Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Louisa Spahl

Kommunikationsunterstützende mobile Anwendung mit integrierter Satzverbesserung sowie Sprachausgabe für Kinder

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Louisa Spahl

Kommunikationsunterstützende mobile Anwendung mit integrierter Satzverbesserung sowie Sprachausgabe für Kinder

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Ulrike Steffens
Zweitgutachter: Prof. Dr. rer. nat. Chris Biemann

Eingereicht am: 23.08.2017

Louisa Spahl

Thema der Arbeit
Kommunikationsunterstützende mobile Anwendung mit integrierter Satzverbesserung sowie Sprachausgabe für Kinder

Stichworte
N-Gramm, Satzverbesserung, Satzergänzung, Counts, Language Model, Grammatik, Soothing, Talker, Corpus, Maximum Likelihood Estimation, MLE

Kurzzusammenfassung
Elektronische Kommunikationshilfen spielen in der Welt der sprach- und sprecheingeschränkten Kinder eine immer größere Rolle. Die Kinder erfahren hierbei eine Akzeptanz gegenüber der Gesellschaft. Doch die Eingabe eines Satzes in eine elektronische Kommunikationshilfe braucht Zeit und verlangsamt die Kommunikation. Hier setzt diese Arbeit mit einer Satzverbesserung bzw. -vervollständigung an. Die Satzverbesserung wird in das *Verben konjugieren*, *Nomen deklinieren* und in das *Hinzufügen von Wörtern* unterteilt. Für die Verbesserung werden Bi-Gramme und Tri-Gramme verwendet, die einen Kontext von einem bzw. zwei vorherigen Wörtern betrachten. Anhand von Beispielsätzen von Probanden werden beide Verfahren verglichen und ausgewertet. Das Ergebnis zeigt die Schwierigkeit, Wörter an der richtigen Stelle hinzuzufügen, ohne den Sinn des Satzes zu verändern.

Louisa Spahl

Title of the paper
A mobile application to support communication for speech impaired children with integrated sentence correction and voice output

Keywords
N-Gram, Sentence correction, Counts, Language Model, Grammar, Smoothing, Talker, Corpus, Maximum Likelihood Estimation, MLE

Abstract
Electronic communication applications play a major role in speech impaired children's lives. Through them, hindered children experience a bigger acceptance in society. However the electronic sentence input on such devices takes time and slows down communication. Based on this aspect this paper tries to find a way to correct entered sentences on the device itself. The developed application used in this study *conjugates verbs*, *declines nouns* and *adds missing words* to a sentence. Bi-Gram and Tri-Gram are used to simplify its process and are tested based on probands sample sentences supplied. The result manifests the complexity to add words in a correct sentence structure without changing the meaning of the intended phrase.

Danksagung

An dieser Stelle möchte ich allen Menschen danken, die mich während meiner Bachelorarbeit fachlich und persönlich unterstützt haben.

Ein ganz besonderer Dank geht an meinen Freund Christian, der mir mit vielen nützlichen Tipps helfen konnte. Zudem möchte ich mich dafür bedanken, dass er mich sowohl während des gesamten Studiums als auch während der Bachelorarbeit aufgebaut hat, vor allem in problematischen Phasen.

Ebenso möchte ich mich bei meinen Eltern für die Unterstützung meiner Entscheidungen und der finanziellen Hilfe während meines Studiums bedanken. Zusätzlich möchte ich meiner Mutter und meiner Cousine Maike für das Korrekturlesen einen herzlich Dank aussprechen.

Außerdem möchte ich mich bei allen Probanden, Katja, Michael, Sophie, Claire, Céline und Ole bedanken. Sie haben mich beim Testen der Anwendung unterstützt.

Weiterhin danke ich Frau Kitzinger, die mir die METACOM Symbole zur Verfügung gestellt hat, denn ohne ihre Symbole hätte meine Applikation nicht einmal halb so gut ausgesehen.

Für die Idee dieser Arbeit möchte ich mich besonders bei Kjell bedanken, den ich jahrelang als Babysitterin betreut habe. Er hat mich auf das Thema dieser Bachelorarbeit gebracht. Ohne ihn hätte ich nicht verstanden, wie frustrierend es ist, alles zu verstehen, aber nicht mit Worten antworten zu können.

Während des Schreibens dieser Arbeit habe ich viel Musik von Ludovico Einaudi in Dauerschleife gehört. Ein Dank gilt daher dieser tollen Musik.

Inhaltsverzeichnis

1. Einführung **1**
- 1.1. Motivation . . . 1
- 1.2. Problemstellung und Zielsetzung . . . 2
- 1.3. Aufbau der Arbeit . . . 2

2. Grundlagen **4**
- 2.1. Sprach- und Sprechstörungen im Kindesalter . . . 4
- 2.2. Sprache und Grammatik . . . 5
  - 2.2.1. Natürliche-, formale Sprache und Sprachsynthese . . . 5
  - 2.2.2. Grammatik . . . 6
  - 2.2.3. Chomsky-Hierarchie . . . 7
- 2.3. N-Gramm . . . 8
  - 2.3.1. Probleme bei N-Grammen . . . 9
  - 2.3.2. Smoothing . . . 10
- 2.4. Vergleichbare Arbeiten . . . 13

3. Realisierung von *Talk To Me* **17**
- 3.1. Anwendungsfall . . . 17
- 3.2. Anforderungen an das System . . . 18
  - 3.2.1. Funktionale Anforderungen: . . . 18
  - 3.2.2. Nichtfunktionale Anforderungen: . . . 19
- 3.3. Komponenten . . . 19
  - 3.3.1. Frontend Android Anwendung . . . 20
  - 3.3.2. Backend Python Anwendung . . . 23
  - 3.3.3. Backend: Trainingsdaten . . . 24
  - 3.3.4. Backend: Satzverbesserung mit Bi-Grammen . . . 26
  - 3.3.5. Backend: Satzverbesserung mit Tri-Grammen . . . 32
  - 3.3.6. IBM Watson Text to Speech . . . 33
  - 3.3.7. Zusammenspiel der Komponenten . . . 33

4. Praxistest **35**
- 4.1. Ausführung . . . 35
- 4.2. Auswertung . . . 36
  - 4.2.1. Vergleich: Satzverbesserung mit Bi- und Tri-Grammen . . . 36
    - 4.2.1.1. Überblick der Satzverbesserung . . . 36
    - 4.2.1.2. Verben konjugieren . . . 38

4.2.1.3. Nomen deklinieren 40
4.2.1.4. Zusammenfassung 41
4.2.2. Benutzbarkeit Benutzeroberfläche 42
4.2.3. Allgemeine Verbesserungsmöglichkeiten 43
4.3. Anforderungen evaluieren 45

5. Fazit und Ausblick **47**

A. Anhang **48**
A.1. Tagset der Part of Speech (POS) 48
A.2. Testsätze 50
A.3. Screenshots der Andorid Applikation 56
A.4. Klassifizierung der ICD-10 60

Abkürzungsverzeichnis **63**

Glossar **64**

Tabellenverzeichnis

2.1. *Counts* von Uni-Gramm und Bi-Grammen 11
2.2. Beispiel der Berechnung von Laplace 11

3.2. Spezifikation Request zur Satzverbesserung 23
3.3. Spezifikation Request für die Audiodatei 23
3.4. Iterationsschritte des Bi-Gramm Verfahren anhand des Beispielsatzes *I like my brother* 32

4.1. Wahrscheinlichkeiten der richtig korrigierten Sätze in allen Testsätzen mit unterschiedlichem Verfahren 36
4.2. Statistik der korrigierten Verben aus allen Testsätzen 38
4.3. Anzahl der korrigierten Nomen, die auf einer Grundzahl oder einen Demonstrativbegleiter folgen, aus allen Testsätzen 40
4.4. Prozentuale Ansicht der richtigen Ergebnisse der verschiedenen Vorgehen . . 41

Abbildungsverzeichnis

2.1. Darstellung der Chomsky-Hierarchie . . . 8
2.2. N-Gramm Frequency gerankt von Cavnar und Trenkle (1994) auf Basis eines technischen Dokuments. Die Y-Achse zeigt die Counts eines N-Grammes an und die X-Achse zeigt wie viele N-Gramme die gleiche Anzahl an Counts haben 10

3.1. Technisches Komponentendiagramm . . . 20
3.2. Beispiel für den Satz *I am two years old* . . . 21
3.3. Startbildschirm der Applikation *Talk To Me* mit eingegebenem Satz „I like you“. Weitere Screenshots befinden sich in Abschnitt A.3 . . . 22
3.4. Sequenzdiagramm: Zusammenspiel der internen Komponenten bei Eingabe eines Satzes . . . 34

A.1. Startbildschirm . . . 56
A.2. Kategorie Zahlen und Buchstaben . . . 56
A.3. Kategorie Arzt . . . 56
A.4. Kategorie Berufe . . . 56
A.5. Kategorie Elektronik . . . 56
A.6. Kategorie Fantasie . . . 56
A.7. Kategorie Farben und Formen . . . 57
A.8. Kategorie Gefühle . . . 57
A.9. Kategorie Haus . . . 57
A.10. Kategorie Kleidung . . . 57
A.11. Kategorie Kleine Wörter . . . 57
A.12. Kategorie Sport . . . 57
A.13. Kategorie Instrumente . . . 57
A.14. Kategorie Hygiene . . . 57
A.15. Kategorie Körperteile . . . 58
A.16. Kategorie Personen . . . 58
A.17. Kategorie Essen . . . 58
A.18. Kategorie Obst und Gemüse . . . 58
A.19. Kategorie Spielzeug . . . 58
A.20. Kategorie Süßigkeiten . . . 58
A.21. Kategorie Tiere . . . 58
A.22. Kategorie Verben . . . 58
A.23. Kategorie Natur . . . 59
A.24. Kategorie Saison . . . 59

1. Einführung

Sprach- bzw. Sprechstörungen können im Kindesalter auftreten (Nonnenmacher, 2016). Besonders in dieser Zeit ist die Entwicklung der Sprache wichtig, weil Kinder über die Sprache lernen Beziehungen zu Bezugspersonen aufzubauen und Gefühle auszudrücken. Durch die gezielte Sprach- und Sprechtherapie eines Logopäden bzw. einer Logopädin können Sprach- bzw. Sprechstörungen minimiert werden. Bei einigen Krankheiten wie Aphasie oder Alalie kann eine Sprache nie erlernt werden. Während dieser Zeit hat das Kind weiterhin Probleme sich mit seiner Sprache zu äußern. Häufig führt es dazu, dass ein Kind Emotionen durch Gewalt in Form von schlagen, kratzen und spucken ausübt, um sich äußern zu können und um Gefühle zu zeigen. Ein Sprachcomputer, z.B. als mobile Applikation, kann bei der Kommunikation über Sprache behilflich sein. Das Kind soll Wörter in die Applikation eingeben können und diese aussprechen lassen. Obwohl das Vokabular eines Kindes noch nicht so umfangreich ist, wie das eines Erwachsenen, kann die Eingabe von Wörtern lange dauern; besonders, wenn der Satz viele kleine Wörter beinhaltet. Die Applikation soll hier unterstützend wirken und kleinere, fehlende Wörter generieren und angegebene Wörter grammatisch korrekt ausgeben.

1.1. Motivation

Aus eigenen Erfahrungen ist es schwieriger die Bedürfnisse des Kindes zu erfahren, wenn es nicht sprechen kann, als bei Kindern, die sprechen können. Insbesondere in dem Alter, in dem es eigentlich sprechen sollte, da die Bedürfnisse eines Kindes mit zunehmenden Alter ansteigen. Es braucht längere *Kommunikation* zwischen Erwachsenem und Kind, bis verstanden wird, dass das Kind einen Apfel essen möchte oder dass das Lieblingskuscheltier im Kindergarten der Eisbär ist. Kinder werden von elektronischen Geräten angezogen und können nur schwer bis unmöglich davon fern gehalten werden (Blank-Mathieu, 2001, Abschnitt „Medien werden von allen Kindern im Vorschulalter benützt“). Diese Faszination kann für das Sprechen ausgenutzt werden. Zudem sind mobile Geräte heutzutage in 75,4% (2016) der Haushalte vorhanden (Statistisches Bundesamt [Destatis], 2017). Aus diesen Gründen wird eine mobile Applikation entwickelt, die das Sprechen des Kindes unterstützt bzw. übernimmt.

1.2. Problemstellung und Zielsetzung

Diese Arbeit soll die Entwicklung eines prototypischen Sprachcomputers beschreiben. Schwerpunkt soll die Benutzbarkeit der Anwendung von Kindern im Kindergarten- bzw. Vorschulalter sein. Die Bedienung soll möglichst einfach gehalten werden und die Grammatik so weit es geht vom Computer übernommen werden. Ziel ist es, das Kind in der Alltagssprache zu unterstützen, um so Motivation für eine Konversation aufzubauen.

Kinder im Kindergarten- bzw. Vorschulalter können in der Regel nicht mehr als ihren eigenen Namen und den von Freunden und Familienmitgliedern lesen und schreiben. So lautet es in der Hamburger Bildungsempfehlung, dass Kinder im Kindergarten „Den eigenen Namen schreiben, Namen der anderen Kinder richtig aussprechen und ‚lesen'" (Hautumm u. a., 2012, S. 74) können sollten. Eine Herausforderung ist, die Anwendung so zu gestalten, dass weder Lesen noch Schreiben Voraussetzungen sind. Eine mögliche Lösung, die in dieser Arbeit behandelt wird, ist das Darstellen von Wörtern anhand von Bildern bzw. Symbolen. Diese Symbole müssten allerdings auch, wie alle anderen Symbole in der Umgebung, erlernt werden (Senatsverwaltung für Bildung, Jugend und Wissenschaft Berlin, 2014, S. 103). Das Grundwissen, dass Wörter Wortarten wie Nomen und Verben besitzen, ist Kindergartenkinder meistens noch nicht bekannt, da diese Lehrinhalte der Schule sind (Landesinstitut für Lehrerbildung und Schulentwicklung, 2011). Eine sinnvolle Anordnung der Bilder bzw. Symbole ist notwendig, damit Kinder bei der Geschwindigkeit des Sprechens über Symbole nicht oder nur gering eingeschränkt werden.

Neben den Zielen der guten Bedienbarkeit und der richtigen Anordnung und Gruppierung von Wörtern, ist die Satzkorrektur der eingegebenen Sätze mit Hilfe eines Language Model (LM) ein weiteres Ziel. Dabei schließt die Korrektur das *Deklinieren von Nomen*, *Konjugieren von Verben* und das *Einfügen von kleineren Worten* (z.B. Artikel oder Präpositionen) ein.

Die Anwendung hat den Namen *Talk To Me* bekommen. Der Name soll eine Anspielung auf das häufige Sprechen in der dritten Person gegenüber Kindern, die nicht sprechen können, sein. So soll er eine Aufforderung sein, mit Kindern direkt zu sprechen.

1.3. Aufbau der Arbeit

Zunächst wird in Kapitel 2 Grundlagenwissen zu Sprach- bzw. Sprechstörungen im Kindesalter, Grammatiken und Algorithmen zur Satzverbesserung geschaffen. Zudem werden vergleichbare Arbeiten in Hinblick auf die Satzverbesserung und auf andere, sogenannten Talkern, kurz beschrieben. In Kapitel 3 wird die Realisierung des Sprachcomputers erläutert. Dazu werden

die modellierten Komponenten und der verwendete Algorithmus vorgestellt. In Kapitel 4 wird ein Praxistest der Anwendung durchgeführt und die Anforderungen evaluiert. Schließlich wird ein Fazit in Kapitel 5 gezogen und ein Ausblick auf zukünftige, mögliche Weiterentwicklungen gegeben.

2. Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen, die für diese Arbeit relevant sind.

2.1. Sprach- und Sprechstörungen im Kindesalter

Kinder können nicht von Anfang an ab der Geburt, sprechen (Wendlandt und Niebuhr-Siebert, 2000, S. 10). Sie erlernen es im Laufe der Zeit. In der Regel hat ein Kind mit vier bis sechs Jahren seine Muttersprache erlernt, sodass es fließend kommunizieren kann (Wendlandt und Niebuhr-Siebert, 2000, S. 23).

