nach Bayes [MS05] ergibt sich dann

$$\arg\max_i p(\omega_i|a_1,\ldots a_k) = \arg\max_i \frac{p(a_1,\ldots a_k|\omega_i)\cdot p(\omega_i)}{p(a_1,\ldots a_k)} \tag{5.4}$$

Dies kann nun, unter Annahme der bedingten Unabhängigkeit der Ereignisse a_j, zu

$$\arg\max_i p(\omega_i|a_1,\ldots,a_k) = \arg\max_i \frac{p(a_1|\omega_i)\cdot\ldots p(a_k|\omega_i)\cdot p(\omega_i)}{p(a_1,\ldots a_k)} \tag{5.5}$$

umgeformt werden. Da nach $\arg\max_i$ gesucht ist, kann der Nenner vernachlässigt werden. Es bleibt also:

$$\arg\max_i p(\omega_i|a_1,\ldots,a_k) = \arg\max_i p(a_1|\omega_i)\cdot\ldots p(a_k|\omega_i)\cdot p(\omega_i) \tag{5.6}$$

$p(a_j|\omega_i)$ stellt die Wahrscheinlichkeit dafür dar, dass die Wortfolge ω_i die Äußerung a_j erzeugt. Diese kann mit Hilfe das akustischen Modells berechnet werden. $p(\omega_i)$ bezeichnet die Wahrscheinlichkeit, dass die Wortfolge ω_i auftritt. Dies kann durch das Sprachmodell angenähert werden.

Um das Auftreten von Rundungsungenauigkeiten zu verringern, wird in der Praxis mit dem negativen Logarithmus der Wahrscheinlichkeiten gerechnet. Es ergibt sich also:

$$\arg\min_i -\log p(\omega_i|a_1,\ldots,a_k) = \arg\min_i -\log p(\omega_i) + \sum_j -\log p(a_j|w_i) \tag{5.7}$$

Es wird also die Wahrscheinlichkeit jeder Wortfolge, aus der Menge der Hypothesen, auf jeder Eingabe neu berechnet. Diejenige Wortfolge, die bei der Fusion die höchste Wahrscheinlichkeit bekommt, wird dann ausgegeben.

5.4.3. Evaluation

Um diese Fusion zu evaluieren werden zwei Experimente durchgeführt. Dabei sollen zum einen mehrere Phonemhypothesen des Phonemerkenners und zum anderen mehrere Buchstabierhypothesen des Buchstabiererkenners fusioniert werden. Dabei werden die Versuche in Abhängigkeit

eines Sprachmodellgewichts λ durchgeführt. Es ergibt sich also die neue Fusionsformel:

$$\arg\min_i -\log p(\omega_i|a_1,\ldots,a_k) = \arg\min_i \lambda \cdot -\log p(\omega_i) + \sum_j -\log p(a_j|w_i) \tag{5.8}$$

Der eingeführte Faktor λ gewichtet, also die Bewertung des Sprachmodells. Dies entspricht einer Potenzierung der Wahrscheinlichkeit $p(\omega_i)$.

5.4.3.1. Fusion der Phonemerkennerhypothesen

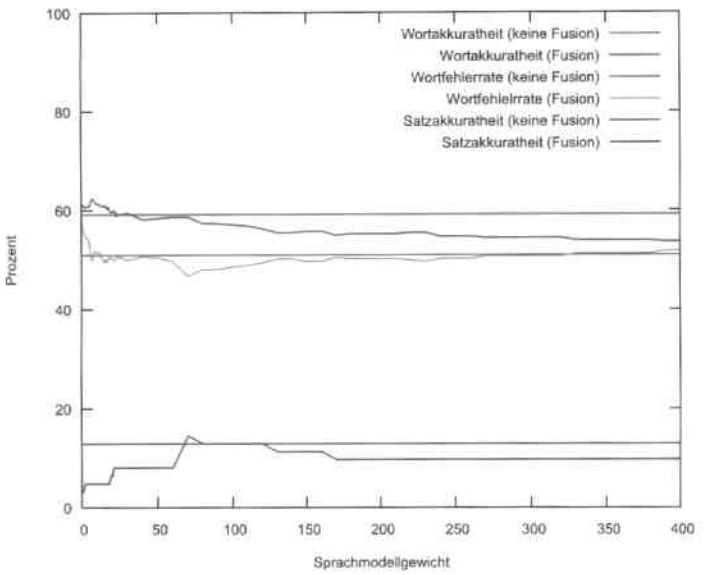

Abbildung 5.13.: Fusionsergebnisse in Abhängigkeit eines Sprachmodellgewichts (Aussprache)

Um eine Phonemerkennerhypothese zu erhalten, wird der Spracherkenner so aufgebaut, dass auf den Framegrenzen eines erkannten Namens oder einer OOV-Äußerung eine zweite Dekodierung mit dem Phonemerkenner gestartet wird. Dessen Hypothesen werden wie oben beschrieben fusioniert. Abbildung 5.13 zeigt, wie sich die Fusion der Phonemerkennung in Abhängigkeit des Sprachmodellgewichts im Vergleich zu der normalen Phonemerkennung verhält. Dabei wurde pro Äußerung einmal die beste Hypothese für diese Äußerung und einmal die beste Hypothese der Fusion aller vorheriger Äußerungen eines Dialogs für die Evaluation verwendet.

Im Durchschnitt stehen in den Testdaten pro Dialog etwa 2 Aussprachen des Namens zur Verfügung allerdings schwankt diese Anzahl stark.

Es zeigt sich, dass bei einem Sprachmodellgewicht von $\lambda = 1$, also der oben beschriebenen Fusion, zwar die Wortakkuratheit tatsächlich besser ist als ohne Fusion, die Wortfehlerrate und die Satzakkuratheit allerdings schlechter sind. Bei einem Gewicht von 7 ist sowohl die Wortfehlerrate als auch die Wortakkuratheit besser. Bei einem Sprachmodellgewicht von 70 ist sogar die Satzakkuratheit besser. Bei einer stärkeren Gewichtung verschlechtert sich die Fusion gegenüber den Einzelhypothesen. Ab einem Gewicht von 400 stellen sich keine Veränderungen mehr ein.

Als Referenz für die Phonemhypothesen dient eine mit dem Graphem-zu-Phonem-Konverter erzeugte Phonemsequenz, die bereits Fehler enthalten kann.

5.4.3.2. Fusion der Buchstabiererkennerhypothesen

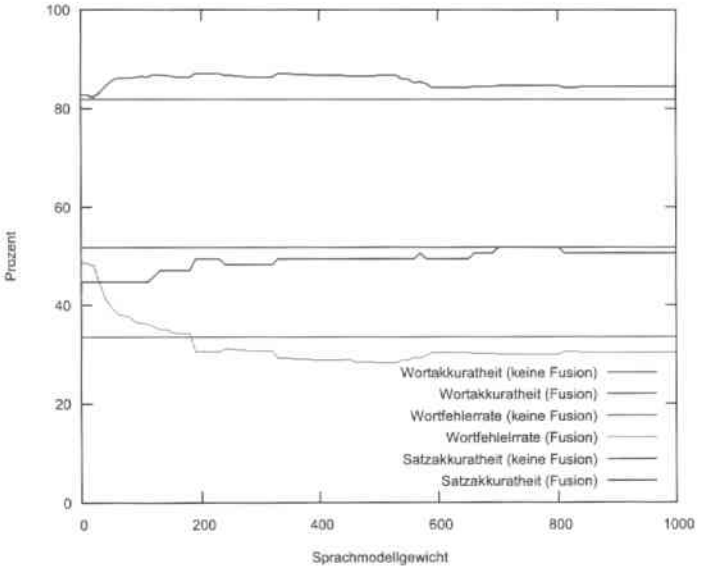

Abbildung 5.14.: Fusionsergebnisse in Abhängigkeit eines Sprachmodellgewichts (Buchstabierung)

In diesem Abschnitt soll die Fusion der Buchstabierhypothesen betrachtet werden. Abbildung 5.14 zeigt die Ergebnisse der Fusion in Abhängigkeit des Sprachmodellgewichts λ. Auch hier wird eine Fusion über alle Buchstabierungen eines Namens, innerhalb eines Dialogs berechnet. Auch

hier schwankt die Anzahl an vorhandenen Buchstabierungen in einem Dialog stark, im Durchschnitt werden aber 3 Buchstabierungen pro Dialog fusioniert.