Wendlandt und Niebuhr-Siebert (2000)[S. 42 f.] geben einen Überblick über einige Störungen des Sprechens und der Sprache.

Bei einer Sprachentwicklungsstörung (SES) treten eine Sprachverständnisstörung, ein eingeschränkter Wortschatz, Dysalie und Dysgrammatismus gemeinsam auf. Dysalie ist eine Lautstörung, d.h. „die Aussprache gelingt nicht in angemessener Weise“ (Wendlandt und Niebuhr-Siebert, 2000, S. 41) und/oder das Kind kann die Laute bilden, verwendet diese aber nicht an der richtigen Position in einem Wort. Zum Beispiel sagt das Kind *Lume*, statt *Blume*. Dysgrammatismus ist die Störung beim Erwerb und Gebrauch der Grammatik. Von einer SES spricht man allerdings erst, wenn ein Rückstand von mindestens einem halben Jahr vorhanden ist, gegenüber dem normalen Prozess des Erlernens einer Sprache.

Alalie ist das Ausbleiben einer Sprachentwicklung. Das Kind spricht gar nicht, nur wenige Wörter oder nur dieselben Laute.

Eine weitere Sprachstörung ist der Mutismus. Darunter fallen Kinder, die das Sprechen bereits erlernt haben, aber plötzlich anfangen zu schweigen. Dies kann zum einen ein totales Schweigen sein oder nur in Gegenwart bestimmter Personen (elektiver Mutismus).

Aphasie ist eine erworbene Störung der Sprachverarbeitung. Sie ist meist organisch bedingt und tritt z.B. nach Schlaganfällen, Traumata und entzündlichen Prozessen auf (Blanken u. a., 2015).

Einige dieser Sprachstörungen lassen sich durch Logopädie verbessern oder ganz aufheben. Sprach- bzw. Sprechstörungen sind schwer zu kategorisieren. Dennoch legt ICD-10 bestimmte

Klassen für Sprach- bzw. Sprechstörungen fest. In Anhang A werden einige Sprachstörungen beschrieben. Diese Arbeit bezieht sich hauptsächlich auf Kinder mit Alalie und Aphasie.

2.2. Sprache und Grammatik

Der folgende Abschnitt befasst sich mit den Unterschieden von natürlicher und formaler Sprache sowie mit der Sprachsynthese. Außerdem werden die vier Klassen der Grammatik kurz erklärt.

2.2.1. Natürliche-, formale Sprache und Sprachsynthese

Zwei Personen kommunizieren mit der gleichen natürlichen Sprache wie z.B. Englisch oder Deutsch. Die natürliche Sprache ist häufig mehrdeutig und es gibt keine eindeutig definierte Menge (Hubwieser u. a., 2010, S. 10). Das Alphabet ist zwar begrenzt, aber die Anzahl der möglichen erzeugbaren Wörter nicht.

Formale Sprachen können eine „[...] eindeutig definierte Menge zugelassener Zeichenketten [...]“ (Hubwieser u. a., 2010, S. 10) erzeugen. Formale Sprachen haben eine feste Grammatik. Diese Grammatik kann auch aus Zuständen bestehen, die ein elektronisches Gerät annehmen kann. In der Linguistik werden formale Sprachen verwendet, um einen Teil der natürlichen Sprache zu modellieren (Jurafsky und Martin, 2008, Kapitel 2). In dieser Arbeit wird eine formale Sprache modelliert bzw. generiert, um einen Teil der natürlichen Sprache abbilden zu können.

Sprachsynthese ist die Erzeugung einer (künstlichen) Sprache mit Hilfe eines regelbasierten Text-to-Speech (TTS) Systems. Dieses System konvertiert einen Text, bestehend aus Zeichen, als *Input* zu einem *Output* in Wellenform (Delogu u. a., 1998). Sprachsynthese wird bei Telefonagenten, bei Software unterstützendem Vorlesen von Texten und bei Kinderspielzeug mit Sprachausgabe verwendet. Außerdem wird Sprachsynthese bei Computern eingesetzt, die eingegebene Texte für Menschen mit Sprach- bzw. Sprechstörungen laut aussprechen. Die bekannteste Person, die Sprachsynthese in einem Computer verwendet, ist der Wissenschaftler Steven Hawking. Er verlor wegen seiner ALS Krankheit die Möglichkeit des Sprechens (Jurafsky und Martin, 2008, Kapitel 8). Eine Schwierigkeit der Sprachsynthese liegt in der Ausdruckskraft einiger Gefühle. Freude, Ärger und Ironie lassen sich in den meisten Fällen nicht über die Stimmlage, der durch Sprachsynthese erzeugten Sprache, ausdrücken (Schlosser u. a., 2007; Delogu u. a., 1998). In dieser Arbeit wird die Sprachsynthese, wie in dem Fall von Steven Hawking, verwendet. Eingegebene Texte werden mit Hilfe der Sprachsynthese ausgesprochen.

2.2.2. Grammatik

Eine Grammatik definiert auch eine formale Sprache. Wörter, die nicht durch diese Grammatik erzeugt werden können, gehören nicht zur formalen Sprache.
So kann beispielsweise die kontextfreie Grammatik (KfG)

1. $S \to abScd$
   $S \to abcd$

die Wörter *abcd*, *ababcdcd* und *abababcdcdcd* etc. erzeugen aber nicht *ababcd*. Teilweise gibt es verschiedene Grammatiken, die eine formale Sprache definieren. So können mit

2. $S \to aTd$
   $T \to bSc$
   $T \to bc$

die gleichen Wörter erzeugt werden wie mit Grammatik 1.

S und T sind Nichtterminale und a, b, c, d sind Terminalsymbole. Die Form $S \to abScd$ ist eine Produktion. Mehrere Produktionen ergeben eine Grammatik.

Im Gegensatz zur natürlichen Sprache muss die Grammatik einer formalen Sprache geändert werden, wenn ein bestimmtes Wort zur Sprache gehören soll. Es kann nicht verallgemeinert werden, dass natürliche Sprache mit Hilfe einer kontextfreien Grammatik (KfG) beschrieben werden kann. Shieber (1985) gab den Schweizerdeutschen Beispielsatz

...mer	em Hans	es	huus	hälfed	aastriiche
...we	Hans-DAT	the	house-ACC	helped	paint
...we	helped	Hans	paint	the	house

und beschrieb, dass eine Abbildung des Schweizerdeutschen in eine (formale) KfG unmöglich ist. Im Schweizerdeutsch stehen teilweise Objekt und Verb nicht unmittelbar hintereinander, obwohl sie voneinander abhängig sind. Das Verb benötigt das Objekt, um an den Kasus ((vier) Fälle) angepasst zu werden. Diese Abhängigkeit lässt sich nicht mit einer KfG beschreiben.

2.2.3. Chomsky-Hierarchie

Der Linguist Noam Chomsky definierte 1956 die Chomsky-Hierarchie. Sie beschreibt formale Grammatiken, die formale Sprachen erzeugen, und lässt diese in vier Klassen aufteilen (Hopcroft und Ullman, 2000). Jede Grammatik ist unterschiedlich komplex und unterschiedlich stark im Generieren von Wörtern. Im Folgenden werden die vier Grammatikklassen kurz beschrieben und deren Regeln erklärt.

Aufzählbare Grammatik auch als Typ 0 in der Chomsky-Hierarchie bezeichnet, erzeugt die am wenigsten eingeschränkte Sprache. Die Grammatik definiert sich als $\alpha \rightarrow \beta$, wobei α und β beliebige Zeichenketten sind und $\alpha \neq \varepsilon$.

Kontextsensitive Grammatik wird als Typ 1 bezeichnet. Im Gegensatz zur aufzählbaren Grammatik darf der rechte Teil der Produktion nicht kleiner als der linke Teil sein. Die Form lautet $\alpha A \beta \rightarrow \alpha \gamma \beta$ mit $\beta \neq \varepsilon$.

Kontextfreie Grammatik (KfG) in der Chomsky-Hierarchie als Typ 2 bezeichnet. Sie wird bei „Definition von Programmiersprachen, bei der Formalisierung der Syntaxanalyse, beim Vereinfachen der Übersetzung von Programmiersprachen und in anderen Prozessen [...] “ (Hopcroft und Ullman, 2000, S. 83) eingesetzt. Die Form ist $A \rightarrow \gamma$, wobei A ein Nichtterminal ist und γ eine beliebige Zeichenkette aus Nichtterminalen und Terminalen.
Die KfG ist die am häufigsten verwendete Grammatik, die zum Beschreiben einer natürlichen Sprache (zu eine formale Sprache) verwendet wird (Jurafsky und Martin, 2008, Kapitel 12).

Reguläre Grammatik auch als Typ 3 bezeichnet, erzeugt eine eingeschränktere Sprache (siehe Abbildung 2.1). Wie bei der KfG darf die linke Seite der Regelform nur aus einer Nichtterminale bestehen und die rechte Seite aus einem oder mehreren Terminalen Symbolen und maximal einem Nichtterminalen Symbol. Die Produktionen sind $A \rightarrow wB$ und $A \rightarrow w$, wenn sie rechts-linear und $A \rightarrow Bw$ und $A \rightarrow w$, wenn sie links-linear ist. w darf leer sein.

Abbildung 2.1 zeigt das Verhältnis der Grammatiken zueinander auf.
Grammatiken können schnell sehr komplex werden, wenn es um die Nachbildung von der natürlichen Sprache geht. Im nächsten Abschnitt wird deshalb eine Approximation von Grammatiken, die *N-Gramme* vorgestellt. Sie beinhalten lediglich Terminale im Gegensatz zu Grammatiken. Es gibt keine Produktionen und der Kontext eines Wortes ist als die N-1 vorherigen Wörter bzw. die N-1 nachfolgenden Wörter beschrieben.

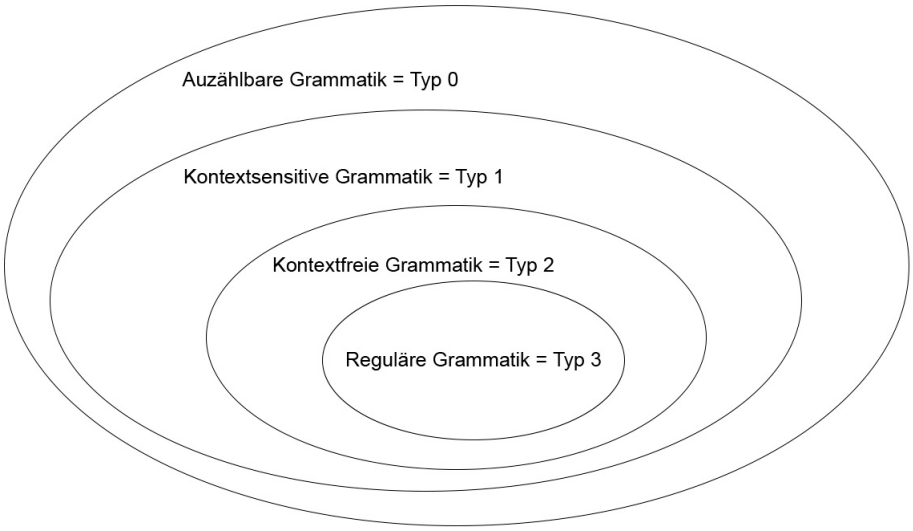

Abbildung 2.1.: Darstellung der Chomsky-Hierarchie

2.3. N-Gramm

N-Gramme sind Sequenzen bzw. Wortfolgen von N Wörtern (Jurafsky und Martin, 2008). Mit Algorithmen wie *Maximum Likelihood Estimation (MLE)* und *Kneser-Ney Smoothing* werden die Wahrscheinlichkeiten einer Wortfolge berechnet. Alle Wortfolgen mit ihren Wahrscheinlichkeiten werden *Language Model (LM)* genannt. Ein Satz wie *April is the fourth month of the year* sieht dann für die N-Gramme wie folgt aus:
Bi-Gramm: (April is), (is the), (the fourth), (fourth month), (month of), (of the), (the year), (year STOP)
Tri-Gramm: (April is the), (is the fourth), (the fourth month), (fourth month of), (month of the), (of the year), (the year STOP), (year STOP STOP)
Als Beispiel wurden Bi-Gramme und Tri-Gramme verwendet. Bei beiden sind am Ende des Satzes STOP-Symbole. Diese symbolisieren das Ende und die Anzahl dieser ist N-1. Mit Hilfe von einem N-Gramm lässt sich die Wahrscheinlichkeit des Folgewortes berechnen. Sei w ein beliebiges Wort aus dem Vokabular V, $w \in V$, so lässt sich die Wahrscheinlichkeit P von w wie folgt berechnen:

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})} \tag{2.1}$$

Hierbei ist C die *Counts* der Wortfolgen und w_{n-N+1}^{n-1} der Kontext in dem das Wort w_n steht. Die Berechnung für die Wahrscheinlichkeit des Wortes *is* in *April is the fourth month of the*

year, mit Hilfe eines Bi-Gramms und folgenden Werten
April is → 8 *Counts* in den Trainingsdaten
April → 15 *Counts* in den Trainingsdaten
ist:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \tag{2.2}$$

$$P(\text{is}|\text{April}) = \frac{C(\text{April is})}{C(\text{April})} \tag{2.3}$$

$$P(\text{is}|\text{April}) = \frac{8}{15} \tag{2.4}$$

$$= 0,533 \tag{2.5}$$

Bei einem Tri-Gramm besteht der Kontext aus den zwei vorherigen Wörtern w_{n-2} und w_{n-1}.

2.3.1. Probleme bei N-Grammen

Cavnar und Trenkle (1994) erklären, dass es in der menschlichen Sprache Wortfolgen gibt, die häufiger vorkommen als andere in derselben Sprache, unabhängig vom Kontext. Werden N-Gramme nach ihrer Häufigkeit von häufig zu selten sortiert und dies über mehrere Corpora, wird ein ähnlicher Graph erkennbar. Ungefähr die obersten 300 Wortfolgen treten unabhängig vom Corpus auf, z.B. würde eine lange englische Passage über das Thema Compiler ähnliche N-Gramme in den obersten 300 Wortfolgen besitzen wie eine lange englische Passage über Gedichte (vgl. Abbildung 2.2).

Die am häufigsten in der Praxis benutzen N-Gramme sind Tri-Gramme (Jurafsky und Martin, 2008, Kapitel 4). Sie betrachten einen größeren Kontext als bei Bi-Grammen, haben aber nicht so viele unbekannte Wortfolgen wie bei einem 4-Gramm. Bei der Gleichung 2.1 treten Probleme auf, wenn bestimmte Wörter oder Wortfolgen nicht im Vokabular, aber in der Eingabesequenz vorkommen. Gibt es das Bi-Gramm bzw. Tri-Gramm im Zähler der Gleichung 2.1 nicht, ist das Ergebnis immer Null. Kritischer wird es, wenn es das Bi-Gramm bzw. Uni-Gramm [1] im Nenner der Gleichung 2.1 nicht im Vokabular gibt. Der Nenner nimmt die Zahl Null an und das Ergebnis der Gleichung wäre dann ein *division by zero*. Eine Lösung für das Problem ist *Smoothing*. Mit Hilfe unterschiedlicher Methoden wird verhindert, dass ein nicht vorhandenes N-Gramm im Vokabular eine Wahrscheinlichkeit von Null hat. Drei Methoden werden vorgestellt.

[1] Uni-Gramm ist ein N-Gramm bestehend aus nur einem Wort

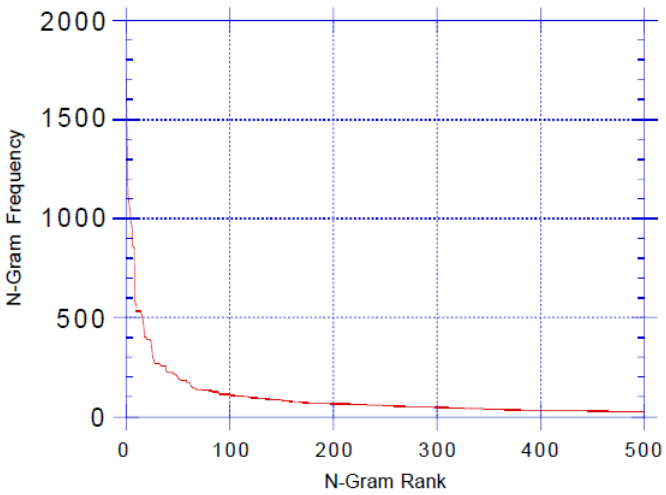

Abbildung 2.2.: N-Gramm Frequency gerankt von Cavnar und Trenkle (1994) auf Basis eines technischen Dokuments. Die Y-Achse zeigt die Counts eines N-Grammes an und die X-Achse zeigt wie viele N-Gramme die gleiche Anzahl an Counts haben

In Kapitel 3 Unterabschnitt 3.3.3 wird auf das gewählte Corpus für diese Arbeit kurz eingegangen. Das Corpus wurde aufgrund der Größe gewählt, der Bezug zur Kindersprache ist nicht direkt gegeben. In der Regel sprechen Kinder in kurzen und einfachen Sätzen und in der Ich-Perspektive. Das Corpus enthält hauptsächlich nur Sätze in der Ich-Perspektive in Zitaten. Dementsprechend kann es sein, dass viele Wahrscheinlichkeiten Null sind. Das im nächsten Abschnitt unter anderem erklärte *add-k-Smoothing* ist das gewählte Vorgehen bei *Talk To Me* die Null Wahrscheinlichkeiten zu umgehen.

2.3.2. Smoothing

Es werden drei Smoothing Algorithmen vorgestellt. Die Verwendung in der Praxis kommt auf den Anwendungsfall an. Der beste und häufigste ist allerdings *Kneser-Ney Smoothing*, erklärt in Abschnitt 2.3.2 (Jurafsky und Martin, 2008, Kapitel 4). Allgemein ist Smoothing eine Variante, wie mit Null Wahrscheinlichkeiten umgegangen werden kann.

Laplace Smoothing ist auch bekannt unter *add-one smoothing*, z. Dt. *füge einen hinzu* (Jurafsky und Martin, 2008, Kapitel 4). Zu der Anzahl der Vorkommen (in der Literatur als *Counts* bezeichnet) eines z.B. Bi-Gramms, wird einer hinzugefügt. So besitzen noch nie vorgekommenen Bi-Gramme einen *Count*. Es wird insgesamt mindestens die Anzahl hinzugefügt, die äquivalent zu den Wörtern in Vokabular V ist. Damit eine (Neu-)verteilung der Wahrscheinlich-

keiten gegeben ist, muss durch die Anzahl aller Wörter im Vokabular geteilt werden, ansonsten würden alle Wahrscheinlichkeiten steigen. Die Gleichung 2.2 wird also verändert zu

$$P^*_{Laplace}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V} \tag{2.6}$$

Beispiel aus Jurafsky und Martin (2008): Sei C_1 die *Counts* des Uni-Gramm (*hier: I*), V die Gesamtanzahl der Worte im Vokabular und C die Anzahl eines Bi-Gramms.

Bi-Gramm	C_1	V	C
I I	2533	1446	5
I want	2533	1446	827
I food	2533	1446	0

Tabelle 2.1.: *Counts* von Uni-Gramm und Bi-Grammen

Die Wahrscheinlichkeiten mit Hilfe der Gleichung 2.6 und den Werten aus Tabelle 2.1 ergibt Tabelle 2.2.

Bi-Gramm	C_1	V	C	Laplace	MLE
I I	2533	1446	5	$\frac{5+1}{2533+1446} = 0,00151$	$\frac{5}{2533} = 0,00197$
I want	2533	1446	827	$\frac{827+1}{2533+1446} = 0,20809$	$\frac{827}{2533} = 0,32649$
I food	2533	1446	0	$\frac{0+1}{2533+1446} = 0,00025$	$\frac{0}{2533} = 0$

Tabelle 2.2.: Beispiel der Berechnung von Laplace

Als Alternative zu *Laplace (add-one)* kann *add-k* benutzt werden. Die Variable *k* ist in diesem Fall zwischen 0 und 1 und verhindert eine zu hohe Wahrscheinlichkeitsverteilung von den bekannten N-Grammen zu den noch nicht bekannten N-Grammen.

$$P^*_{Add-k}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+V*k} \tag{2.7}$$

Katz Backoff Der *Katz Backoff Algorithmus* (entwickelt 1987 von Slava M. Katz) verwendet rekursiv die (N-1)-Gramme eines N-Gramms, wenn dessen *Counts* einer Wortfolge Null ist. Es wird soweit rekursiv N-1 ausgeführt bis das (N-1)-Gramm eine Anzahl von mindestens eins

hat.
Alle bekannten N-Gramme bekommen eine Wahrscheinlichkeit P^*, etwas geringer als die von MLE P. Bei den unbekannten N-Grammen wird die Wahrscheinlichkeit des (N-1)-Gramms (oder kleiner) mit der Gewichtung α multipliziert.

$$P_{Katz}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } C(w_{n-N+1}^{n}) > 0 \\ \alpha(w_{n-N+1}^{n-1}) P_{Katz}(w_n|w_{n-N+2}^{n-1}) & \text{otherwise.} \end{cases} \tag{2.8}$$

Beispiel:
Wäre $C(\text{April is the}) = 0$ und $C(\text{is the}) \neq 0$ dann
$P_{Katz}(\text{the}|\text{April is}) = \alpha(\text{April is}) P^*(\text{the}|\text{is})$

Kneser-Ney Smoothing ist der in der Praxis am häufigsten verwendete Smoothing Algorithmus (Jurafsky und Martin, 2008, 2014). Im Gegensatz zu *Katz Backoff* und *Laplace* betrachtet *Kneser-Ney* andere Wortfolgen, in dem das Wort w_n vorkommt. Es wird eine Heuristik berechnet, mit der eine bessere Aussage getroffen werden kann, mit welcher Wahrscheinlichkeit ein Wort in einem unbekannten Kontext vorkommen kann. Dazu werden alle unterschiedlichen Wortfolgen einer N-Gramm Ordnung des Wortes w_n zusammen gezählt. Ein Wort kommt mit einer höheren Wahrscheinlichkeit in einem neuen Kontext vor, wenn es bereits in mehr als nur einem Kontext zu finden war. Wörter, die häufig nur in einem Kontext gesehen werden, haben demzufolge eine geringere Wahrscheinlichkeit in dem aktuellen Kontext aufzutreten. Ein beliebtes Beispiel ist das Wort *Francisco*. Es kommt häufig in Sätzen vor, allerdings nur in Zusammenhang mit *San* in *San Francisco* (Chen und Goodman, 1999). Hier ein Beispiel der Formel für Bi-Gramme:

$$P_{KN}(w_i|w_{w-1}) = \underbrace{\frac{max(C(w_{w-1}, w_i) - d, 0)}{C(w_{i-1})}}_{\text{I}} + \underbrace{\lambda(w_{i-1})}_{\text{II}} \underbrace{P_{Continuation}(w_i)}_{\text{III}} \tag{2.9}$$

I Der erste Teil nimmt die *Counts* des Bi-Gramms (w_{w-1}, w_i), subtrahiert davon einen festen *Discount* d. Diese Variable d ist ein vorher festgelegter Wert zwischen 0 und 1. Ist $C(w_{w-1}, w_i)$ kleiner als d so würde die Wahrscheinlichkeit eine negative Zahl annehmen, deshalb wird das Maximum von den *Counts* abzüglich des Discounts und 0 genommen. Das Ganze wird dividiert mit der Anzahl der überhaupt vorkommenden vorangestellten Wörter $C(w_{i-1})$. Wichtig ist hierbei, dass die vorangestellten Wörter auch das Symbol für den Satzbeginn sein können, aber nicht das Symbol für das Ende des Satzes.

II Die Lambda Funktion $\lambda(w_{i-1})$ normiert den Discount mit

$$\lambda(w_{i-1}) = \frac{d}{C(w_{i-1})}|w : C(w_{i-1}, w) > 0| \tag{2.10}$$

Der Discount wird dividiert durch die *Counts* des vorangehenden Wortes und das Ganze wird multipliziert mit der Anzahl aller Wörter, die auf w_{i-1} folgen können und dessen *Counts* größer als 0 sind.

III $P_{Continuation}(w_i)$ ist der Teil, der den *Kneser-Ney Algorithmus* ausmacht. Es ist die Wahrscheinlichkeit, der ein Wort in einem unbekannten Kontext folgen kann.

$$P_{Continuation}(w_i) = \frac{|w_{i-1} : C(w_{i-1}, w_i) > 0|}{|(w_{j-1}, w_j) : C(w_{j-1}, w_j) > 0|} \tag{2.11}$$

Diese Wahrscheinlichkeit wird berechnet anhand der Anzahl der unterschiedlichen vorangehenden Wörter vor w_i, dividiert durch alle möglichen N-Gramme (*hier: Bi-Gramme*), dessen Anzahl größer als 0 ist.
$P_{Continuation}(w_i)$ lässt sich auch mit

$$P_{Continuation}(w_i) = \frac{|w_{i-1} : C(w_{i-1}, w_i) > 0|}{\sum_{w_j} |w_{j-1} : C(w_{j-1}, w_j) > 0|} \tag{2.12}$$

berechnen.

2.4. Vergleichbare Arbeiten

Satzverbesserung Verwandte Arbeiten konzentrierten sich auf das Lösen von Lückentexten mit vorgegebenen Füllwörtern (Spiccia u. a., 2015; Lee und Lee, 2014) oder auf Fehlererkennung von Wörtern verschiedenster Art (Hein, 1998; Samanta und Chaudhuri, 2013).

Spiccia u. a. (2015) nutzen eine *Word-Document Frequency Matrix* und eine *Word-Word Frequency Matrix* zum Lösen des Lückentextes. Matrizen seien weniger kostspielig mit wachsendem *n* als N-Gramme und es gäbe weniger 0-Wahrscheinlichkeiten als bei N-Grammen mit höherem *n*. Bei den beiden Matrizen wird eine Singulärwertzerlegung durchgeführt, sodass die Matrizen in ihre charakteristischen Eigenschaften zerlegt werden. Mit Hilfe dieser werden Codierungen berechnet, die später für die Berechnung des fehlenden Wortes verwendet werden. Spiccia u. a. (2015) verglichen deren Methode mit anderen Methoden aus der Literatur mit identischen Trainingsdaten. Sie kamen zu dem Ergebnis, dass die Kombination aus *Word-Document*

und *Word-Word Frequency Matrix* mit einer höheren Treffsicherheit das fehlende Wort finden als z.B. N-Gramme.

Lee und Lee (2014) kombinieren N-Gramme von $n = 2$ bis $n = 5$. Zusätzlich wird die Wahrscheinlichkeiten der m Wörter ($m = 1$ bis $m = 4$) berechnet. Die Sequenzen sehen dann z.B. folgendermaßen aus:
$m = 3, n = 1\ P(w_{i-2}, w_{i-1}, w_i|w_{i-3})$
$m = 2, n = 2\ P(w_{i-1}, w_i|w_{i-2})$
$m = 1, n = 3\ P(w_i|w_{i-1}, w_{i-2}, w_{i-3})$
Der Schwerpunkt lag im Füllen des Lückentextes auf Basis von *Web-Scale Data*. Das beste Ergebnis erzielten Lee und Lee (2014) mit 4-Grammen und *Laplace-Smoothing*. Sie erreichten eine Fehlerfreiheit von 85,8%.

Hein (1998) konzentriert sich auf die Fehlerverbesserung bestehender Wörter. Die Fehlerverbesserung bezieht sich bei ihr auf die schwedische Sprache. Sie unterscheidet zwischen dem *spell-checker* (Rechtschreibüberprüfer), der nur einzelne Wörter betrachtet und dem *grammar checker* (Grammatiküberprüfer), der Wörter und ihren Kontext betrachtet. Hein (1998) entwickelte den *ScarCheck*, der sich in zwei Module, dem Parser und dem Scanner aufteilt. Der Parser baut aus dem Satz eine Grammatikstruktur und der Scanner traversiert über diese Struktur und meldet Fehler. Die Fehlermeldung des Scanners beinhaltet den Kontext, in dem der Fehler aufgetreten ist und ebenso eine textuelle Beschreibung des Fehlers. Eine Schwierigkeit der Fehlersuche läge darin, sich für einen Fehler zu entscheiden. Sind z.B. ein Nomen und ein Verb in ihrem Numerus unstimmig, müsste in beiden Richtungen korrigiert werden. Beeinflusst ein Fehler mehrere Wörter, so müssten alle Wörter korrigiert werden. Aus Laufzeitgründen entschied sich Hein (1998) dazu nur das erste Wort zu verbessern.

Samanta und Chaudhuri (2013) kategorisieren Fehler in die *non word* (unechtes Wort) Fehler und in die *real-word* (echt-Wort) Fehler. In dem Paper werden aber nur *real-word* Fehler betrachtet, die z.B. häufig bei der Autokorrektur vorkommen. Für die Fehlersuche werden die Sätze in Bi-Gramme und Tri-Gramme aufgeteilt. Linkes Bi-Gramm ($w_{i-1}w_i$), rechtes Bi-Gramm (w_iw_{i+1}) und Tri-Gramm mit dem untersuchtem Wort in der Mitte ($w_{i-1}w_iw_{i+1}$). Es wird ein *Confusion Set* [2] aufgebaut, das alle Wörter beinhaltet, die sich aus nur einer einzelnen Operation ergeben. Einzelne Operationen sind z.B. Entfernen, Hinzufügen und Austauschen eines Buchstabens im Wort. Es wird der MLE Algorithmus verwendet, um die Wahrscheinlichkeiten der N-Gramme zu berechnen. Der Score eines Wortes ergibt sich aus der Addition des rechten und linken Bi-Grammes und des Tri-Grammes und dessen einzelnen

[2] *Confusion Set* ist die Menge von Wörtern, die mit einem Wort vertauscht werden könnten z.B. {his, him, this, is, has} für das Wort *his*

Gewichtungen. Der Score liegt zwischen 0 und 1. Das Wort mit dem höchsten Score wird verwendet.

Elektronische Kommunikationshilfe Elektronische Kommunikationshilfen gibt es in unterschiedlichen Ausführungen mit unterschiedlich großem Vokabular.

MyCORE ist eine Kommunikationshilfe, die zwischen Kern- und Randvokabular unterscheidet (RehaMedia GmbH). Das Konzept von *MyCORE* wurde von Wissenschaftlern der Universität zu Köln, als *Kölner Vokabular*, entwickelt. Das Kernvokabular besteht aus 200-300 Wörtern, die zu jedem Thema verwendet werden können und ca 80% der Alltagssprache ausmachen (Sachse und Boenisch, 2013). Das Randvokabular ist themenspezifisch nach „der kindlichen Erfahrungswelt "(Sachse u. a., 2013) sortiert und kann, zumindest in der reduzierten Version, mit fortschreitendem Alter des Kindes erweitert werden (RehaMedia GmbH; Sachse u. a., 2013). Dabei bleiben bestehende Symbole immer an der selben Stelle. Die Grammatik von *MyCORE* in der Vollversion ist komplex, denn sie beinhaltet alle möglichen Konjugationen. Verben können in der Funktion *Automorphen* automatisch konjugiert werden, allerdings generiere dies teilweise falsche Formen. Deshalb gibt es ebenfalls die manuelle Grammatik. Das Kind kann die Verb-Endungen selbständig wählen und so die Grammatik lernen und die Verwendung fördern (Sachse u. a., 2013). *MyCORE* gibt es in drei Versionen, der Vollversion mit circa 2400 Wörtern, der reduzierten Version mit ungefähr 800 Wörtern und *MyCORE* mini mit circa 300 Wörtern und größeren Symbolen (RehaMedia GmbH). Die reduzierte Version blendet lediglich Vokabeln aus der Vollversion aus, sodass diese nach und nach wieder eingeblendet werden können. Die *MyCORE* Software verwendet die METACOM-Symbole von Frau Kitzinger (Kitzinger, 2016; RehaMedia GmbH).