Es zeigt sich, dass hier die Fusion ab einem Gewicht von ca. 200 bessere Werte für die Wortfehlerrate und die Wortakkuratheit als die direkte Erkennung liefert. Bei einem Gewicht zwischen 700 und 800 ist zusätzlich die Satzakkuratheit genauso gut wie ohne Fusion.

Bei der Buchstabiererkennung wurden zur Fusion allerdings nur richtig buchstabierte Eingaben für die Fusion verwendet. Würden auch noch die falsch buchstabierten Eingaben verwendet, brächte diese Funktion keine Verbesserung.

5.4.3.3. Zusammenfassung der Experimente

Es zeigt sich, dass die Fusion bei ähnlichen Eingaben zumindest die Wortfehlerrate senken und die Wortakkuratheit steigern kann. Einen Nachteil dieser Fusion stellt die benötigte Laufzeit dar, da für jede Eingabe alle Hypothesen berechnet werden müssen. Dadurch ist sie bei vielen zu fusionierenden Eingaben nicht mehr für einen Online-Betrieb geeignet.

Für die Fusion von Phonemhypothesen eignet sich das hier beschriebene Verfahren, da immer mit ähnlichen Eingaben gerechnet werden kann. Auch bei der Fusion der Buchstabierung scheint dieses Verfahren eine Verbesserung der Wortfehlerrate und der Wortakkuratheit mit sich zu bringen, allerdings nur unter der Annahme, dass alle Eingaben richtig buchstabiert wurden. In der Praxis zeigt, sich allerdings, dass beim Buchstabieren oft Fehler gemacht werden, dazu gehören Auslassungen von Buchstaben oder Buchstabendreher.

6. Auswertung der Experimente

Aus den in Abschnitt 5.3 ermittelten Daten kann nun eine Vorhersage für das zu erwartende Erkennerverhalten getroffen werden. Dazu wird bei gegebener Vokabulargröße #DB und gegebener Wahrscheinlichkeit p(bekannt), dass der Name einer Person im Erkennervokabular steht, mit Hilfe der Messwerte interpoliert. Die Wahrscheinlichkeit p(bekannt) kann allerdings nur geschätzt werden. So gibt (5.2) eine erste, sehr grobe Schätzung für diese Wahrscheinlichkeit in Abhängigkeit der Datenbankgröße. Ist allerdings mehr über die Umwelt des Systems oder den Aufbau der Datenbank bekannt, so kann dieser Wert besser geschätzt werden.

Ist in diesem Kapitel von einer *Datenbank* die Rede, sind damit die Namen der Personen gemeint, die das System bereits kennt. Dazu zählen initial eingetragene Namen ebenso wie automatisch erlernte Namen. Ist im Folgenden von einem *Hintergrundwörterbuch* die Rede, sind damit die Namen aus der Namensstatistik, besprochen in Abschnitt 5.2, gemeint. Wird eine *Top-X-Liste* erwähnt, so bezeichnet dies die X häufigsten Namen, aus dem Hintergrundwörterbuch.

Im Folgenden sollen einige Szenarien durchgespielt werden, um am Ende des Kapitels den Entwurf aus Abschnitt 4.5 zu vervollständigen.

6.1. Szenario 1: Name ist mit Sicherheit unbekannt

Für den Grenzfall, dass ein Name nicht in der Datenbank steht und dem System somit unbekannt ist (p(bekannt) = 0), macht es keinen Sinn, zuerst auf der Datenbank zu dekodieren. Vielmehr ist ein Dekoderlauf mit einer Namensliste aus dem Hintergrundwörterbuch zu wählen. Wird eine große Liste gewählt, steigt die Fehlerkennungsrate, wird eine kleine Liste gewählt, so ist der gesuchte Name unter Umständen nicht in der Top-X

Liste. Eine Liste, die den Namen mit $p(\text{Name in Top-X}) = 0,5$ enthalten soll, muss mindestens 94 Einträge haben (vgl. Abbildung 5.3). Es sei also:

- $p(\text{Name in Vokabular}) = 0,5$
- $\#\text{DB} = 94$

Aus den interpolierten Messwerten ergeben sich folgende Wahrscheinlichkeiten:

- $p(\text{richtig}|\text{bekannt}) = 0,81$
- $p(\text{falsch}|\text{bekannt}) = 0,17$
- $p(\text{oov}|\text{bekannt}) = 0,015$
- $p(\text{oov}|\text{unbekannt}) = 0,37$
- $p(\text{falsch}|\text{unbekannt}) = 0,61$

$$p(\text{Name richtig erkannt}) = p(\text{richtig}|\text{bekannt}) \cdot p(\text{bekannt}) \tag{6.1}$$

$$p(\text{Name richtig erkannt}) = 0,81 \cdot 0,5 = 0,41 \tag{6.2}$$

Es ist also zu erwarten, dass ca. 41% der Personennamen erkannt werden. Weiterhin würden in ca. 19% der Fälle ein OOV erkannt.

$$\begin{aligned} p(\text{oov}) &= p(\text{oov}|\text{bekannt}) + p(\text{oov}|\text{unbekannt}) \\ p(\text{unbekannt}) &= 1 - p(\text{Name in Vokabular}) \\ p(\text{OOV erkannt}) &= p(\text{oov}) \cdot p(\text{unbekannt}) \\ p(\text{OOV erkannt}) &= (0,37 + 0,015) \cdot 0,5 = 0,19 \end{aligned}$$

In ca. 39% der Fälle würde ein falscher Name erkannt werden. Um möglichst beim ersten Erkennungslauf eine große Erkennungsrate zu erzielen, ist also ein Abwägen zwischen der Größe des Vokabulars und der Wahrscheinlichkeit, dass ein Name bekannt ist, nötig.

Der beste Wert ergibt sich in diesem Fall unter Verwendung der Top-1047. Hier steht ein Name mit einer Wahrscheinlichkeit von knapp 0,83 im Vokabular. Zudem wird bei dieser Vokabulargröße ein bekannter Name mit einer Wahrscheinlichkeit von 0,68 richtig erkannt. Demnach würde der Name einer Person mit einer Wahrscheinlichkeit von 0,56 richtig erkannt werden. Allerdings könnten gerade noch ca. 2% der OOVs als solche erkannt werden.

6.2. Szenario 2: Name ist mit Sicherheit bekannt

Ist der Name mit Sicherheit bekannt $p(\text{Name in Vokabular}) = 1,0$, dann ist mit sehr hoher Wahrscheinlichkeit damit zu rechnen, dass auch ein Name erkannt wird. Hat die verwendete Datenbank weniger als 200 Einträge, so ist allerdings mit falschen OOV Erkennungen zu rechnen (bis zu 3%). Das bedeutet, es wird ein OOV erkannt, obwohl kein OOV vorliegt. Bei einem zweiten Lauf mit der Top-1047-Liste würden die OOVs auch nicht richtig erkannt werden können, da die richtigen Namen bereits im Vokabular vorhanden waren. Es ist dagegen anzunehmen, dass stattdessen ein falscher Name erkannt wird.

6.3. Szenario 3: Name ist mit der Wahrscheinlichkeit $p(\text{Name in DB})$ bekannt

Unter der Annahme, dass ein Name mit Wahrscheinlichkeit

$$0,0 < p(\text{Name in DB}) < 1,0$$

in der Datenbank steht, sind folgende Ergebnisse nach dem ersten Erkennungslauf zu erwarten: Es wird ein Name erkannt oder es wird ein OOV erkannt.