Eine weitere elektronische Kommunikationshilfe ist der *Accent 1400*, der die Kommunikationssoftware *NuVoice* nutzt (Prentke Romich GmbH). Laut Prentke Romich GmbH „verfügt der Accent 1400 über eine lernfähige Wortvorhersage, eine IR-Umfeldsteuerung und die Möglichkeit der Computersteuerung.". Die Verben auf dem Gerät werden bereits auf der Oberfläche konjugiert. In der zweiten Zeile des Startbildschirms befinden sich die Pronomen. Das Kind kann auf ein Symbol im mittleren Bereich des Bildschirms (Tasten weiß hinterlegt) drücken und ein passendes Verb zu dem Symbol erscheint konjugiert in der zweiten Zeile des Bildschirms. Auch die jeweilige Vergangenheitsform, Partizip II, wird angezeigt. Die Hintergrundfarbe eines Symbols sagt aus, um was für ein Wort es sich handelt. So bedeutet z.B. die Hintergrundfarbe *hellgelb* eine *Kategorie für Verben* und *lachsfarben* bedeutet die Wortart *Nomen*. Bei *NuVoice* gibt es zwei Wortvorhersagesysteme: *PRD-Wortvorhersage* und *WordQ-Wortvorhersage* (Babst, 2016). *WordQ* kann zusätzlich eine Vorhersage zu dem nächsten Wort treffen (Babst, 2016). Die

Wortvorhersagesysteme müssen in den Einstellungen eingeschaltet werden und befinden sich dann in der Zeile unter dem Textfenster. Die Wortvorhersage wird nur im Schriftmodus (einzelne Buchstaben) angezeigt. Babst (2016) erklärt, dass Wortvorhersagen in der Regel zunächst in der Grundform angezeigt werden. Der Benutzer muss weitere Buchstaben eingeben, um die Auswahl zu verfeinern. Wortvorhersagen mit einer ~ weisen eine Gruppe von einem Wort auf. Nach dem Berühren dieses Wortes, kommen weitere Vorschläge mit dem gleichen Wortanfang. Beispielsweise kommt nach *haus~*, die Auswahl *Haus*, *Haustier* und *Haustür* etc..
Es kann eingestellt werden, wie viele kürzlich verwendete Wörter (maximal 11) angezeigt werden sollen. Der restliche Platz wird für *häufig verwendete Wörter* eingesetzt (Babst, 2016).

3. Realisierung von *Talk To Me*

Dieses Kapitel beschreibt zunächst die Anforderungen an *Talk To Me*, die mit Hilfe eines Anwendungsfalls herausgefiltert werden. Darauf aufbauend werden Komponenten modelliert und vorgestellt. Besonders die Komponente der Satzverbesserung hat hier ihren Schwerpunkt. Zum Schluss wird das Zusammenspiel der Komponenten aufgezeigt.

3.1. Anwendungsfall

„Mit Anwendungsfällen werden funktionale Anforderungen an ein System aus der Außensicht formuliert." (Ludewig und Lichter, 2010). Damit funktionale Anforderungen bei *Talk To Me* formuliert werden können, verfasst die Autorin einen Hauptanwendungsfall, den *Satz verbessern* Anwendungsfall. Dieser wird in der Notation wie in Ludewig und Lichter (2010) beschrieben, die auf der Notation von Cockburn (2000) basiert. Ein weiterer möglicher Anwendungsfall wird nicht betrachtet, da er für die Außensicht nicht relevant ist. Dieser Anwendungsfall ist das Wechseln zwischen Bi-Gramm und Tri-Gramm, das nur zum Testen und zum Auswerten benötigt wird.

Name	Satz verbessern
Ziel	Der Benutzer möchte seinen eingegebenen Satz verbessert haben.
Vorbedingung	Die Applikation wurde erfolgreich gestartet
	Wortsymbole sind auf dem Display zu sehen
Nachbedingung	Der Benutzer sieht auf dem Display den korrigierten Satz
	Der korrigierte Satz wird laut ausgesprochen
Nachbedingung	Der Benutzer hat die Möglichkeit einen Satz einzugeben
Sonderfall	
Akteure	Benutzer

Normalablauf	1. Der Benutzer gibt einen Satz in englischer Sprache über die Symbole ein.
	2. Der Benutzer drückt nach vollständiger Eingabe auf den Sendeknopf
	3. Die mobile Anwendung liest den Satz und schickt ihn an das Satzverbesserungssystem (SvS)
	4. Das SvS verbessert den Satz
	5. Die Anwendung zeigt den korrigierten Satz auf dem Display an
	6. Die Anwendung spricht den korrigierten Satz laut aus
Sonderfall 3a	Satz SvS ist nicht erreichbar
	3a.1 Die Anwendung zeigt eine Fehlermeldung an
	3a.2 Der Benutzer hat die Möglichkeit einen Satz einzugeben
Sonderfall 6a	Das Gerät ist auf Stumm gestellt
	6a.1 Der Benutzer erhöht die Lautstärke bzw. deaktiviert den Stummmodus
	6a.2 Der Benutzer drückt den *Aussprechen wiederholen* Knopf
	6a.3 Die Anwendung fragt das SvS erneut nach der Audiodatei
	6a.4 Das SvS antwortet mit der Audiodatei
	6a.5 Die Anwendung spricht den korrigierten Satz laut aus
	6a.6 Der Benutzer hat die Möglichkeit einen Satz einzugeben

3.2. Anforderungen an das System

Die Anforderungen sind in zwei Gruppen geteilt. Zum einen in die funktionalen Anforderungen, zum anderen in die nichtfunktionalen Anforderungen. Diese Anforderungen ergeben sich unter anderem aus dem oben aufgezeigten Anwendungsfall.

3.2.1. Funktionale Anforderungen:

1. Das System nimmt einen grammatikalisch falschen Satz entgegen und gibt einen grammatikalisch richtigen Satz zurück. Die Sprache des Satzes ist Englisch.
2. Die Sprache der UI zur Eingabe und Ausgabe ist Englisch.
3. Das System kann die Grammatik größtenteils eigenständig verbessern.
4. Die Applikation basiert auf Symbolen bzw. Bildern zur Eingabe von Worten.
5. Die Applikation kann einen Satz laut aussprechen.

3.2.2. Nichtfunktionale Anforderungen:

1. Das System läuft auf einem Android Tablet mit der minimalen Android-Version 4.1.x., um auf ungefähr 98% (Statista GmbH, 2017) [1] aller Android Geräten laufen zu können.
2. Das System stellt Schnittstellen bereit, um andere UIs anbinden zu können (z.B. für andere Betriebssysteme).
3. Das System ist leicht bedienbar. Das heißt ein Kindergartenkind im Alter von drei bis vier Jahren kann das System ohne langes Überlegen benutzen. Hierzu wird aber eine kurze Einführung vorausgesetzt.
4. Das System kann den eingegebenen Satz nach Abschicken der Eingabe innerhalb von fünf Sekunden aussprechen, sofern eine stabile Verbindung zum Server besteht.

3.3. Komponenten

Aus den obigen Anforderungen lassen sich Komponenten modellieren. Aus den nichtfunktionalen Anforderungen Punkt 1 und Punkt 2 sind zwei Komponenten erkennbar. Zum einen ein auf Android laufendes Frontend und zum anderen ein unabhängiges Backend. Die Backend Komponente lässt sich zudem in eine Grammatikkomponente (vgl. funktionale Anforderungen Punkt 3; In Abbildung 3.1 genannte *Grammar Component*) und eine Sprachverarbeitungskomponente (vgl. funktionale Anforderung Punkt 5; In Abbildung 3.1 genannte *Text processing Component*) aufteilen. Dabei soll die Grammatikkomponente einen grammatisch inkorrekten Satz entgegennehmen und einen grammatisch korrekten Satz zurückliefern. Die Sprachverarbeitungskomponente nimmt einen beliebigen Satz entgegen und liefert eine Audiodatei zurück. Mit dieser Aufteilung haben die Komponenten eine hohe Kohäsion und geringe Kopplung und können anderweitig wiederverwendet werden. Die Entscheidung für die Sprachverarbeitungskomponente fiel auf eine externe Api. Die gewählte Api besitzt eine gute Anbindung sowie eine große Auswahl an synthetischen Sprachen. Die Stimmen klingen weniger nach einer typischen *Computerstimme* bzw. *Roboterstimme*.

Interne Komponenten Die internen Komponenten sind in Abbildung 3.1 innerhalb der Box mit der Überschrift *Talk To Me*. Hierzu gehören die Android Applikation als Frontend (siehe in Abbildung 3.1 Frontend Component) und das, in der Programmiersprache *Python* implementierte Backend (siehe in Abbildung 3.1 Backend Component).

[1] Angaben sind von *Statista* im Zeitraum 30. Juni bis 06. Juli 2017. Es wird nicht zwischen Smartphone und Tablet unterschieden.

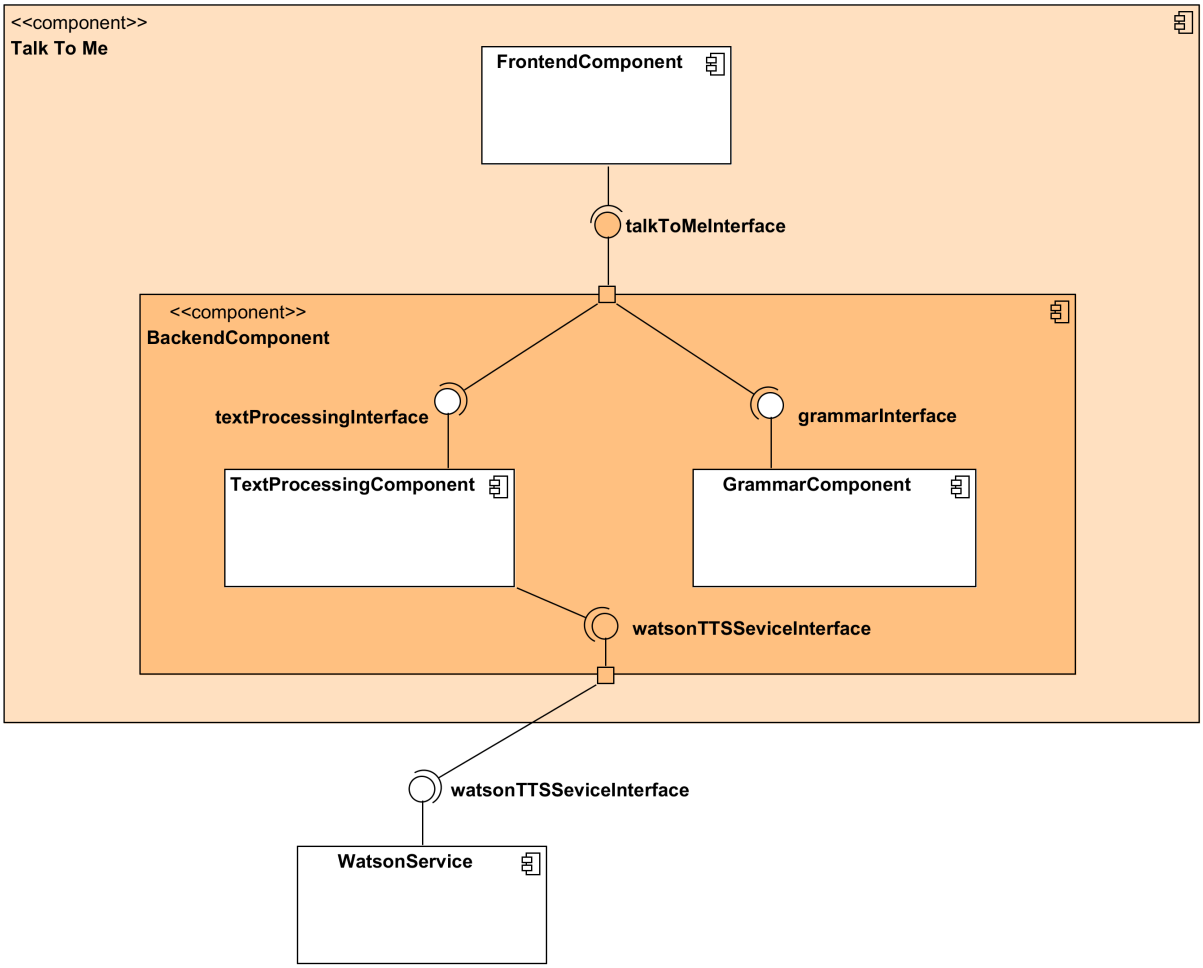

Abbildung 3.1.: Technisches Komponentendiagramm

Externe Komponente Die externe Komponente ist ein *Watson Service* (siehe Abbildung 3.1). Sie enthält den Service *Text-to-Speech* von IBM Watson.

3.3.1. Frontend Android Anwendung

Das Frontend soll durch eine Tablet Applikation realisiert werden. Zum einen, weil heutzutage die meisten Kinder Zugang zu einem Tablet haben (Statistisches Bundesamt [Destatis], 2017) und zum anderen, weil ein Tablet meist groß genug ist für eine Anwendung wie diese. Zudem soll die Anwendung, in diesem Fall, auf einem Android Gerät laufen.

Design Ein wichtiger Punkt ist die Gestaltung der Applikation, sodass sie ansprechend für Kinder ist. Es werden kräftige Farben wie die Primärfarben Gelb und Blau und die Sekundärfarben Orange gewählt. Als Symbole werden die Metacom Symbole verwendet (Kitzinger, 2016). Diese sind in einem einheitlichen Stil gehalten und sollen daher klar zu verstehen sein (siehe Abbildung 3.3).

In der Regel haben Kinder zwischen drei Jahren und sechs Jahren das Lesen und Schreiben noch nicht erlernt. Die Applikation darf zur Folge nicht auf Worten basieren, sondern muss Bilder oder Symbole enthalten. Hier liegt die Schwierigkeit die Wörter mit Hilfe von Bildern eindeutig darzustellen. Tiere und Nomen wie *Haus* (engl. *house*) oder *Apfel* (engl. *apple*) sind leicht darzustellen. Schwierig wird es bei Verben. Das Verb *laufen* (engl. *run*) kann durch eine Person in Laufbewegung dargestellt werden, aber das Verb *sein* (engl. *be*) kann nicht vernünftig visualisiert werden. Ein Symbol bzw. Bild kann nicht mehr für nur ein Wort stehen, sondern es müsste für einen Satz gelten, der noch anpassbar ist. Möchte ein Kind z.B. sein Alter verraten, kann es dies über drei Bilder.
Das erste Bild zeigt eine Person mit der Hand auf sich gerichtet, das zweite Bild eine Hand mit der jeweiligen Anzahl von Fingern, die das Alter des Kindes zeigen und das dritte Bild einen Jahreskalender (siehe Abbildung 3.2).

Abbildung 3.2.: Beispiel für den Satz *I am two years old*

Vergangenheitsformen für Kinder darzustellen ist ebenfalls schwierig. Hier muss es eine zweite Ebene mit nur Verben in der Vergangenheit geben. Diese Anwendung deckt nur Sätze in Präsensform ab.

Die meisten Wörter sind in frei gewählten Kategorien gefasst. Diese Kategorien sind aus der erwachsenen Sicht erstellt. Wie Sachse u. a. (2013) erklären, gibt es Unterschiede zwischen Kindern und Jugendlichen bzw. Erwachsenen in der Zuordnung von Wörtern in Kategorien. Jugendliche und Erwachsene ordnen Wörter eher nach gleichen Wortarten, wohingegen Kinder Wörter nach ihren alltäglichen Erfahrungen sortieren; z.B. weisen Kinder, laut Sachse u. a. (2013), das Wort *duschen* und *Dusche* der gleichen Kategorie zu, Erwachsene eher *baden* und *duschen*. Diese Aufteilung sollte im Folgenden beachtet werden.
Die Anwendung hat 21 Kategorien (siehe dunkler orange gefärbte Symbole in Abbildung 3.3). Darunter sind

doctor, professions, electronics, fantasy, colors and shapes, letters and numbers,

feelings, house, clothing, small words, sports, instruments, hygiene, bodyparts,
people, meal, fruits/veggis, toys, sweets, animals, verbs, nature, seasons

Einige Wörter (yes, no, do not know, and, or, thank you, you are welcome, please, hello, name) sind keiner Kategorie zugeordnet und befinden sich direkt auf der Hauptseite, da sie leicht zugreifbar sein sollen. Die Kategorien *small words* und *verbs* beinhalten die meisten Wörter. Bei *verbs* liegt es an der groben Benennung der Kategorie. Verben werden nicht in mehrere Kategorien wie z.B. *outside verbs* mit *run, walk, drive, ride* und *inside verbs* mit *stick, sleep, draw* aufgeteilt. Gleiches gilt für die Kategorie *Small verbs*, außerdem sind kleinere Wörter wie *in, on, after, right, by* unabhängig vom Kontext. Sie gehören zu den ersten 300 Worten einer Sprache (siehe auch Abbildung 2.2) und sollten gut zugänglich sein. Da der Platz auf der Hauptseite nicht für alle Wörter ausreicht, sind diese *kleinen Wörter* in einer eigenen Unterkategorie einsortiert und durch einen Klick abrufbar.

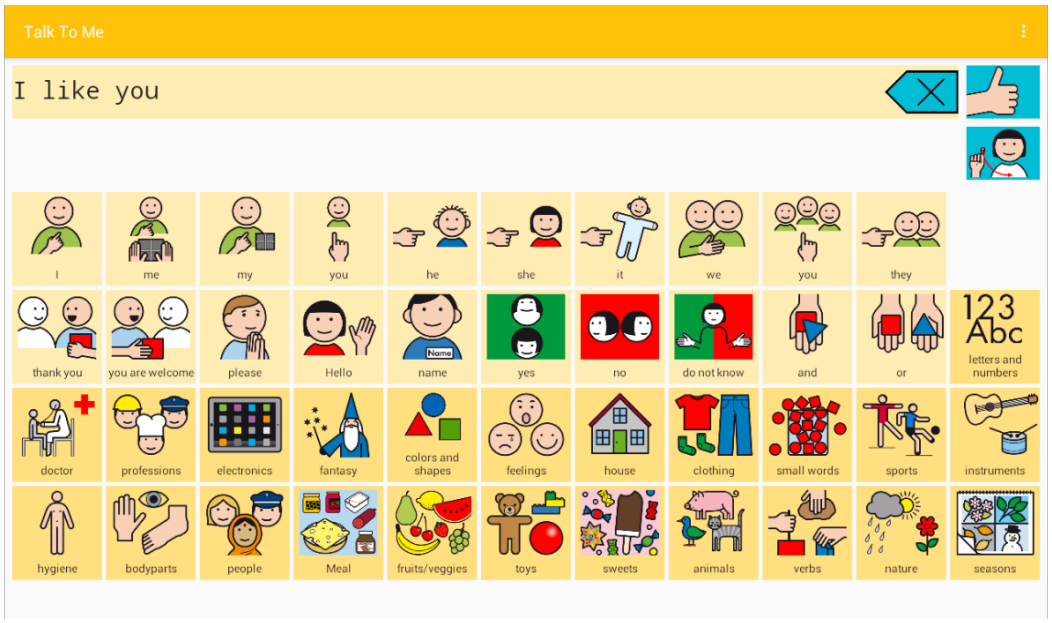

Abbildung 3.3.: Startbildschirm der Applikation *Talk To Me* mit eingegebenem Satz „I like you“. Weitere Screenshots befinden sich in Abschnitt A.3

Funktionalität Kinder mit Sprach- bzw. Sprechstörungen geben mittels Symbole und Bilder einen Satz ein. Im Anzeigefeld wird dieser Satz in Worten angezeigt. Dieser Satz kann zunächst noch falsch sein. Mit Bestätigen der OK-Taste (Button neben dem Eingabefeld mit dem Daumen hoch) wird die Schnittstelle des Backends aufgerufen und der Satz übergeben. Dort wird der

Satz verarbeitet und eine Audiodatei der mobilen Anwendung zur Verfügung gestellt. Die Audiodatei wird von der mobilen Anwendung automatisch abgespielt (siehe Abbildung 3.4).

3.3.2. Backend Python Anwendung

Das Backend ist in der Programmiersprache *Python* geschrieben. Es nimmt einen Satz entgegen, korrigiert ihn, gibt ihn weiter an die Sprachverarbeitungskomponente und liefert eine Audiodatei zurück.

Eine REST-Schnittstelle abstrahiert serverseitige Funktionen der Anwendung. Die Schnittstelle wird mit der Python Library *Bottle* [2] realisiert. Dieses Framework ist sehr einfach gehalten und reicht für die gegebenen Anforderungen aus. Die Schnittstelle stellt zwei Funktionen zur Verfügung. Eine Funktion nimmt einen Satz entgegen und gibt den korrigierten Satz sowie eine Referenz zur Audiodatei zurück.

Request	**Response**

GET /sentence/<text>	{
	„correct_text“: „<corrected text>“,
	„origin_text“: „<text>“,
	„file_path“: „/audio/<file name>.ogg“,
	„time_needed“: „<time in msec>“
	}

Tabelle 3.2.: Spezifikation Request zur Satzverbesserung

Die andere Funktion nimmt den Pfad zur Audiodatei entgegen und liefert die Audiodatei zurück.

Request	**Response**
GET /audio/<filename.ogg>	Binary file

Tabelle 3.3.: Spezifikation Request für die Audiodatei

Weitere Funktionen, wie das Anfragen eines Tokens zur sicheren Kommunikation im Internet, werden hier nicht weiter thematisiert.

Durch die Aufteilung in zwei Funktionen, *Satzkorrektur* und *Audiodatei*, lassen sie sich flexibel einsetzten. Bei GET /sentence/<text> wird eine Audiodatei bereits erzeugt, aber es

[2] Bottle ist ein einfaches Webframemwork u.a. für Rest-APIs für Anwendungen in Python geschrieben. Es beinhaltet nur die nötigsten Funktionalitäten. Für weitere Informationen siehe http://bottlepy.org/docs/dev (abgerufen am 20.08.2017)

wird nur eine Referenz zu dieser Datei zurück geliefert. Diese Funktion lässt sich also auch für eine Satzverbesserung verwenden, ohne eine Stream-Verarbeitung implementieren zu müssen. Hat der Client einmal den Pfad zur Audiodatei erhalten, kann er diesen mit Hilfe von GET /audio/<filename.ogg> aufrufen und die Audiodatei, so häufig wie nötig, abfragen. Der Client ist nicht verpflichtet die Datei lokal zu speichern und die erste Anfrage muss nicht erneut für den Erhalt der Datei aufgerufen werden.

3.3.3. Backend: Trainingsdaten

Das Corpus ist zusammengestellt aus *Simple English Wikipedia* (Wikipedia, 2016) mit über 260.000 Wörtern. Die Part of Speech-Tags (POS-Tags) kommen aus der *Web-As-Corpus Kool Yinitiative (WaCky)* (Evert u. a., 2016) Sammlung. Eine vollständige Liste aller POS-Tags und ihre Bedeutungen befinden sich im Anhang Abschnitt A.1. Speziell wird die englische Version *ukWaC* verwendet. Die Sammlung besteht aus rund zwei Milliarden Wörtern *getagged* mit Hilfe des *TreeTaggers*[3].

Alle Daten werden nach der Verarbeitung in eine Key-Value NoSql-Datenbank abgelegt. In diesem Projekt wurde sich für Redis[4] entschieden, da diese Datenbank die erforderlichen Anforderungen optimal erfüllen kann und mit Python leicht zu verwenden ist. Am Anfang werden die benötigten Daten einmalig berechnet und in die Datenbank geschrieben. Danach wird nur noch aus Redis über den Key gelesen. Dadurch bleibt die Abfrage simpel und ist performant (Han u. a., 2011). Zusätzlich lassen sich Datenstrukturen wie Hashes, Listen und Sets in Redis abspeichern. Folgende Daten sind in der Datenbank abgelegt:

- Alle Wörter mit dem jeweiligen Count (Text aus Wikipedia)
- Bi-Gramme mit dem jeweiligen Count (Text aus Wikipedia)
- Bi-Gramme mit der berechneten Wahrscheinlichkeit (Text aus Wikipedia)
- Tri-Gramme mit dem jeweiligen Count (Text aus Wikipedia)
- Tri-Gramme mit der berechneten Wahrscheinlichkeit (Text aus Wikipedia)
- Alle Wörter und dessen POS-Tags (Text aus WaCky)
- POS-Tags als Bi-Gramm, zur Bestimmung einer Grammatik (Text aus WaCky)

Sowohl bei den Bi-Grammen als auch bei den Tri-Grammen wird die Wahrscheinlichkeit einmal für das Folgewort (bzw. zwei Folgewörter) sowie für das vorangehende Wort (bzw. zwei

[3] TreeTagger ist ein von Helmut Schmid (Dozent an der Ludwig-Maximilians-Universität München) entwickeltes Tool zum Erweitern von Texten um POS und Lemma Informationen. Für weitere Informationen siehe http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ (abgerufen am 20.08.2017).

[4] Redis ist eine von Salvatore Sanfilippo und Pieter Noordhuis entwickelte Key-Value NoSql-Datenbank. Sie ist Open Source und läuft hauptsächlich In-memory. Außerdem unterstützt sie unterschiedliche Datenstrukturen wie Strings, Hashes, Lists, Sets und Sorted Sets. Siehe https://redis.io (abgerufen am 20.08.2017).

Vorgängerwörter) berechnet. Die Wahrscheinlichkeiten sind mit Hilfe des MLE-Algorithmus berechnet und als log_{10} abgespeichert. Log-Wahrscheinlichkeiten sind nicht so klein wie normale Wahrscheinlichkeiten. Zudem verhindern Log Wahrscheinlichkeiten einen Fließkommaunterlauf (underflow), da die Zahlen miteinander addiert statt multipliziert werden und anschließend wie im Folgenden zu sehen als Exponent der Exponentialfunktion gerechnet werden.

$$exp(log_{10}(P(w_{i-1}w_i)) + log_{10}(P(w_iw_{i+1}))) = P(w_{i-1}w_i) * P(w_iw_{i+1})$$

Die meisten Daten aus der Liste liegen als Hash-Wert vor.

Hier ist ein Beispiel aufgezeigt, wie die Daten aus den Trainingsdaten *ukWaC* strukturiert sind:

<s>
For IN for
the DT the
gaming NN gaming
community NN community
jolt NN jolt
runs VVZ run
tournaments NNS tournament
and CC and
leagues NNS league
that WDT that
attract VVP attract
the DT the
best RBS best
European JJ European
players NNS player
and CC and
clans NNS clan
. SENT .
</s>

Die in dem Beispiel erste (<s>) und letzte (</s>) Zeile signalisieren den Anfang und das Ende des Satzes. Jede Zeile, bis auf die erste und letzte, hat den Aufbau: Wort | POS-Tag | Lemma. Für eine *POS-Tag Bi-Gramm Grammatik* wird lediglich die zweite Spalte einer jeweiligen Zeile verwendet und mit der nächsten Zeile zweite Spalte verbunden. Die POS-Tag Bi-Gramme sind

in diesem Beispiel dann folgende: (<s>, IN), (IN, DT), (DT, NN), (NN, NN), (NN, VVZ), (VVZ, NNS), (NNS, CC), (CC, NNS), (NNS, WDT), (WDT, VVP), (VVP, DT), (DT, RBS), (RBS, JJ), (JJ, NNS), (NNS, SENT), (SENT, </s>)
Duplikate werden gezählt und sind hier nicht doppelt aufgelistet.

Das zweite Beispiel ist ein Ausschnitt aus dem Corpus aus *Simple English Wikipedia*:

> <doc id="1" url="?curid=1" title="April">
> April
>
> April is the fourth month of the year, and comes between March and May.
> It is one of four months to have 30 days.
> April always begins on the same day of week as July, and additionally, January in leap years.
> April always ends on the same day of the week as December. [...]
> </doc>

Die Autorin hat, für die bessere Verarbeitung zu Bi-Grammen und Tri-Grammen, jeden Satz im Corpus in eine Zeile geschrieben. Zudem werden die ersten drei Zeilen (inkl. der Leerzeile) und die letzte Zeile bei der Verarbeitung ignoriert. Es werden ebenfalls Sätze mit weniger als drei Wörtern ignoriert sowie geprüft, ob ein Satz einen Punkt am Ende besitzt.

3.3.4. Backend: Satzverbesserung mit Bi-Grammen

Die Satzverbesserung sieht sich jedes Wort iterativ an. Dabei korrigiert sie noch nicht untersuchte Wörter. Alle vorherigen Wörter werden aufgrund von aktuellen Wortkorrekturen nicht rückwirkend verändert. Vergangene Wortkorrekturen (vom vorherigen Wort) beeinflussen jedoch die aktuelle Wortkorrektur. Die Satzverbesserung konjugiert ausschließlich Verben und dekliniert Nomen. Zusätzlich überprüft sie, ob das Wort dem vorangehenden Wort folgen kann. Eine POS-Grammatik liefert die benötigten Parameter. Diese wurde vorher aus den Trainingsdaten *ukWaC* (Evert u. a., 2016) generiert (siehe Unterabschnitt 3.3.3). Grammatikregeln, die sehr selten vorkommen werden ignoriert. So lässt sich an der Grammatik erkennen, dass ein Verb einem Nomen folgen kann. Genau genommen folgt etwa 5.575.911 Mal die Wortart *Verb* (inkl. Vergangenheitsform) der Wortart *Nomen*. Aber in selteneren Fällen oder nur in der Umgangssprache findet ein *Ausrufewort* (uh, ah, oh) seinen Platz nach einem Nomen. Insgesamt 624 Mal. Da 624 im Gegensatz zu 5.575.911 sehr gering ist, wird z.B. diese Grammatikregel nicht beachtet. Die verwendeten Trainingsdaten sind dafür zu groß und nicht auf Formalität geprüft.

Alle Wörter, die einem Symbol im Frontend zugeordnet sind, haben einen, maximal zwei POS-Tags fest zugewiesen bekommen. Diese sind in einer eigenen Datenbank hinterlegt und können so schnell abgefragt werden. Mit diesem Vorgehen wird die Entscheidung für die POS-Tags einiger Wörter erleichtert, denn in den Daten von *WaCky* können Wörter fünf oder mehr POS-Tags besitzen. Auf diese Weise kann ausgeschlossen werden, dass vordefinierte Wörter aus dem Frontend keine im Kontext falschen Tags besitzen. So wird beispielsweise verhindert, dass das POS-Tag *VV* für *cat* von *to cat* in dieser Anwendung verwendet wird.

```
correct_sentence(sentence_as_list)
   correct sentence = create new list
   correct sentence = add first elem from sentence_as_list

   for next word in sentence_as_list from second to last element
      previous word = last word from correct_sentence
      PoS previous word = PoS from previous word
      PoS next word = PoS from next word
      next PoS of previous word = next PoS as a list

      if PoS next word is Noun then
        noun = decline noun
      endif

      if PoS next word is Verb then
        verb = conjugate verb
      endif

      if PoS next word not in next PoS of previous word then
        missing word = find missing word between words
      endif

      if prob(missing word) <= prob(original word)
        word = max(prob(noun), prob(verb))
      else
        word = max(prob(noun), prob(verb), prob(missing word))
      endif

      correct sentence = add word to correct sentence
   endfor
```

Algorithmus 3.1: Pseudocode der Satzverbesserung für Bi-Gramme. *Part of Speech* wurde mit *PoS* abgekürzt

Der Pseudocode in Algorithmus 3.1 zeigt die Hauptfunktion der Satzverbesserung. Der im Frontend eingegebene Satz kommt als Liste in die Funktion. Das erste Wort des Satzes wird direkt zu dem korrigierten Satz hinzugefügt. Bei einem Bi-Gramm wird ein Startsymbol dem Satz hinzugefügt; beim Tri-Gramm zwei Startsymbole. Dieses wäre dann das erste Element im korrigierten Satz. Dann wird jedes Element aus der Liste bzw. jedes Wort aus dem Satz überprüft. Dabei werden sich das POS-Tag des vorherigen Wortes und auch die möglichen nachfolgenden POS-Tags als *folge-POS-Tag* Liste geholt. Je nachdem zu welchem POS-Tag das aktuelle Wort gehört wird es verbessert. In einigen Fällen stimmt das POS-Tag nicht mit einem aus der *folge-POS-Tag* Liste überein, hier wird dann nach möglichen Wörtern gesucht, die zwischen dem vorherigen und dem aktuellen Wort passen. Die Liste der *folge-POS-Tags* wird auf die sieben am häufigsten folgenden POS-Tags begrenzt, um unwahrscheinliche Wörter zu vermeiden. Am Ende jeder Iteration wird das Wort mit der höchsten Wahrscheinlichkeit, aus den drei Methoden der Satzkorrektur, als Folgewort genommen. *Verben konjugieren* und *Nomen deklinieren* werden im Normalfall disjunkt ausgeführt (Ausnahme sind Wörter, die sowohl Nomen als auch Verb sein können wie z.B. *swing*), wohingegen *Wörter hinzufügen* kumulativ zu den anderen durchgeführt werden kann. Es kann gelegentlich vorkommen, dass ein hinzugefügtes Wort die gleiche Wahrscheinlichkeit als Folgewort hat, wie das Ausgangswort. In diesem Fall wird das hinzugefügte Wort nicht weiter betrachtet und die verbesserten Wörter (falls vorhanden) aus *Verben konjugieren* und *Nomen deklinieren* werden verglichen.