Für den Fall, dass ein OOV erkannt wird, ist dies mit einer Wahrscheinlichkeit von mindestens 0,97 auch tatsächlich ein unbekannter Name. In diesem Fall bietet es sich also an, einen zweiten Erkennerlauf mit der Top-1047-Liste auf den selben Eingabedaten durchzuführen. Dann ist zu erwarten, dass mit einer Wahrscheinlichkeit von 0,56 die im vorherigen Lauf als OOV erkannten Namen diesmal richtig erkannt werden.

Aus Szenario 1 ist nun bekannt, dass der Name einer Person mit einer Wahrscheinlichkeit von 0,56 richtig erkannt wird, wenn auf der Top-1047-Liste dekodiert wird.

$$p_{\text{Top-1047-Liste}}(\text{richtig}) = 0,56 \tag{6.3}$$

Weiterhin ist bekannt, wie wahrscheinlich ein bekannter Name, der in einer Datenbank der Größe S steht, richtig erkannt wird. Es gilt nun einen

Schwellwert für die Wahrscheinlichkeit $p_{\mathrm{DB}(S)}(\mathrm{bekannt})$ zu berechnen, ab dem eine bessere Erkennung auf einer vorhandenen Datenbank zu erwarten ist. Es soll also gelten:

$$p_{\mathrm{DB}(S)}(\mathrm{richtig}|\mathrm{bekannt}) \cdot p_{\mathrm{DB}(S)}(\mathrm{bekannt}) \geq p_{\text{Top-1047-Liste}}(\mathrm{richtig}) \quad (6.4)$$

Umgestellt zu

$$p_{\mathrm{DB}(S)}(\mathrm{bekannt}) \geq \frac{p_{\text{Top-1047-Liste}}(\mathrm{richtig})}{p_{\mathrm{DB}(S)}(\mathrm{richtig}|\mathrm{bekannt})} \quad (6.5)$$

und mit Hilfe der interpolierten Werte kann nun für jede Datenbankgröße ein Schwellwert für $p_{\mathrm{DB}(S)}(\mathrm{bekannt})$ berechnet werden, bei dem die zu erwartende Erkennungsleistung von der Top-1047-Erkennung besser sein sollte als die direkte Erkennung mit der Datenbank.

Es zeigt sich, dass bereits bei einer Datenbankgröße von 10 gelten sollte, dass $p_{\mathrm{DB}(10)}(\mathrm{bekannt}) \geq 0,59$ bei einer Datenbankgröße von 100 sollte $p_{\mathrm{DB}(100)}(\mathrm{bekannt})$ sogar größer als $0,69$ sein. Abbildung 6.1 zeigt den Verlauf des berechneten Schwellwerts für $p_{\mathrm{DB}(S)}(\mathrm{bekannt})$.

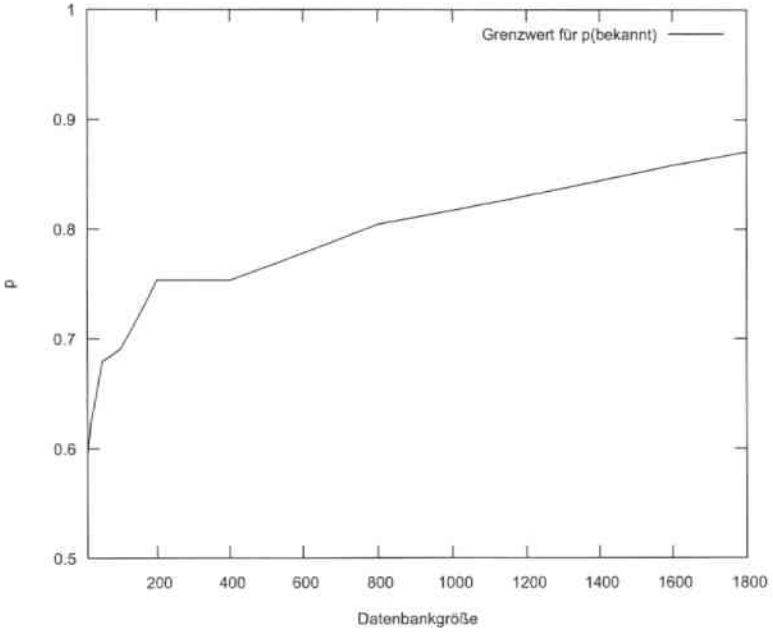

Abbildung 6.1.: Grenzwert für $p_{\mathrm{DB}(S)}(\mathrm{bekannt})$

6.4. Entwurf der Entscheidungs-Fusion

Abbildung 6.2 zeigt den Entwurf aus Abschnitt 4.5 mit den ergänzten Entscheidungskriterien. In Abhängigkeit von den beiden Parametern $\#DB$ und $p_{\mathrm{DB}(\#DB)}(\text{bekannt})$ und mittels der berechneten Kurve für den Grenzwert, wird entschieden ob im ersten Lauf erst auf der Datenbank oder gleich auf der Top-1047-Liste dekodiert wird. Ist die Wahrscheinlichkeit $p_{\mathrm{DB}(\#DB)}(\text{bekannt})$ größer als der Grenzwert, wird auf der Datenbank dekodiert. Tritt dort dennoch ein OOV auf, so handelt es sich bei der Eingabe mit hoher Wahrscheinlichkeit um einen unbekannten Namen. Bei einem zweiten Dekoderlauf mit der Top-1047-Liste wird dieser Name mit 56% richtig erkannt. In den anderen Fällen wird mit hoher Wahrscheinlichkeit ein Name erkannt. Sollte der gesuchte Name auch hier nicht in der Liste enthalten sein, wird nur noch in 2% der Fälle ein OOV erkannt.

6.5. Evaluation der Entscheidungs-Fusion

In diesem Abschnitt soll eine erste Evaluation, des oben beschriebenen Verfahrens vorgestellt werden. Dazu sollen folgende Punkte untersucht werden:

- Wie verhalten sich die Evaluationswerte bezogen auf die Grammatikerkennung?
- Wie verändern sich die Hypothesen für einen Namen?

6.5.1. Aufbau der Experimente

Für dieses Experiment ist eine initiale Datenbasis nötig. Um keine bestimmten Namen zu bevorzugen sollen, unter der Annahme, dass das System bereits die Mitarbeiter des Lehrstuhles an dem es entwickelt wurde kennt, ebendiese die initiale Datenbasis bilden. Es handelt sich dabei um 48 verschiedene Vornamen aller ISL-Mitarbeiter. Tabelle 6.1 listet diese Namen auf. Wird weiter angenommen, dass sich hauptsächlich Besucher mit dem System unterhalten, können wieder die Testdaten aus Abschnitt 5.1 verwendet werden. Dabei handelt es sich um 15 verschiedene Namen und insgesamt 63 Testsätzen.

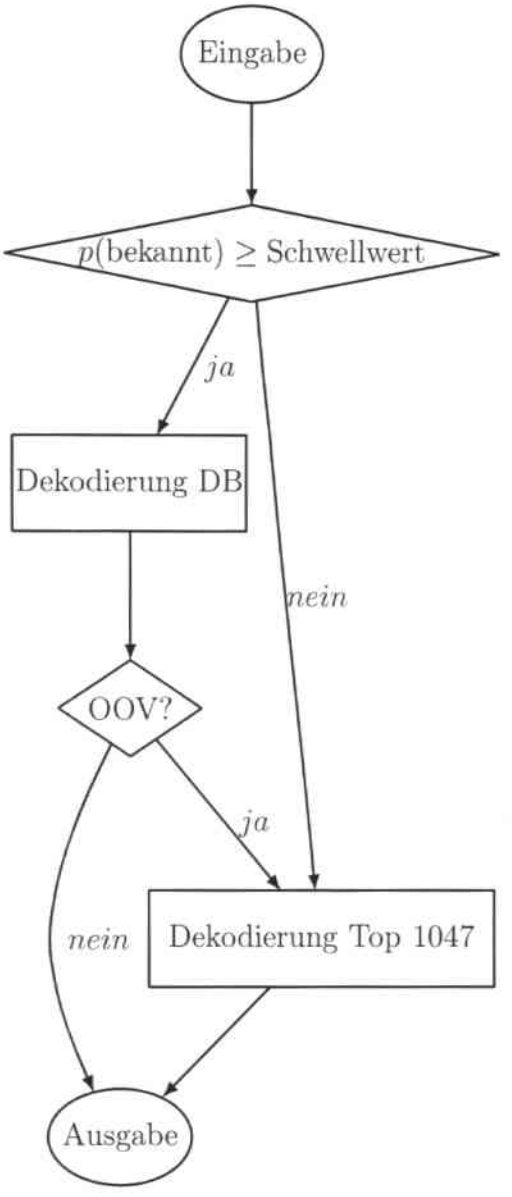

Abbildung 6.2.: Aufbau der Entscheidungsfusion

Alex	Annette	Anthony	Ashish	Bing
Chiori	Christian	Dan	Florian	Freya
Hartwig	Hazim_Kemal	Ian	Isaac	Jie
Kai	Keni	Kenichi	Kornel	Linda
Lisa	Margit	Maria	Mark	Matthew
Matthias	Michael	Mohamed	Muntsin	Nimish
Norbert	Paisarn	Qin	Rainer	Roger
Sameer	Sanijika	Sebastian	Sharath	Silke
Stephan	Stephen	Susanne	Szu_Chen	Tanja
Udhyakumar	Yik_Cheung	Ying_Joy		

Tabelle 6.1.: Alle Vornamen der ISL-Mitarbeiter (Stand 14.06.2007)

6.5.2. Es sind wenig Namen aus den Testdaten bekannt

Nach der oben beschriebenen Entscheidungsfusion, liegt der Grenzwert für eine Datenbank der Größe 48 bei etwa 67%. Wird $p_{\mathrm{DB}(48)}(\text{bekannt})$ also kleiner geschätzt, würde eine erste Dekodierung auf der Top-1047-Liste ein besseres Ergebniss versprechen. Tatsächliche enthalten 37% der Testdaten einen Namen der bereits in der Datenbank steht. Tabelle 6.2 gibt die Evaluationswerte für eine direkte Dekodierung auf der Top-1047-Liste und für eine zweistufige Dekodierung, erst auf der Datenbank und, bei erkanntem OOV, auf der Top-1047-Liste an. Zum Vergleich sind in der letzten Spalte der Tabelle die Werte ohne eine OOV-Erkennung eingetragen. Für die Evaluation wurden Homophone unter den Namen, wie in 5.3, zugunsten der richtigen Schreibweise ersetzt.

	Top-1047-Liste	**DB dann Top-1047-Liste**	**Kein OOV**
SAcc	**68, 3%**	52, 4%	34, 9%
WAcc	**86, 3%**	77, 7%	69, 8%
WER	**30, 9%**	41, 7%	56, 1%

Tabelle 6.2.: Evaluationswerte auf der Grammatikerkennung

Um die Qualität der erkannten Namen zu untersuchen, sollen diese separat evaluiert werden. Dazu wird die minimale Editierdistanz zwischen dem Referenznamen und dem Namen aus der Hypothese berechnet (Tabelle 6.3). Zur Erinnerung, hier entspricht ein Wort bezogen auf die Evalu-

ation einem Buchstaben und ein Satz entspricht einem Namen. Eine Satzakkuratheit von 71,5% bedeutet also, dass von allen Testdaten 71,5% der Namen richtig vertsanden wurden. Es zeigt sich, dass sowohl die Evalua-

	Top-1047-Liste	**DB dann Top-1047-Liste**	**Kein OOV**
SAcc	**71,5%**	54,0%	34,9%
WAcc	**86,2%**	73,1%	57,7%
WER	**15,6%**	30,9%	47,2%

Tabelle 6.3.: Evaluationswerte der erkannten Namen (Wort=Buchstabe)

tionswerte für die Grammatikerkennung als auch für die Hypothesen der einzelnen Namen deutlich besser ausfallen, wenn direkt auf der Top-1047-Liste dekodiert wird. Für diese Unterschiede sind hauptsächlich zwei Faktoren verantwortlich. Zum einen, ist, gerade bei kleineren Datenbanken, mit etwa 4% Wahrscheinlichkeit damit zu rechnen, dass weder ein Name noch ein OOV erkannt wird, wenn ein unbekannter Name vorliegt, zum anderen werden etwa 44% der OOVs als Name erkannt. In beiden Fällen wird keine zweite Dekodierung mit der Top-1047-Liste durchgeführt.

6.5.3. Es sind viele Namen aus den Testdaten bekannt

In einem weiteren Experiment werden *zusätzlich* Namen aus den Testdaten in die Datenbank aufgenommen. Dabei handelt es sich um all die Namen, für die es wenigstens fünf gesprochene Testdatensätzen gibt. Dies geschieht unter der Annahme, dass häufig auftretende Namen bereits gelernt wurden und sich somit ebenfalls in der Datenbank befinden. In diesem Experimente handelt es sich dabei um die Namen: Clemens, Dennis, Fabian, Philipp, Stefan und Verena.

Die Datenbank enthält nun 54 verschiedene Namen. Für diese Größe gilt ein Grenzwert von $p_{\mathrm{DB}(54)}(\text{bekannt}) \approx 0,68$. Tatsächlich sind 79% der Testdaten bekannt. Dieser Wert liegt also über dem berechneten Grenzwert. Deshalb sollte die Strategie, erst auf der Datenbank dekodieren und dann die Top-1047-Liste zu verwenden, besser abschneiden, als direkt mit der Top-1047-Liste zu beginnen.

Tabelle 6.4 gibt die Ergebnisse für die Grammatikerkennung wieder. In Tabelle 6.5 stehen die Ergebnisse für die Namenserkennung. Auch hier sei wieder angemerkt, dass ein Satz einem Vornamen entspricht. Durch die zweite Dekodierung, ausgelöst durch eine erkanntes OOV, lässt sich

	Top-1047-Liste	DB dann Top-1047-Liste	Kein OOV
SAcc	68,3%	**69,8%**	66,7%
WAcc	86,3%	**87,8%**	87,1%
WER	30,9%	**25,2%**	28,8%

Tabelle 6.4.: Evaluationswerte auf der Grammatikerkennung

	Top-1047-Liste	DB dann Top-1047-Liste	Kein OOV
SAcc	71,5%	**74,6%**	73,0%
WAcc	86,2%	**87,2%**	85,2%
WER	15,6%	**15,6%**	18,5%

Tabelle 6.5.: Evaluationswerte der erkannten Namen (Wort=Buchstabe)

auch hier die Erkennerleistung steigern. Durch die hohe Wahrscheinlichkeit, dass ein Name aus dem Testset in der Datenbank steht, kann der Vorteil einer kleineren Perplexität genutzt werden. So werden schon bei dem ersten Durchlauf 73% der Namen richtig erkannt, was die Spalte *Kein OOV* der Tabelle 6.5 zeigt. Durch den zweiten Durchlauf lässt sich dies auf 74,6% steigern. Würde in diesem Fall zuerst die Top-1047-Liste verwendet, wäre das Ergebnis sogar schlechter als bei einem Lauf ohne OOV-Erkennung.

6.6. Zusammenfassung

In diesem Kapitel wurde untersucht, was für eine Erkennungsleistung zu erwarten ist, wenn eine Datenbankgröße sowie eine Wahrscheinlichkeit, dass die Person in dieser Datenbank steht, gegeben ist. Erste Experimente mit dem Umgesetzten Entwurf, bestätigt die theoretisch ermittelten Werte.

Es zeigt sich, dass für eine gegebene Datenbankgröße ein Schwellwert für $p_{\mathrm{DB}(S)}(\text{bekannt})$ berechnet werden kann, bis zu dem die Erkennung auf der Top-1047-Liste bessere Erkennungsergebnisse liefern sollte. In jedem Fall ist ein zweiter Lauf bei einem erkannten OOV mit der Top-1047-Liste sinnvoll, da es sich mit hoher Wahrscheinlichkeit um einen tatsächlich unbekannten Namen handelt. Nach dem Top-1047-Lauf können so gut wie keine OOVs mehr erkannt werden, es werden also bevorzugt falsche Namen an Stelle eines OOVs erkannt. In diesem Fall kann mit dem mehrstufigen

System auf nur einer Eingabe kein wesentlich besseres Ergebnis mehr erzielt werden, da dem System ohne weitere Eingaben nicht bekannt ist, ob der erkannte Name richtig oder falsch erkannt wurde. Da bei einer zunehmenden Vokabulargröße auch die Anzahl an ähnlich klingenden Namen zunimmt, besteht auch zunehmend die Gefahr, dass ein solcher Namen, fälschlicherweise, eine bessere Bewertung bekommt, als der tatsächlich gesprochene.