Die folgenden Paragraphen beschreiben die drei verwendeten Vorgehen der Satzverbesserung: *Verben konjugieren*, *Nomen deklinieren* und *Wörter hinzufügen*.

Verben konjugieren Der Schwerpunk der Konjugation liegt ausschließlich bei der Präsensform eines Verbs. Es gibt drei Formen von Verben:
1. Infinitiv und konjugierte Form von *be* (POS: Infinitiv: VB, Plural: VBP, 3. Pers. sgl.: VBZ),
2. Infinitiv und konjugierte Form von *have* (VH, VHP, VHZ) und
3. alle anderen Verben (VV, VVP, VVZ).

Die Verben *be* und *have* sind unregelmäßige Verben. Sie lassen sich nicht, wie die anderen Verben, mit festen Regeln konjugieren. Für die Konjugation eines Verbs in die 3. Person Singular wird meist ein *-s* am Ende hinzugefügt. Ausnahmen sind: Verben, die auf *-o, -sh, -ch, -tch, -x, -z oder -ss* enden, bekommen *-es* hinzugefügt. Bei Verben, die auf einen Konsonanten und *-y* enden, wird das *y* zu *-ies* verändert.

Nomen deklinieren Nomen werden dekliniert, wenn das Vorgängerwort eine Zahl zwischen eins und 100 (POS: CD) oder ein Artikel wie *these*, *those*, *this* oder *that* (POS: DT) ist. Ein

Nomen wird in die Pluralform verändert, falls der Artikel plural ist und in die Singularform, falls der Artikel im Singular steht. Bei dieser Form muss der Artikel unmittelbar vor dem Nomen stehen. Der Satz *three little chicken* wird nicht zu *three little chickens* verändert, da sich bei einem Bi-Gramm nur maximal zwei Wörter angesehen werden können. Die Pluralform eines Nomens lässt sich als Symbol im Frontend dieser Anwendung nicht darstellen. Deshalb wird im Gegensatz zu der in Abschnitt 2.4 Vergleichbare Arbeiten erwähnten Arbeit von Hein (1998) nur in eine Richtung (das Nomen wird ausschließlich an die Zahl oder Artikel angepasst) korrigiert.

Sowohl für die Konjugation als auch die Deklination wurde das *pattern.en* Modul von der Python Lybrary von CLiPS Research Center (2017) verwendet.

Wörter hinzufügen Zusätzlich zu der Konjugation von Verben und Deklination von Nomen können Wörter hinzugefügt werden. Damit nicht jedes beliebige Wort hinzugefügt wird, fiel die Auswahl auf Wörter, die schwierig in Symbolen auszudrücken sind.

- Hilfsverben (be, have)
- Artikelwörter (a, the)
- Vorbestimmungswörter engl. predeterminer (all, both)
- Infinitiv Begleiter (to)
- Preposition (on, of)

Liste 3.1: Verwendete Wortarten zum Filtern von möglichen Wörtern

Die Satzverbesserung schaut sich bei einem Bi-Gramm zwei aufeinander folgende Wörter an. Die POS der Wörter werden ermittelt sowie die möglichen Nachfolge-POS des ersten Wortes. Das POS des zweiten Wortes kann in der Liste der möglichen Nachfolge-POS des ersten Wortes sein. In diesem Fall wird kein Wort hinzugefügt, da das Wort laut Grammatik an der Stelle richtig ist. Wenn es nicht in der Liste vorhanden ist, dann werden alle möglichen Nachfolge-POS mit den Wortarten in Liste 3.1 abgeglichen. Die Schnittmenge sind alle Wortarten (POS), die dem ersten Wort folgen und die POS der erlaubten Füllwörter sind. Das Füllwort mit der größten Wahrscheinlichkeit in diesem Kontext, wird als Ergebnis dieser Rechnung verwendet. Damit nicht zu viele Wörter an einer Stelle eingefügt werden und die Laufzeit beeinträchtigen, wurde sich für maximal ein Wort pro Satzstelle entschieden.

Es kann vorkommen, dass ein Wort einem anderen Wort mit hoher Wahrscheinlichkeit folgt, aber selten vor dem nächsten Wort steht. Damit die Wahrscheinlichkeitsberechnung nicht nur von einer Richtung (Leserichtung) abhängig ist, werden die Wahrscheinlichkeiten zweier Richtungen addiert und das Ergebnis als Exponent der Exponentialfunktion mit der eulerschen Zahl genommen.

$$exp(log_{10}(P(w_{i-1}w_i)) + log_{10}(P(w_i w_{i+1}))) \tag{3.1}$$

Im einfachen Fall, wie bei *Nomen deklinieren* und *Verben konjugieren* addiert die Satzverbesserung die Wahrscheinlichkeit von Wort A und B in Leserichtung (A, B) mit der Wahrscheinlichkeit von Wort A und B entgegen der Leserichtung (B, A). In dem Fall, wo ein Wort hinzugekommen ist, addiert die Satzverbesserung die Wahrscheinlichkeiten von Wort A und B (B ist das hinzugefügte Wort) in Leserichtung (A, B) und die Wahrscheinlichkeiten von B und C entgegen der Leserichtung (C, B). Die Wahrscheinlichkeiten entgegen der Leserichtung wurden zuvor ebenfalls berechnet und in die Datenbank geschrieben. Dazu sind die *Simple English Wikipedia* Trainingsdaten auch rückwärts gelesen worden.

Tabelle 3.4 zeigt die Iterationsschritte bei der Satzverbesserung mit Hilfe von Bi-Grammen und dem Beispielsatz *I like my brother*. Initial erhalten die Rückgabewerte von *Nomen deklinieren*, *Verben konjugieren* und *Wörter hinzufügen* einen *None*[5]-Wert als Wort und eine Wahrscheinlichkeit von minus Unendlich. Das bedeutet, dass alle Werte in der Tabelle bei *decline noun*, *conjugate verb* und *find missing word* die Werte *None* und *-inf* besitzen, wenn der POS-Tag des Wortes nicht für die Bedingung der Funktion ausreichend war.
Die POS-Tag Liste, die hier verwendet wurde, befindet sich im Anhang Abschnitt A.1.

[5]Der Wert ist *None*, da in der Programmiersprache Python *None* ein Singleton Objekt ist und kennzeichnet, dass z.B. die Variable keinen Wert hat. *None* ist vergleichbar mit *Null* in *Java* und *nil* in *Ruby*.

Iteration	Ausgabe
1	previous word **<s>** current word **I** probability of **<s>** and **I** is **0.00817635062672** possible next Part of Speech of word **<s>** with PoS **['<s>']** are **['DT', 'PP', 'NP', 'IN', 'NN', 'RB', 'JJ']** decline noun: **None** has probability of **-inf** conjugate verb: **None** has probability of **-inf** find missing word: **None** has probability of **-inf** current word: **I** has probability of **0.00817635062672** chosen word is **I**
2	previous word **I** current word **like** probability of **I** and **like** is **0.00375863920426** possible next Part of Speech of word **I** with PoS **['PP']** are **['MD', 'VVP', 'VVD', 'VBZ', 'VBP', 'VBD', 'RB']** decline noun: **None** has probability of **-inf** conjugate verb: **like** has probability of **0.00375863920426** find missing word: **None** has probability of **-inf** current word: **like** has probability of **0.00375863920426** chosen word is **like**
3	previous word **like**' current word **my** probability of **like** and **my** is **0.00276806174275** possible next Part of Speech of word **like** with PoS **['VV', 'VVP']** are **['IN', 'DT', 'RB', 'TO', 'PP', 'NN', 'JJ']** intersection of words between **like** and **my** is **set(['all', 'because', 'as', 'through', 'at', 'in', 'if', 'from', 'for', 'to', 'under', 'before', 'that', 'after', 'with', 'by', 'on', 'about', 'of', 'near', 'so', 'In'])** decline noun: **None** has probability of **-inf** conjugate verb: **None** has probability of **-inf** find missing word: **to** has probability of **0.0537218234478** current word: **my** has probability of **0.00276806174275** chosen word is **to my**

Iteration	Ausgabe
4	previous word **my** current word **brother** probability of **my** and **brother** is **0.00782312254313** possible next Part of Speech of word **my** with PoS **['PP$']** are **['NN', 'JJ', 'NNS', 'NP', 'JJS', 'VVG', 'RB']** decline noun: **brother** has probability of **0.00782312254313** conjugate verb: **None** has probability of **-inf** find missing word: **None** has probability of **-inf** current word: **brother** has probability of **0.00782312254313** chosen word is **brother**
5	previous word **brother** current word **</s>** probability of **brother** and **</s>** is **-inf** possible next Part of Speech of word **brother** with PoS **['NN']** are **['IN', 'NN', 'SENT', 'CC', 'NNS', 'TO', 'VBZ']** intersection of words between **brother** and **</s>** is **set([])** decline noun: **None** has probability of **-inf** conjugate verb: **None** has probability of **-inf** find missing word: **</s>** has probability of **-inf** current word: **</s>** has probability of **-inf** chosen word is **</s>**
	I like to my brother

Tabelle 3.4.: Iterationsschritte des Bi-Gramm Verfahren anhand des Beispielsatzes *I like my brother*

An dem Iterationsschritt 3 ist erkennbar, dass die Wahrscheinlichkeit für $exp(P(like, to) + P(my, to))$ höher ist (0.0537218234478) als für $exp(P(like, my) + P(my, like))$ mit 0.00276806174275. So wird sich für *... like to my ...* entschieden.

3.3.5. Backend: Satzverbesserung mit Tri-Grammen

Tri-Gramme betrachten einen größeren Kontext im Satz. Ein Vorteil ist, dass die vom Algorithmus hinzugefügten Wörter besser in den Satz passen. Der Nachteil ist, dass möglicherweise weniger oder keine passenden Tri-Gramme mit Wahrscheinlichkeiten existieren. Smoothing

aus Unterabschnitt 2.3.2 verhindert das nicht bekannte Tri-Gramme eine Wahrscheinlichkeit von 0 besitzen.

Der Algorithmus ist wie bei Bi-Grammen mit ein paar Ausnahmen. Die Berechnung der Wahrscheinlichkeit für ein Wort basiert auf vier Wörtern. Zwei vor (w_{i-2}, w_{i-1}) und eins nach (w_{i+1}) dem aktuellen Wort w_i und das Wort selbst. Das verbesserte Wort bei *Nomen deklinieren* und *Verben konjugieren*, ist das aktuelle Wort w_i. Alle vier Wörter sind bekannt und kommen aus dem Satz. Bei einem neu hinzugefügten Wort werden nur drei bekannte Worte aus dem Satz für die Wahrscheinlichkeitsberechnung benötigt, da das Wort selber nicht aus dem Satz stammt.

Die Tri-Gramme sind nicht eins zu eins von den Bi-Grammen übertragen, denn die Wahrscheinlichkeit von $P(w_i\ w_{i+1}\ w_{i+2})$ ist in den meisten Fällen 0. Eine Multiplikation der beiden Tri-Gramm $P(w_{i-2}\ w_{i-1}\ w_i)$ und $P(w_i\ w_{i+1}\ w_{i+2})$ würde das Ergebnis zu häufig verfälschen. Das liegt daran, dass die Wörter w_{i+1} und w_{i+2} in der Iteration noch nicht vorkamen und keine Verbesserung durchgeführt wurde. Wörter können in den weiteren Iterationsschritten noch hinzugefügt werden oder die Wörter w_{i+1} und w_{i+2} können noch konjugiert bzw. dekliniert werden. Dieses Problem wird mit der Wahl von $P(w_{i-1}\ w_i\ w_{i+1})$ nicht gelöst, aber eingeschränkt.

3.3.6. IBM Watson Text to Speech

Watson ist ein kognitives System des Unternehmens *IBM*. Es bietet unterschiedliche Services wie *Text to Speech*, *Speech to Text* oder *Conversation* an. In dieser Arbeit wird der Service *Text to Speech* verwendet. Ein Satz wird an die REST-Schnittstelle von Watson geschickt und eine Audiodatei wird als Byte Stream zurückgegeben. Die Audiostimme und Sprache ist bei *Watson* wählbar und lässt mögliche Spracheinstellungen in *Talk To Me* für die Zukunft offen. Der Klang der Stimme ist ähnlich wie die einer realen Person und weniger die einer *Roboterstimme*.

Kinder sollten sich mit der Stimme einer elektronischen Kommunikationshilfe einigermaßen identifizieren können. Noch besser als *Watson* wäre eine Kinderstimme, entweder eines Mädchens oder eines Jungen. An dieser Stelle wurde nicht weiter geforscht welche anderen *Text to Speech* Services eine Kinderstimme anbieten. Bei der Umsetzung von *Talk To Me* wurde auf eine Abstraktion geachtet, sodass sich der Sprachservice austauschen lässt.

3.3.7. Zusammenspiel der Komponenten

Die Abbildung 3.4 zeigt den Ablauf der Anwendung von der Eingabe eines Wortes bis hin zur Aussprache des korrigierten Satzes. Sie spiegelt den internen Ablauf des Anwendungsfalls aus

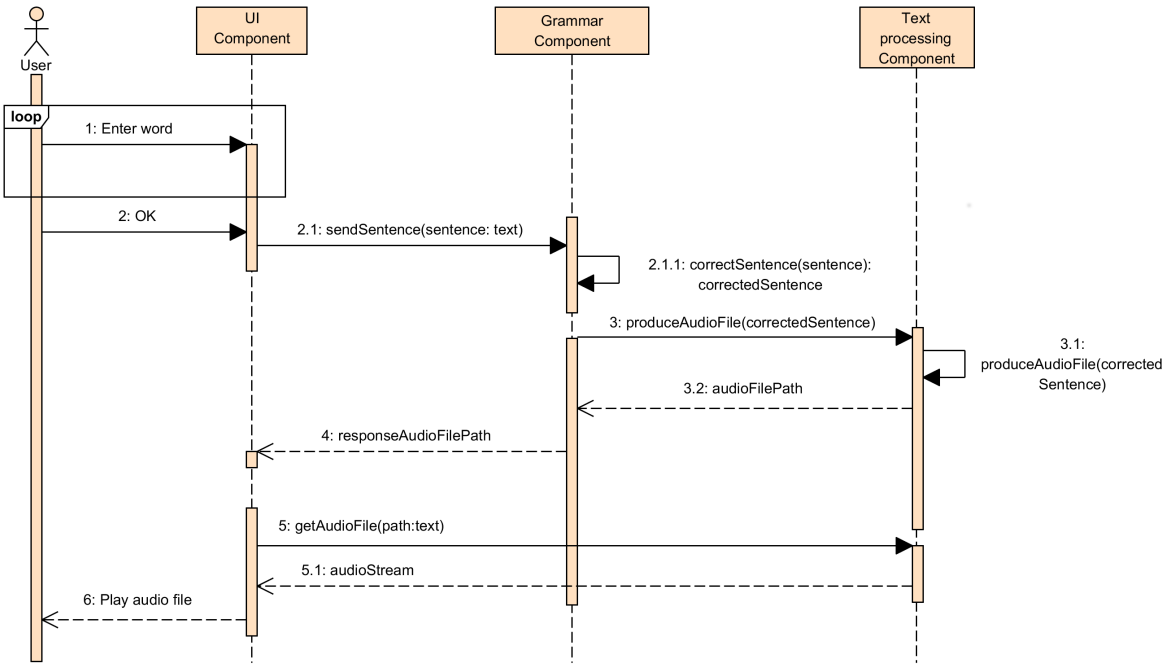

Abbildung 3.4.: Sequenzdiagramm: Zusammenspiel der internen Komponenten bei Eingabe eines Satzes

Abschnitt 3.1 wider.