7. Wiedererkennen gelernter Namen

Wird ein neuer Name, der vorher nicht in der Datenbank war, erkannt und vom Benutzer bestätigt, wird dieser sowohl in die Datenbank und somit in die Grammatik, als auch in das Wörterbuch aufgenommen. Wurde der Name unter Verwendung eines Hintergrundwörterbuches erkannt, so ist auch bereits eine Phonemschreibweise für ihn bekannt. Wurde der Name hingegen durch einen Lauf des Buchstabiererkenners ermittelt, so muss für diesen eine Phonemschreibweise gefunden werden. Eine Möglichkeit besteht darin, den Graphem-zu-Phonem-Konverter zu verwenden. Eine weitere Möglichkeit ist die Verwendung der Phonemhypothese aus der Bayesfusion (siehe Abschnitt 5.4). In diesem Abschnitt sollen beide Verfahren miteinander verglichen werden.

7.1. Aufbau und Ablauf

Für jede Art der Phonemsequenzgenerierung wird ein Testlauf durchgeführt. Dazu wird der zu evaluierende Name mit der generierten Phonemsequenz und 20, 100 bzw. 1000 zusätzlichen Namen in den Erkennerwortschatz eingetragen. Anschließend werden alle Äußerungen, die den Namen enthalten, als Testdaten verwendet.

Als Evaluationskriterium wird der Anteil der richtig erkannten Namen, der Anteil an fälschlicherweise erkannten OOVs sowie der Anteil an sonstigen Falscherkennungen berechnet. Existieren für einen Namen mehrere mögliche Phonemsequenzen, so werden diese einzeln evaluiert und daraus der Mittelwert berechnet. Dieser Fall tritt beispielsweise bei der Phonemfusion ein. Die Fusion wird immer über einen Dialog berechnet. Das bedeutet, existieren mehrere Dialoge einer Person oder mehrerer Personen mit dem gleichen Namen, so wird pro Dialog eine Phonemfusion berechnet. Pro Name wird dann aus den Evaluationswerten aller Testdaten wiederum ein Mittelwert berechnet. Es existiert dann also ein Wert, der angibt, wie gut ein bestimmter Name bezogen auf eine gegebene Art der Phonemsequen-

zerstellung wiedererkannt werden kann. Werden diese Werte der Namen wieder gemittelt, so erhält man eine Aussage darüber, wie gut mit der Art der Phonemsequenzerstellung, Namen wiedererkannt werden können.

Für die Experimente wurden vier verschiedene Methoden ausprobiert. Zum einen der Graphem-zu-Phonem-Konverter, der aus den richtig geschriebenen Namen eine Phonemsequenz erzeugt. Des weiteren die Ergebnisse der Phonemfusion mit einem Sprachmodellgewicht von 70, 7 und 1. Dabei handelt es sich um die Gewichte, bei denen die Phonemfusion ihre besten Werte annimmt bzw. der Theorie entspricht. Zur Erinnerung, nach der Theorie wäre ein Sprachmodellgewicht von 1 zu verwenden. Nach den Experimenten, beschrieben in Abschnitt 5.4, stellte sich heraus, dass bei einem λ von 7 sich die besten Werte sowohl für die Wortfehlerrate als auch die Wortakkuratheit ergeben und dass bei 70 die beste Satzakkuratheit zu erwarten ist.

7.2. Ergebnis

Tabelle 7.1 zeigt die Ergebnisse der oben beschriebenen Evaluation, bei einer Anzahl von 20 zusätzlichen Dummynamen. Tabelle 7.2 zeigt die Ergebnisse bei 100 und die Tabelle 7.3 mit 1000 zusätzlichen Dummynamen.

Phonemgewinnung	richtig erkannt	OOV erkannt	falsch erkannt
Graphem zu Phonem	67%	6%	27%
Fusion 70	68%	9%	23%
Fusion 7	**75%**	7%	18%
Fusion 1	73%	8%	19%

Tabelle 7.1.: Wiedererkennung gelernter Wörter mit 20 Dummynamen

Es zeigt sich, dass die Fusion mit dem Sprachmodellgewicht von 70 immer schlechter abschneidet als die Sequenzen, die aus den anderen beiden Fusionen hervorgehen. Sind nur 20 oder 100 Namen zusätzlich eingetragen, eignen sich die Fusionssequenzen 7 und 1 sogar besser zur Wiedererkennung, als die Sequenzen aus dem Graphem-zu-Phonem-Konverter.

Phonemgewinnung	richtig erkannt	OOV erkannt	falsch erkannt
Graphem zu Phonem	63%	1%	36%
Fusion 70	62%	1%	37%
Fusion 7	69%	3%	28%
Fusion 1	**73%**	4%	23%

Tabelle 7.2.: Wiedererkennung gelernter Wörter mit 100 Dummynamen

Phonemgewinnung	richtig erkannt	OOV erkannt	falsch erkannt
Graphem zu Phonem	**46%**	0%	54%
Fusion 70	37%	0%	63%
Fusion 7	41%	0%	59%
Fusion 1	41%	0%	59%

Tabelle 7.3.: Wiedererkennung gelernter Wörter mit 1000 Dummynamen

Die Phonemsequenzen, die durch Fusion mehrerer Spracheingaben eines Namens gewonnen wurden, eignen sich also zum Eintrag in ein Wörterbuch. Der Vorteil dieses Verfahrens liegt darin, dass keine Information über die Schreibweise des Namens vorhanden sein muss um einen gesprochenen Namen wiedererkennen zu können. Ein mehrmaliges Aussprechen des Namens genügt demnach. Allerdings ist zu beachten, dass die fusionierten Phonemsequenzen auf Teilmengen der Daten berechnet wurden, die später auch im Test verwendet wurden. Eine Evaluation mit einer größeren Datenmenge, die ein Aufteilen in Lern- und Evaluationsdaten erlaubt, wäre für dieses Experiment sicher aussagekräftiger. Weiterhin sollte beachtet werden, dass die Phonemsequenzen, die mit dem Graphem-zu-Phonem-Konverter ermittelt wurden, anhand der richtigen Schreibweise eines Namens erstellt wurden. Wird ein Name z. B. durch mehrmaliges

Buchstabieren nicht richtig verstanden, so kann auch nicht die *richtige* Phonemsequenz gefunden werden.

8. Ergebnis und Diskussion

In dieser Arbeit wurde ein System vorgestellt, das es ermöglicht, unbekannte Namen zu lernen. Dazu wird einerseits auf ein Hintergrundwörterbuch zurückgegriffen, zum anderen besteht die Möglichkeit, einen neuen Namen durch Buchstabieren einzuführen.

Die Fähigkeit, einen neuen Namen zu erlernen, besitzt auch das System von Chung et al., das in Abschnitt 3.2 vorgestellt wurde. Allerdings wird zur Wiedererkennung eines gelernten Namens nicht zwingend seine Schreibweise benötigt, wie sie dort entweder durch Buchstabieren oder durch Eintragen über eine Telefontastatur erfasst wird. Es zeigt sich, den Experimenten aus Abschnitt 7 zufolge, dass zur Wiedererkennung auch eine Fusion der Phonemhypothesen ausreicht. Dies kann vor allem dann nützlich sein, wenn selbst durch mehrfaches Buchstabieren der Name nicht richtig erkannt wird und der Dialog erfolglos abgebrochen werden muss. In diesem Fall kann zumindest durch das Fusionsergebnis des mehrfach ausgesprochenen Namens eine Phonemsequenz zusammen mit einer ID in das Wörterbuch aufgenommen werden, mit der das Wiedererkennen des Namens möglich ist. Es bleibt allerdings zu untersuchen, ob diese Phonemsequenz auch zur Ansprache der Person geeignet ist.

Ein Nachteil der Phonemfusion besteht in der steigenden Laufzeit für jede zusätzliche Äußerung, da für jede Äußerung alle Hypothesen bewertet werden müssen. Dieser Anstieg kann allerdings durch Zwischenspeichern rechenintensiver Ergebnisse und Beschränkung der zu untersuchenden Hypothesen in Grenzen gehalten werden.

Ein Vorteil der vorgestellten Fusion ist ihre Unabhängigkeit gegenüber der verwendeten Akustik und des verwendeten Sprachmodells. Das bedeutet erstens, dass keine Änderungen für eine neue Sprache vorgenommen werden müssen und zweitens, dass auch beispielsweise eine Fusion auf Buchstabierungen möglich ist.

In der Praxis bietet sich diese Fusion allerdings nicht für die Buchstabierung an. Es zeigt sich, dass beim Buchstabieren oft Fehler gemacht werden, etwa Buchstabendreher oder abgebrochene Buchstabiersequenzen. Dies führt zu sehr unterschiedlichen Eingaben und die Annahme, dass die richtige Hypothese auf alle tatsächlichen Eingaben am besten

passt, kann nicht mehr getroffen werden. Bei der Phonemerkennung sind die Eingaben, einen kooperativen Benutzer vorausgesetzt, immer ähnlich.

Durch das verwendete zweistufige System bei der Ausspracheerkennung wird die Wahrscheinlichkeit einen Namen zu erkennen anhand der Wahrscheinlichkeit, dass ein Name dem System bekannt ist $p(\text{bekannt})$ und anhand der Größe des verwendeten Vokabulars, maximiert. Dazu wird die zu erwartende Erkennerleistung aus den Messwerten berechnet. Dies ist vor allem dann nützlich, wenn sich das Gesamtsystem anhand anderer Informationen *sicher* ist, dass es die Person kennt. In diesem Fall, *weiß* der Erkenner, dass er nur mit dem eingeschränkten Vokabular arbeiten kann, was zur Folge hat, dass die gemachten Aussagen besser verstanden werden.

Weiterhin bietet das in dieser Arbeit entwickelte System eine Möglichkeit, um mit einem gewünschten Sprachmodell auf einem wählbaren Zeitbereich einer Äußerung ...

- gezielt eine Hypothese zu bewerten.
- Hypothesen in Form einer N-Bestenliste zu generieren.

Diese zusätzliche Funktionalität, in Verbindung mit dem entwickelten Buchstabier- sowie Phonemerkenner, bildet eine wichtige Grundlage, um auf höherer Ebene die gemachten Hypothesen gezielt zu fusionieren.

9. Zusammenfassung und Ausblick

In diesem Kapitel soll eine kurze Zusammenfassung über diese Arbeit gegeben werden. Anschließend werden einige Punkte zur Erweiterung und Verbesserung des Systems vorgeschlagen.

9.1. Zusammenfassung

In dieser Arbeit wurde ein System entwickelt, das in der Lage ist neue Namen zu erlernen um diese bei einem späteren Lauf wieder zu erkennen. Dazu wurden folgende Teilkomponenten entwickelt:

- OOV Erkennung für ein deutschsprachiges System
- Buchstabiererkenner (verwendet für das englischsprachige und das deutschsprachige System)
- Phonemerkenner für das deutschsprachige System
- Phonemerkenner für das englischsprachige System
- eine allgemeine Möglichkeit Hypothesen mehrerer gleichartiger Äußerungen zu fusionieren.

Weiterhin wurde ein Experiment durchgeführt, das eine zu erwartende Erkennerleistung in Abhängigkeit der Vokabulargröße abschätzen lässt. Somit konnte ein System entworfen werden, das anhand einer Datenbankgröße und einer Wahrscheinlichkeit, dass eine Person dem System bekannt ist, eine Entscheidung über die Art der Dekodierung trifft, um so die zu erwartende Erkennerleistung zu maximieren. Erste Experimente bestätigten, diese Strategie, bei der anhand eines Grenzwertes ermittelt wird, ob zuerst auf einer Top-1047-Liste oder auf der Datenbank des Systems dekodiert wird.

In einem weiteren Experiment wurde untersucht, ob die Phonemhypothese, die durch die Fusion ermittelt wurde, zur Wiedererkennung eines gelernten Namens verwendet werden kann. Es zeigt sich, dass sich diese Hypothese auf den gegebene Daten und einer Datenbank mit bis zu 100 Einträgen, sogar besser zur Wiedererkennung eignet als die Phonemsequenz, die der Graphem-zu-Phonem-Konverter erzeugt.

9.2. Ausblick

Das entwickelte System lässt sich verallgemeinern und erweitern um damit beliebige Wörter oder Wortfolgen erlernen zu können. Das vorgestellte Vorgehen zur Erkennung unbekannter Vornamen lässt sich auch auf andere Bereiche wie Produktnamen, Nachnamen oder Ortsnamen übertragen. Um allerdings in einem Dialog Wörter von verschiedenen Bereichen, wie Vornamen, Produktnamen usw. zu erlernen, ist es sinnvoll, die Fusion mehrerer Hypothesen auf einer höheren Ebene durchzuführen. Damit muss sich der Spracherkenner keine Zustände oder Fusionsergebnisse zwischenspeichern, sondern nur die Funktionalität zur Erkennung von Sprache, dem gezielten Bewerten von Hypothesen auf einer gegeben Äußerung und der Möglichkeit neu gelernte Wörter in sein Suchvokabular aufzunehmen bereitstellen.

Weiterhin bietet es sich an, zwei Erkenner zu verwenden. Einen der die Eingaben des Benutzers möglichst in Echtzeit dekodiert und seine Hypothesen an eine höhere Komponente weiterreicht und einen weiteren, der asynchron gezielte Anfragen beantworten kann. Dazu gehört sowohl die Dekodierung als auch die Bewertung einer einzelnen Hypothese auf einem beliebigen Zeitabschnitt der Eingabe mit einem gewählten Erkennermodell. Damit ist es dann einer höheren Komponente, wie beispielsweise Tapas, möglich, nochmal *gezielt hinzuhören*. Also ein Stück einer Äußerung mit einer, an den Kontext angepassten, Konfiguration erneut zu dekodieren. Weiterhin kann auf diese Weise die in dieser Arbeit beschriebene Fusion auf höherer Ebene mit anderen Fusionen vereinigt werden. Da die Bewertung der Einzelhypothesen durch einen einfachen Aufruf des zweiten Erkenners möglich ist.

Informationen die auf höherer Ebene zur Verfügung stehen und somit für eine Fusion auf höherer Ebene sprechen, könnten im Fall von Vornamen, das Alter und das Geschlecht einer Person sein, sowie weitere Wissensquellen wie Statistiken über Namensverteilungen in Abhängigkeit des Alters. Weiterhin kann ähnlich wie in dem System von Gruenstein,

erwähnt in Abschnitt 3.3, eine Suche im Internet angestoßen werden um anhand von Informationen, wie dem Aufenthaltsort des Systems, gezielt nach damit verbundenen Namen zu suchen.

Um beliebige Wortfolgen, z. B. für Buchtitel, erlernen zu können, bietet sich ein ähnliches Vorgehen an, wie bei der Buchstabierung. Dazu wird der Spracherkenner um ein N-Gramm Sprachmodell erweitert, das statt Buchstaben Wörter enthält. So kann ein beliebiger Titel gesprochen werden. Auch die vorgestellte Fusion kann mit diesem Sprachmodell arbeiten, es ist allerdings zu evaluieren wie gut diese mit solchen Äußerungen arbeitet. Wird dieses N-Gramm Sprachmodell noch um eine OOV-Erkennung erweiter, so kann auch ein unbekanntes Wort in einem Buchtitel wie in dieser Arbeit beschrieben erkannt und erlernt werden.