Ein Benutzer gibt eine Reihe von Wörtern, die einen Satz bilden, ein (1). Er klickt auf den vorgesehenen Button zum Abschicken des Satzes an die Backendkomponente (2). Diese nimmt den nicht korrekten bzw. korrekten Satz entgegen (2.1). Der Satz wird von der Grammatikkomponente korrigiert (2.1.1) und anschließend an die Sprachverarbeitungskomponente gegeben (3). Die Sprachverarbeitungskomponente erzeugt aus dem korrigierten Satz eine Audiodatei im *Ogg* Dateiformat (3.1) und gibt den Pfad zu der angelegten Datei an die Grammatikkomponente zurück (3.2). Es wird mit einer JSON response inklusive des Pfades der Audiodatei an die UI geantwortet (4). Die UI parst die JSON Antwort und fragt die Backendkomponente, mit dem Pfad, die Audiodatei an (5). Diese wird von der Api an die zuständige Sprachverarbeitungskomponente weitergeleitet. Nun antwortet die Sprachverarbeitungskomponente mit der Audiodatei (5.1). Die UI kann diese dann abspielen (6).

4. Praxistest

4.1. Ausführung

Probanden unterschiedlichen Alters wurden gebeten Sätze in die Applikation *Talk To Me* einzugeben. Am Anfang wurde den Probanden der Sinn der Applikation und die Applikation selbst oberflächlich erklärt. Es wurde darauf hingewiesen, dass die Applikation nicht alle Wörter generiert, sondern nur bestimmte kleine Wörter. Beim Testen wurden die Probanden nicht beeinflusst, nur auf Nachfragen wurde geantwortet und bestimmte Anmerkungen wurden dokumentiert. Außerdem wurden keine Vorgaben zu den Eingaben gegeben. Die Probanden sollten sich selber beliebige Sätze ausdenken und wenn möglich, sich in das Alter eines kleinen Kindes (drei bis sechs Jahre alt) versetzten.

Während des Testens wurden alle eingegebenen Sätze, die auch berechnet wurden, automatisch in einer *csv* Datei gespeichert. Die Spalten der *csv* Datei lauten: *Eingegebener Satz*, *Korrigierter Satz* und *Satzverbesserungsverfahren*. Letzteres kann die Werte *Bi-Gramm*, *Tri-Gramm* oder *Tri-Gramm mit add-k-Smoothing* (k = 0,5) annehmen. Die Probanden testeten allerdings nur mit *Bi-Grammen*. Später, vor der Auswertung, wurden exakt die gleichen Sätze mit *Tri-Grammen* und *Tri-Grammen mit add-k-Smoothing* getestet. Am Ende jeder Durchführung konnten die Probanden ein Feedback zur Applikation geben.

Nachdem alle Probanden getestet hatten, wurden die Sätze überarbeitet. Das heißt, einige Sätze wurden so modifiziert, dass sie alle Wörter beinhalten, die auch bei der Eingabe zur Verfügung standen. Der Satz *I name XY* wurde z.B. zu *My name XY* verändert, da *My* in der UI verfügbar ist, aber die Probanden *My* zum Zeitpunkt der Eingabe nicht gefunden hatten. Andere Sätze wurden ganz herausgenommen, da sie z.B. keinen Sinn ergaben wie beispielsweise der Satz *actor rainbow cheese ocean beard toilet potty dry diaper*. Insgesamt kamen 137 gültige Sätze bei der Durchführung heraus.

Personen Die Probanden sind zwischen sechs und 26 Jahre alt. Ihr Vorwissen reicht von noch nicht eingeschult bis hin zu einem abgeschlossenen Studium in unterschiedlichen Fachrichtungen.

4.2. Auswertung

In diesem Abschnitt werden die Ergebnisse des vorher beschriebenen Praxistests ausgewertet. Zum einen wird die Satzverbesserung analysiert, zum anderen wird die Benutzbarkeit der UI untersucht. Alle ausgewerteten Sätze sind in Anhang Abschnitt A.2 zu finden.

4.2.1. Vergleich: Satzverbesserung mit Bi- und Tri-Grammen

Wie Samanta und Chaudhuri (2013) erklären, haben N-Gramme höherer Ordnung und niedrigerer Ordnung unterschiedliche Stärken und Schwächen. N-Gramme höherer Ordnung sind sensitiver gegenüber einem größeren Kontext, haben aber dafür spärliche Counts. Andersherum haben N-Gramme niedrigerer Ordnung einen kleineren Kontext, dafür mehr Counts.

Im folgenden Abschnitt werden alle Sätze in drei Formen ausgewertet und die drei in Abschnitt 4.1 genannten Satzverbesserungsverfahren verglichen. Tabelle 4.1 zeigt die Wahrscheinlichkeit der richtig korrigierten Sätze. Tabelle 4.2 zeigt die Anzahl der korrigierten Verben und Tabelle 4.3 die Anzahl der korrigierten Nomen.

4.2.1.1. Überblick der Satzverbesserung

	Bi-Gramm	Tri-Gramm	Tri-Gramm (add-k-smoothing)
Anzahl Sätze	137	137	137
Richtig korrigierte Sätze in %	29,9	36,5	35.0

Tabelle 4.1.: Wahrscheinlichkeiten der richtig korrigierten Sätze in allen Testsätzen mit unterschiedlichem Verfahren

Insgesamt wurden 137 Sätze getestet. In der Tabelle 4.1 ist erkennbar, dass knapp ein Drittel der Sätze mit Hilfe von Bi-Grammen und Tri-Grammen mit Smoothing richtig korrigiert wurden. Bei den reinen Tri-Grammen sind es immerhin fast 37%. Da Bi-Gramme einen kleineren Kontext haben, gibt es mehr Wörter die einem Bi-Gramm folgen, als bei einem Tri-Gramm. Somit werden mehr Wörter zu einem Satz hinzugefügt, die den Satz negativ verändern können. Zum Beispiel wurde der Satz *I eat and sleep* vom Bi-Gramm zu *I eat in and to sleep* verändert. Ein Blick in die Datenbank zeigt, dass die Wahrscheinlichkeit für *eat and* geringer ist als für *eat in and*. Hauptgrund hierfür ist, dass das Wort *and* viel häufiger in den Trainingsdaten vorkommt als das Wort *eat*. Bei der Wahrscheinlichkeitsberechnung wird durch diese Häufigkeit

dividiert und wie bekannt ist, wird das Ergebnis einer Division kleiner, je größer der Nenner ist. Gleichung 4.2 zeigt die Berechnung von *(eat and)* und Gleichung 4.3 zeigt die Berechnung von *(and eat)*.
Die Wahrscheinlichkeit wird berechnet mit:

$$[1]P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \tag{4.1}$$

$$P(\text{and}|\text{eat}) = \frac{C(\text{eat and})}{C(\text{eat})} \tag{4.2}$$

$$P(\text{and}|\text{eat}) = \frac{35}{2105}$$

$$P(\text{eat}|\text{and}) = \frac{C(\text{and eat})}{C(\text{and})} \tag{4.3}$$

$$P(\text{eat}|\text{and}) = \frac{35}{413767}$$

Gleichung 4.4 und Gleichung 4.5 zeigen die Berechnung mit dem hinzugefügten Wort *in*.

$$P(\text{in}|\text{eat}) = \frac{C(\text{eat in})}{C(\text{eat})} \tag{4.4}$$

$$P(\text{in}|\text{eat}) = \frac{13}{2105}$$

$$P(\text{in}|\text{and}) = \frac{C(\text{and in})}{C(\text{and})} \tag{4.5}$$

$$P(\text{in}|\text{and}) = \frac{522}{413767}$$

Die Ergebnisse von Gleichung 4.2 und Gleichung 4.3 werden multipliziert und ergeben $1,406462438_{x10}-6$, sowie die Ergebnisse von Gleichung 4.4 und Gleichung 4.5 ergeben $7,791227839_{x10}-6$. Die Wahrscheinlichkeit von *eat in and* ist demnach größer.

Den gleichen Satz *I eat and sleep* korrigiert das Tri-Gramm und das Tri-Gramm mit Smoothing zu *I eat and sleep*. Hier existieren keine Wahrscheinlichkeiten zu *(I eat in)* und *(eat in and)* sowie *(I eat and)* aber es existiert eine Wahrscheinlichkeit für *(eat and sleep)*. Diese Wahrscheinlichkeit ist zwar 0,02857 aber addiert mit minus Unendlich für *(I eat and)* wird sie zu minus Unendlich. Minus Unendlich als Exponenten der Exponentialfunktion lässt das Ergebnis Null werden. Es existieren auch keine anderen kleinen Wörter, die die Bedingung eines Folgewortes von *(I eat)* erfüllen.

Die Testsätze, die mit *My* anfangen (siehe Anhang Abschnitt A.2), sind vom Bi-Gramm zu *In My* verbessert worden. Der Grund ist, dass zunächst *(<s> My)* seltener vorkommt (62 Mal) als *(<s> In)* mit 60944 Mal. Außerdem kommt *(My nose)* in den Trainingsdaten gar nicht vor und *(My name)* nur einmal, wohingegen *(In My)* 30 Mal in den Trainingsdaten auftritt. In den meisten Fällen kommt die Kombination von *In My* in Titeln von Liedern oder Filmen vor. So ist die Wahrscheinlichkeit für *(<s> In My)* höher. Das Kleinschreiben von *My* ändert das Ergebnis

[1] Tatsächlich werden die Wahrscheinlichkeiten als logarithmische Zahl gespeichert, um zu viele Nullen hinter dem Komma zu vermeiden. $\log_{10}\left(\frac{C(w_{n-1}w_n)}{C(w_{n-1})}\right)$
Dies beeinflusst das Ergebnis jedoch nicht.

nicht. Bei den Tri-Grammen wird kein *In* hinzugefügt, da es kein vorheriges Wort für *(My nose)* oder *(My name)* gibt, das mindestens eins der gewünschten POS-Tags besitzt. Im Corpus kam also die Kombination *(In My name)* nie vor.

Unter den als *falsch korrigiert* gekennzeichneten Sätzen sind auch Sätze, die in bestimmten Zusammenhängen als *richtig korrigiert* gewertet werden könnten. Beispielsweise *I like August* und *I work like bee*, wenn man davon ausgehen kann, dass *August* und *bee* Eigennamen, Spitzname oder Synonyme für jemanden oder etwas sind. Außerdem wäre es möglich den Satz *but that my brother hates it* (korrigiert vom *Tri-Gramm mit Smoothing*) als richtig zu werten, wenn der Satz weitergeführt werden würde, wie z.B. *but that my brother hates it, is not supportable.* Ein anderes Beispiel wäre: *caterpillar is and duck is nice* (korrigiert vom Bi-Gramm). Grammatikalisch ist der Satz richtig, allerdings würde man eher *caterpillar and duck are nice* sagen.
Alle genannten Sätze sollten und wurden unabhängig von einem Zusammenhang bewertet, denn diese Satzverbesserung kennt keinen semantischen Zusammenhang.

4.2.1.2. Verben konjugieren

	Bi-Gramm	Tri-Gramm	Tri-Gramm (add-k-smoothing)
Anzahl vorhandene Verben	144	144	144
davon richtig korrigiert [2]	128	132	132
davon falsch korrigiert	16	12	12
Anzahl hinzugefügte Verben	37	0	5
davon richtig konjugiert [2]	37	0	2
davon falsch konjugiert	0	0	3

Tabelle 4.2.: Statistik der korrigierten Verben aus allen Testsätzen

Die Tabelle 4.2 zeigt die Statistik zu dem zweiten Teil der Satzverbesserung: Die Verbkonjugation. Die Tabelle wurde in den Teil *Konjugation bestehender Verben im Satz* und den Teil der *Konjugation der hinzugefügten Verben im Satz* aufgeteilt. Ein Verb gilt als richtig korrigiert, wenn es dem Nomen oder Pronomen angepasst wurde. Die Tabelle trifft keine Aussage zu der Richtigkeit des gesamten Satzes. Somit gilt *I like that nobody is but that you are because they are mean* als richtig korrigierte Verben. Das Verb in blau existierte bereits im Satz, die Verben

[2] Der Satz kann insgesamt falsch korrigiert worden sein, aber das Verb oder die Verben bzw. Nomen wurden richtig korrigiert.

in orange kamen hinzu. Allgemein wurden die meisten Verben richtig konjugiert. Tri-Gramm und Tri-Gramm mit Smoothing haben diese Verbesserung etwas besser meistern können, was auf die Größe des Kontextes zurückzuführen ist. Bi-Gramme kennen nur das Wort unmittelbar vor dem zu korrigierenden Verb. Steht ein Wort mit einer anderen Wortart als Nomen und Pronomen vor dem Verb, wie in *... sister also like ...*, so kann das Verb nicht mehr korrigiert werden. Dennoch gab es auch bei den Tri-Grammen Probleme. Bezieht sich ein Verb auf ein Nomen, dass viel früher im Satz vorkam, aber ein weiteres Nomen später folgt, wird das Verb anhand des zweiten Nomens konjugiert. *I eat sausage and drinks apple juice* und *Today my Mum and I go pet shop and buys new puppy* sind zwei Beispiele, wo sich die Verben *drink* und *buy* eigentlich auf das Pronomen *I* bzw. *Mom and I* beziehen, aber die Nomen *sausage* und *shop* später im Satz auftreten.

Ein weiteres Beispiel der Schwierigkeit bei Bi-Grammen ist: *Sweets and yoghurt is delicious.* Zum einen wurde das Verb *is* hinzugefügt und hinzugefügte Verben werden aus Laufzeitgründen nicht im Nachhinein konjugiert, zum anderen wäre es dennoch der Fall, so würde ein Bi-Gramm nur den Kontext *yoghurt* kennen und *is* wäre dann korrekt. Der größere Kontext beim Tri-Gramm hilft in diesem Fall nicht unbedingt weiter, da dieser nur *and yoghurt* umfasst. Da *is* jedoch ein hinzugefügtes Wort ist, hätte es dennoch wie bei *Hello my name Louisa and I are a four years* korrigiert werden können. Dieser Satz wurde in dieser Form vom Tri-Gramm mit Smoothing korrigiert. Das Verb *are* wurde hinzugefügt, was in dem vorher beschriebenen Satz korrekt gewesen wäre, hier jedoch falsch ist. Wir als Menschen erkennen, dass das *and* in

Sweets and yoghurt is delicious (4.6)

und

Hello my name Louisa and I are a four years (4.7)

unterschiedliche Bedeutungen haben - eine Maschine nicht. Bei 4.6 verbindet das *and* zwei oder mehr Nomen oder Pronomen zu einem Ganzen. Bei mehreren Nomen werden die ersten häufig durch ein Komma getrennt z.B. *apple, banana and orange.* Bei 4.7 verbindet das *and* zwei Hauptsätze miteinander. Diese Semantik ist dem Bi-Gramm, Tri-Gramm und Tri-Gramm mit Smoothing Verfahren nicht bekannt und ist einem 4-Gramm oder sogar 5-Gramm auch nicht bekannt. Es wird jedoch nicht ausgeschlossen, dass ein 4-Gramm oder 5-Gramm die Verben in den Sätze richtig korrigiert hätten.

Bei Tri-Gramm und Tri-Gramm mit Smoothing wurden zwölf Verben falsch korrigiert. Jeweils zwei davon sind das Verb *running*. Dieses Verb alleinig ist als falsch bewertet worden, da es nur in *Partizip Präsens* verbunden mit den Progressive Zeitformen *is* richtig ist.