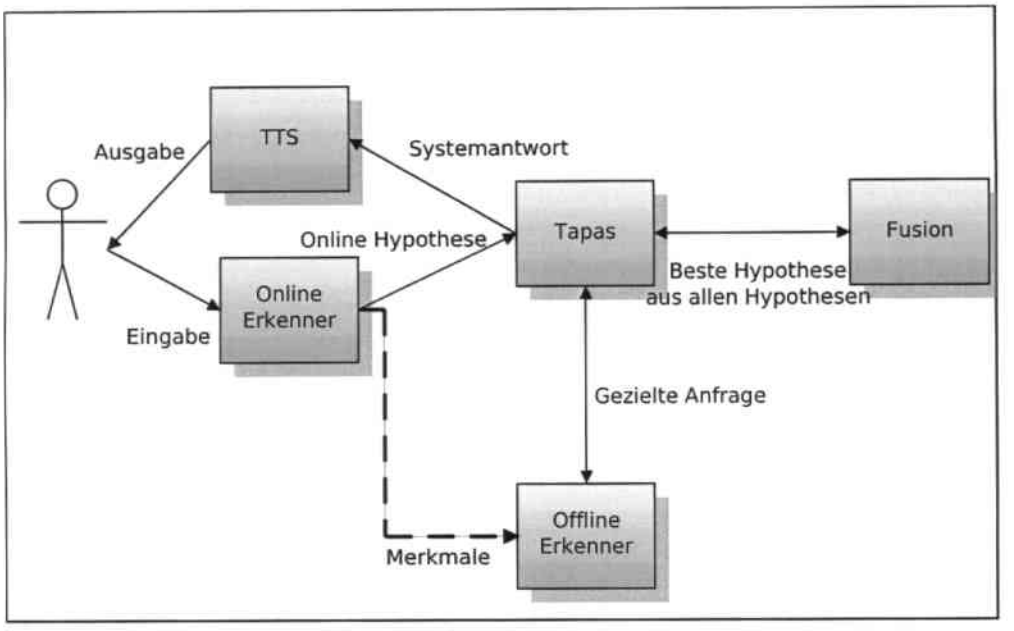

Abbildung 9.1.: Entwurf für ein erweitertes System zum erlernen unbekannter Wörter und Wortfolgen

Abbildung 9.1 zeigt einen Entwurf für ein solches erweitertes System. Dabei kann der Offline-Erkenner asynchron arbeiten und somit zeitaufwändigere Arbeiten ausführen. Ist das Ergebnis des Offline-Erkenners berechnet, kann es mit anderen Hypothesen zu einer neuen Hypothese verknüpft werden. Eine solche Verknüpfung kann ein Verfahren wie es Meyer und Hild vorstellen, beschrieben in Abschnitt 3.1, oder die in Abschnitt 5.4 besprochene Fusion sein. Weiterhin können dort vom Benutzer abgelehnte

Hypothesen gestrichen werden, damit diese nicht ein zweites mal vorgeschlagen werden.

A. Evaluationsergebnisse: Erkennerverhalten bei wachsendem Vokabular

In diesem Abschnitt befinden sich die in Abschnitt 5.3.3.2 erwähnten Graphen.

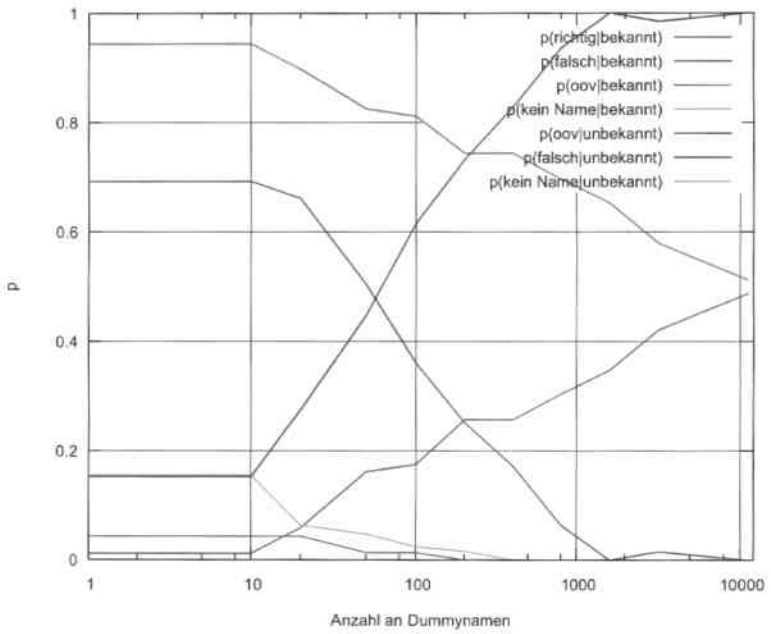

Abbildung A.1.: Schätzungen für 33% bekannte Namen in den Testdaten

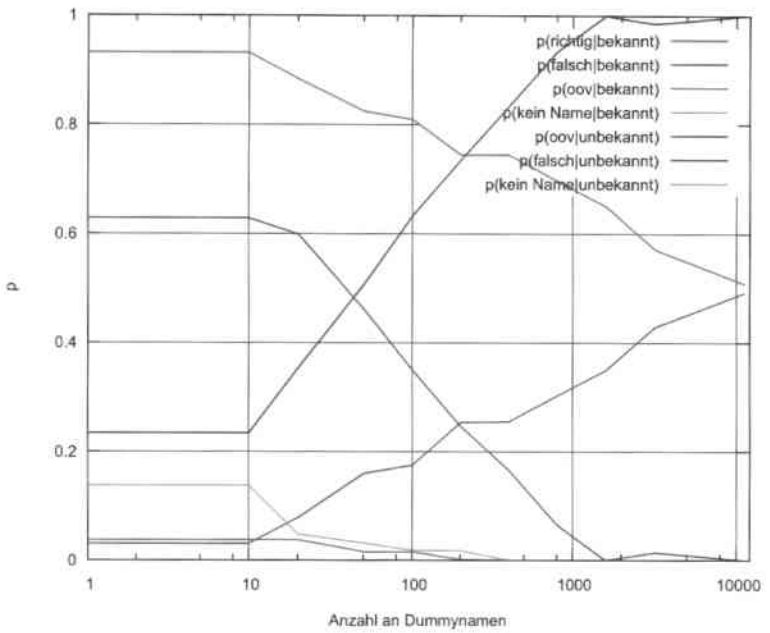

Abbildung A.2.: Schätzungen für 66% bekannte Namen in den Testdaten

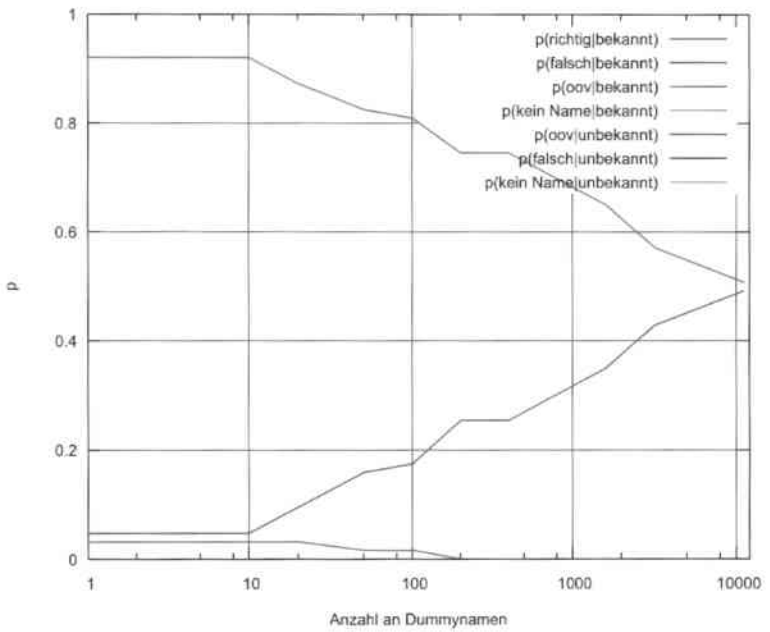

Abbildung A.3.: Schätzungen für 100% bekannte Namen in den Testdaten

Literaturverzeichnis

[Aub01] AUBERLE, ANETTE [BEARB.]: *Duden, Deutsches Universalwörterbuch.* Dudenverl., 4., neu bearb. und erw. Aufl. / [Bearb. der 4. Aufl.: Anette Auberle] Auflage, 2001.

[Bay] *An Essay towards solving a Problem in the Doctrine of Chances.*

[BS06] BEYERER, J. und J. SANDER: *Fusion agents - realizing Bayesian fusion via a local approach.* In: *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*, Seiten 249–254, Sept. 2006.

[Bus02] BUSSMANN, HADUMOD (Herausgeber): *Lexikon der Sprachwissenschaft.* Kröner, 3., aktualis. u. erw. Aufl. Auflage, 2002.

[CSW03] CHUNG, GRACE, STEPHANIE SENEFF und CHAO WANG: *Automatic acquisition of names using speak and spell mode in spoken dialogue systems.* In: *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Seiten 32–39, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[CT94] CAVNAR, WILLIAM B. und JOHN M. TRENKLE: *N-Gram-Based Text Categorization.* In: *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Seiten 161–175, Las Vegas, US, 1994.

[EASD00] EHRENMANN, M., D. AMBELA, P. STEINHAUS und R. DILLMANN: *A Comparison of Four Fast Vision Based Object Recognition Methods for Programming by Demonstration Applications.* In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Seiten 1862–1867, San Francisco, 2000.

[ES05] EKENEL, H. K. und R. STIEFELHAGEN: *Local Appearance based Face Recognition Using Discrete Cosine Transform.* In: *13th European Signal Processing Conference (EUSIPCO)*, Antalya, Turkey, 9 2005.

[FGH+97] FINKE, M., P. GEUTNER, H. HILD, T. KEMP, K. RIES und M. WESTPHAL: *The Karlsruhe-Verbmobil Speech Recognition Engine.* In: *Proc. ICASSP '97*, Seiten 83–86, Munich, Germany, 1997.

[FHW04] FUEGEN, CHRISTIAN, HARTWIG HOLZAPFEL und ALEX WAIBEL: *Tight Coupling of Speech Recognition and Dialog Management - Dialog-Context Grammar Weighting for Speech Recognition.* In: *Proceedings of the International Conference on Spoken Language Processing, ICSLP 2004*, 2004.

[Gav00] GAVALDÀ, MARSAL: *Soup: a parser for real-world spontaneous speech.* In: *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT-2000),*, Trento, Italy, 2 2000.

[GSW06] GRUENSTEIN, ALEXANDER, STEPHANIE SENEFF und CHAO WANG: *Scalable and PortableWeb-Based Multimodal Dialogue Interaction with Geographical Databases.* In: *Proceedings of the Ninth International Conference on Spoken Language Processing (Interspeech 2006 - ICSLP)*, Seiten 453–456, 2006.

[Hil97] HILD, HERMANN: *Buchstabiererkennung mit neuronalen Netzen in Auskunftssystemen.* Shaker, 1997.

[Hol05] HOLZAPFEL, HARTWIG: *Towards Development of Multilingual Spoken Dialogue Systems.* In: *Proceedings of the 2nd Language and Technology Conference*, 2005.

[HSE+07] HOLZAPFEL, HARTWIG, THOMAS SCHAAF, HAZIM KEMAL EKENEL, CHRISTOPH SCHAA und ALEX WAIBEL: *A Robot learns to know people - First Contacts of a Robot.* In: FREKSA, C., M. KOHLHASE und K. SCHILL (Herausgeber): *KI 2006: Advances in Artificial Intelligence, Lecture Notes in Computer Science*, Band 4314, 2007.

[HW93] HILD, HERMANN und ALEX WAIBEL: *Speaker-Independent Connected Letter Recognition With a Multi-State Time Delay Neural Network.* In: *3rd European Conference on Speech, Communication and Technology (EUROSPEECH) 93*, Band 2, Seiten 1481–1484, 1993.

[JMBB77] JELINEK, F., R. L. MERCER, L. R. BAHL und J. K. BAKER: *Perplexity–a measure of the difficulty of speech recognition tasks.* The Journal of the Acoustical Society of America, 62:63, 12 1977.

[Kai06] KAISER, EDWARD C.: *Using redundant speech and handwriting for learning new vocabulary and understanding abbreviations.* In: *ICMI '06: Proceedings of the 8th international conference on Multimodal interfaces*, Seiten 347–356, New York, NY, USA, 2006. ACM Press.

[Kel83] KELLEY, J. F.: *An empirical methodology for writing user-friendly natural language computer applications.* In: *CHI '83: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, Seiten 193–196, New York, NY, USA, 1983. ACM Press.

[KHDM98] KITTLER, J., M. HATEF, R.P.W. DUIN und J. MATAS: *On combining classifiers.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(3):226–239, March 1998.

[KHEW07] KÖNN, STEPHAN, HARTWIG HOLZAPFEL, HAZIM KEMAL EKENEL und ALEX WAIBEL: *Integrating Face-ID into an Interactive Person-ID Learning System.* 3 2007.

[MH97] MEYER, MICHAEL und HERMANN HILD: *Recognition of Spoken and Spelled Proper Names.* In: *Proc. Eurospeech '97*, Seiten 1579–1582, Rhodes, Greece, 9 1997.

[MMD+04] MCCOWAN, I., 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-RR 73, IDIAP, Martigny, Switzerland, 2004.

[MS05] MEINTRUP, DAVID und STEFAN SCHÄFFLER: *Stochastik.* Springer, 2005.

[PG06] PARK, A. und J.R. GLASS: *Unsupervised Word Acquisition from Speech using Pattern Discovery.* In: *ICASSP 2006 Proceedings*, Band 1, Seiten 409–412, 32 Vassar Street, Cambridge, MA 02139, USA., 2006. MIT Computer Science and Artificial Intelligence Laboratory.

[Rog05] ROGINA, IVICA: *Sprachliche Mensch-Maschine-Kommunikation.* 2 2005.

[RT99] RAINES, PAUL und JEFF TRANTER: *TCL/TK in a nutshell.* O'Reilly, 1. ed. Auflage, 1999.

[Sch01] SCHÖNING, UWE: *Theoretische Informatik - kurzgefasst.* Spektrum, Akad. Verl., 4. Aufl. Auflage, 2001.

[Sch04] SCHAAF, THOMAS: *Erkennen und Lernen neuer Wörter.* Doktorarbeit, Universität Karlsruhe, 2004.

[Sch05] SCHAA, CHRISTOPH: *Proaktive Initiierung von Dialogen für humanoide Roboter.* Diplomarbeit, Universität Karlsruhe (TH), 2005.

[Sha48] SHANNON, C. E.: *A Mathematical Theory of Communication.* Reprinted with corrections from The Bell System Technical Journal, 27:379–423, July 1948.

[SLM96] SENEFF, S., R. LAU und H. MENG: *ANGIE: A New Framework for Speech Analysis Based on Morpho-phonological Modelling.* In: *Proc. ICSLP '96*, Band 1, Seiten 110–113, Philadelphia, PA, 1996.

[SMFW01] SOLTAU, H., F. METZE, C. FÜGEN und A. WAIBEL: *A one-pass decoder based on polymorphic linguistic context assignment.* In: *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on*, Seiten 214–217, 9-13 Dec. 2001.

[ST95] SCHUKAT-TALAMAZZINI, ERNST GÜNTER: *Automatische Spracherkennung.* Vieweg, 1995.

[Sto02] STOLCKE, ANDREAS: *SRILM - An Extensible Language Modeling Toolkit.* In: *Intl. Conf. Spoken Language Processing*, September 2002.

[WF74] WAGNER, ROBERT A. und MICHAEL J. FISCHER: *The String-to-String Correction Problem.* J. ACM, 21(1):168–173, 1974.