4.2.1.3. Nomen deklinieren

	Bi-Gramm	Tri-Gramm	Tri-Gramm (add-k-smoothing)
Anzahl Grundzahlen [3] & Demonstrativbegleiter	7	7	7
davon darauffolgende Nomen richtig korrigiert [2]	6	1	7
davon darauffolgende Nomen falsch korrigiert	1	6	0

Tabelle 4.3.: Anzahl der korrigierten Nomen, die auf einer Grundzahl oder einen Demonstrativbegleiter folgen, aus allen Testsätzen

Als einen weiteren Teil der Satzverbesserung wurde *Nomen deklinieren* überprüft und diese Ergebnisse sind in Tabelle 4.3 dokumentiert. Wie in Kapitel 3 erklärt, werden Nomen anhand von bestimmten Zahlen und Demonstrativbegleitern dekliniert. Die Sätze wurden auf diese untersucht und gezählt (siehe Zeile *Anzahl Grundzahlen & Demonstrativbegleiter* in Tabelle 4.3). Grundzahlen als Ziffern werden vor der Korrektur in ausgeschriebene *Grundzahlen* von eins bis 100 umgewandelt, damit keine Ziffern berücksichtigt werden müssen.

Das Tri-Gramm hat bei diesem Vorgehen gegenüber Bi-Grammen und Tri-Grammen mit Smoothing schlecht abgeschnitten und nur eins von sieben Nomen richtig dekliniert. Bei den sechs falsch deklinierten Nomen gab es keine Wahrscheinlichkeiten für die eigentlich richtig deklinierten Nomen. In den meisten Fällen war der Aufbau *…I <Zahl> year </s>*. Bei dem Beispiel *…I four year </s>* gibt es keine Wahrscheinlichkeiten in der Datenbank für (I four years) oder (four years </s>). Der Satz *…I am four year </s>* würde ebenfalls nicht weiter helfen, da keine Wahrscheinlichkeit für (am four years) existiert. Jedoch sind Wahrscheinlichkeiten für (is four years), (four years old) oder (four years and) in der Datenbank vorhanden. Würde der Satz *He is five year* heißen, so wäre das Nomen richtig dekliniert worden. Im Gegensatz zu den reinen Tri-Grammen haben Tri-Gramme mit add-k-smoothing eine geringe Wahrscheinlichkeit für das korrigierte Nomen, weil zu den ursprünglichen Wahrscheinlichkeiten k mit 0,5 addiert wird.

Das Bi-Gramm hat den Satz *I like two onion therefore I eat onion* nicht richtig korrigiert. Bei der Konzeption des Algorithmus hat sich die Autorin dafür entschieden, das korrigierte

[3] Grundzahlen sind z.B. *eins*, *zwei*, *fünfzig* …

Wort mit der höchsten Wahrscheinlichkeit aus den drei Verfahren zu wählen. Der obige Satz wurde falsch korrigiert, da alle drei Wahrscheinlichkeiten trotz Verbesserung minus Unendlich ergaben. Das Nomen wurde eigentlich korrekt zu *onions* verbessert, doch das Bi-Gramm *(two onions)* war in den Trainingsdaten nicht vorhanden.

Der Algorithmus setzt die Wahrscheinlichkeit eines Wortes auf minus Unendlich, wenn es die Bedingung für diese Korrektur nicht erfüllt. Ein Nomen beispielsweise wird nicht wie ein Verb korrigiert. Das Ergebnis der Funktion der *Verbkorrektur* ist demnach minus Unendlich.

4.2.1.4. Zusammenfassung

Bi-Gramm	
Satz	29,9 %
Verben	88,9 %
Verben neu hinzugefügt	100,0 %
Nomen	85,7 %
Tri-Gramm	
Satz	36,5 %
Verben	91,7 %
Verben neu hinzugefügt	- %
Nomen	14,3 %
Tri-Gramm (add-k-smoothing)	
Satz	35,0 %
Verben	91,7 %
Verben neu hinzugefügt	40,0 %
Nomen	100,0 %

Tabelle 4.4.: Prozentuale Ansicht der richtigen Ergebnisse der verschiedenen Vorgehen

Die Tabelle 4.4 zeigt die Ergebnisse der einzelnen Verfahren in Prozent an. Diese beschreiben nur die richtig verbesserten Wörter bzw. Sätze. Innerhalb eines Verfahrens können die N-Gramme miteinander verglichen werden. Eine Ausnahme sind allerdings die hinzugefügten Verben. Bei jedem N-Gramm werden zum Teil an unterschiedlichen Stellen Verben hinzugefügt. Bei Bi-Grammen sind es zum Beispiel 37 Wörter und bei Tri-Grammen werden keine hinzugefügt. Da es sich nicht um die gleiche Grundmenge handelt können die einzelnen Verben nicht miteinander verglichen werden.

Aus Tabelle 4.4 kann entnommen werden, dass das Tri-Gramm das beste getestete N-Gramm ist, denn 36,5% der 137 Sätze wurden richtig verbessert. Beim Bi-Gramm wurden nur 29,9% aller

Sätze richtig korrigiert, wohingegen Verben zu circa 89% und *neu hinzugefügte Verben* sogar zu 100% richtig korrigiert worden. Die *Korrektur der Nomen* wurde ebenfalls zu 85,7% richtig durchgeführt. Daraus lässt sich schließen, dass am häufigsten *neu hinzugefügte Wörter* die Sätze negativ beeinflussen. Dies bestätigt sich beim Betrachten der einzelnen Sätze. Das Verfahren *Verben hinzufügen* bei Tri-Grammen hat kein Verb in die Sätze hinzugefügt und wurde deshalb mit Bindestrich (-) gekennzeichnet. Tri-Gramm mit add-k-smoothing war insgesamt nicht besser als die reine Tri-Gramm Satzverbesserung. Neue Wörter wurden hinzugefügt, aber besonders die neu hinzugefügten Verben passten nur zu 40% zu ihren Nomen oder Pronomen. Demnach wurden Verben eingefügt, bei denen es eigentlich berechtigt ist, dass sie beim reinen Tri-Gramm nicht im Satz vorkommen. Den einzigen Vorteil hatten die Tri-Gramme mit add-k-smoothing bei der *Deklination von Nomen*. Zu 100% wurden die Nomen nach einer Grundzahl bzw. Demonstrativbegleiter richtig angepasst.

4.2.2. Benutzbarkeit Benutzeroberfläche

Ein wichtiger Aspekt ist die Benutzbarkeit der Anwendung, da sie z.B. entscheidet, wie lange die Eingabe eines Satzes dauert. Einige Bemerkungen kommen von den Probanden selbst, andere resultieren aus der Beobachtung während ein Proband das System benutzt hat.

Es wurde bemängelt, dass Wörter fehlen, allerdings ist die Anwendung ein Prototyp und wird nicht alle möglichen Wörter umfassen. Oxford University Press (2017) schätzt die Anzahl, der heutzutage benutzen Wörter auf 171,476. Zudem unterscheidet sich der Wortschatz von Person zu Person. Eine bestimmte Menge an Wörtern aus einer Sprache benutzt jede Person. Alle Wörter, die nicht in dieser Menge vorkommen, werden von der einen Person häufig, von der anderen Person wenig oder sogar gar nicht verwendet. Auf der Benutzeroberfläche ist es jedoch möglich einzelne Buchstaben einzugeben und so ein Wort zu tippen. Allerdings kann es dazu führen, dass das System dem Wort ein unerwünschtes POS-Tag gibt. Beispielsweise wurde während des Testens das Wort *cat* eingegeben. Im System war das POS-Tag (zu dem Zeitpunkt) nicht festgelegt und somit wurde das am meisten vorkommende POS-Tag aus der Datenbank genommen. In diesem Fall war es ein Verb: *to cat* (deutsch: kotzen). Es führte dazu, dass es wie ein Verb behandelt und ein *-s* angefügt wurde.

Einige Probanden haben erwartet, dass entweder bei der Verbesserung der gesamte Satz umgestellt wird oder einige Wörter umgewandelt werden. Es wurde z.B. erwartet, dass aus *He dog like*, *He likes dog* wird oder aus *I name is…*, *My name is…* wird. Dies war allerdings nicht vorgesehen.

Das Eingabefeld ist auf die Länge des sichtbaren Feldes begrenzt. Dieses kann unter Umständen nachteilig sein, zum Beispiel wenn besonders lange Wörter in einem Satz vorkommen oder der Satz aus vielen Wörtern besteht. Die Sätze sollten dennoch auf eine bestimmte Länge begrenzt werden, damit die Verbesserung nicht zu lange dauert.

Besonders wichtig bei der Benutzbarkeit ist die Anordnung der Symbole. Einige Wörter wurden bereits in Kategorien zusammengefasst, sodass ein schnelles Finden möglich ist. Die Schwierigkeit ist jedoch, die Wörter in die richtige Kategorie hinzuzufügen, aber auch innerhalb der Kategorie richtig anzuordnen. Eine mögliche Lösung für Kategorien mit vielen Wörtern wäre, häufig verwendete Wörter weiter oben anzuordnen. Eine weitere Lösung wäre, die Wörter je nach Benutzung dynamisch anzuordnen, sodass es individuell auf das Kind abgestimmt ist. Zudem könnten alle Wörter verschiebbar sein, damit das Kind im Nachhinein noch entscheiden kann in welche Kategorie ein Wort gehört. Dies könnte entweder auf dem Tablet selber geschehen oder in einer Software am PC.

Einige Sätze brauchen länger als andere, um korrigiert zu werden. Dies muss nicht unbedingt in Verbindung mit der Länge des Satzes stehen. Beim Testen ist aufgefallen, dass die Probanden unruhig wurden, wenn die Satzverbesserung zu lange dauerte und es kein optisches Feedback der Anwendung gab. Eine Verbesserung wäre also ein Wartebalken oder ein textuelles Feedback dem Benutzer zu geben, wie beispielsweise ein drehendes *Wartesymbol*.

Wird der Satz wiederholt ausgesprochen, kommt es zu Überschneidungen wie bei einem Kanon. Die Sequenz startet von vorne, während eine andere noch läuft. Der erste Lösungsansatz wäre jedes weitere Starten der Sequenz zu ignorieren und den ersten durchlaufen zu lassen. Ein anderer intuitiverer Lösungsansatz wäre, die erste Sequenz zu stoppen und die neue von Vorne beginnen zu lassen. Das Kind möchte eventuell die Sequenz von Neuem starten, wenn der Gesprächspartner den Anfang nicht verstanden oder mitbekommen hat. Dies ist in einem normalen Gespräch ebenfalls der Fall. Ein neuer Satz sollte jedoch immer erst nach dem vorherigen ausgesprochen werden.

4.2.3. Allgemeine Verbesserungsmöglichkeiten

Die Wahl des Corpus spielt eine wichtige Rolle, da dadurch die N-Gramm Wahrscheinlichkeiten verändernt werden können. Cavnar und Trenkle (1994) beschreiben, dass eine bestimmte Wortmenge in fast jedem Text einer Sprache vorkommt. Andere Wörter sind themenspezifisch oder hängen vom Autor ab. Unterschiedliche Text-Typen können die Ergebnisse verändern. Wissenschaftliche Arbeiten sind nicht in der Ich-Perspektive geschrieben und besitzen selten das Wort *I* in der englischen Sprache; ausgenommen in Zitaten. Würden wissenschaftliche Arbeiten, als Grundlage für das Corpus in dieser Anwendung genommen werden, gäbe es

geringe Wahrscheinlichkeiten zu Sätzen mit *I*. Aber auch der gewählte Corpus (Simple English Wikipedia) ist selten in Ich-Perspektive geschrieben. Es gibt allerdings mehr Zitate und Konversationen mit Ich-Perspektive als in wissenschaftlichen Arbeiten. Ein Beispiel aus *Simple English Wikipedia Nathan Hale* wäre *He is probably best remembered for his last words before being hanged: "I only regret that I have but one life to give for my country.".*

Außerdem könnte eine Bearbeitung des Corpus sinnvoll sein. Zitate, die im Corpus vorkommen sind mit Anführungsstrichen gekennzeichnet und meistens steht vor einem Zitat noch ein Ausdruck. Fängt ein Testsatz mit *My* an, so würde ein Zitat, das auch mit *My* anfängt nicht als Count zählen, da das Wort vorher kein <*s*> ist. Zitate, die einen vollständigen Satz bilden, sollten als ein Satz gekennzeichnet werden, da sie für diese Anwendung nützlich wären. Nur eine Eingabe von einfachen Sätzen ohne Anführungsstriche und Doppelpunkt sind in der Anwendung möglich. Komplizierte Sätze in den Trainingsdaten würden das Ergebnis verfälschen.

Es ist sinnvoll, ein Corpus zu wählen, das bereits mit POS-Tags versehen ist, damit ein Wort mit mehreren Wortarten im N-Gramm unterschieden werden kann. In der Applikation kam es zwischenzeitig zu Problemen mit dem Wort *I*. Es folgte bei der Verbesserung immer wieder das Wort *of*. Der Grund ist, dass im Corpus das Wort *I* häufig die Bedeutung von einer Zahl besaß. Im Folgenden sind Beispiele aus dem Corpus, in dem *I* eine Zahl ist:

> ...remember about World War I and the...
> ...Kamehameha I was the first king of Hawaii.
> ...Italy participated in World War I as an ally of Great Britain,...
> ...found in the Slovenian cave Divje Babe I in 1995.
> In 1561 Felipe I moved the royal court ...
> Their grandson was the famous Charles I of Spain ...
> ...Pope John Paul I (; , 17 October 1912 ? 28 September 1978), born Albino Luciani...
> ...as the capital city by King Rama I of Chakri Dynasty...
> ...World War I when the federal government...
> ...Haile Selassie I of Ethiopia...
> ...original Apple I computer.

Einige der Testsätze sind ähnlich aufgebaut. So fangen beispielsweise 13 Sätze mit *My* an und fünf davon mit *My name*. Wenn ein Bi-Gramm oder ein Tri-Gramm eine Wortfolge eines Satzes falsch verbessert hat, dann wird es die gleiche Wortfolge eines ähnlichen Satzes ebenfalls falsch verbessern. Dies führt zu einer Abweichung der Statistiken. Eine Lösung wäre zusätzlich zu den Testsätzen der Probanden andere Sätze hinzuzunehmen, um eine größere Variation

an Sätzen zu bekommen. Jedoch war es der Autorin wichtig ein möglichst reales Szenario nachzubilden und entschied sich daher nur die Testsätze der Probanden auszuwerten.

Bei der Satzverbesserung muss beachtet werden, dass je besser und präziser die Korrektur ist, die Laufzeit umso länger ist. Es wäre z.B. möglich einen Satz auf verschiedene Arten zu verbessern und zum Schluss zu vergleichen. Für diese Anwendung macht diese Option dennoch wenig Sinn, da das Generieren des Satzes und das Vergleichen zu lange dauern würde.

Eine Voraussetzung ,um diese Anwendung benutzen zu können ist eine Verbindung zum Internet. Das Frontend spricht mit dem Backend über das Internet. Ist dieses nicht verfügbar, kann kein Satz verbessert werden. In diesem Fall kann die Anwendung nur zum Schreiben verwendet werden, aber weder zum Verbessern noch zum Aussprechen. Die gesamte Satzverbesserung auf ein mobiles Gerät ohne Internetverbindung laufen zu lassen, stellt eine weitere Herausforderung dar.

4.3. Anforderungen evaluieren

In diesem Abschnitt wird evaluiert, welche Anforderungen aus den funktionalen Anforderungen 3.2.1 und den nichtfunktionalen Anforderungen 3.2.2 erfüllt oder nicht erfüllt wurden und welche verbessert werden könnten. Von den fünf funktionalen Anforderungen wurde Punkt 2, Punkt 4 und Punkt 5 erfüllt. Punkt 1, einen grammatikalisch falschen Satz entgegennehmen und einen grammatikalisch richtigen Satz zurückliefern, erfüllt die Satzverbesserung nur mit dem Entgegennehmen eines Satzes. Einen grammatikalisch richtigen Satz zurückliefern, das Punkt 3 ähnelt, kann die Satzverbesserung nicht vollständig leisten. Der Satz *she like you* wird mit Hilfe von Tri-Grammen richtig korrigiert zu *She likes you* aber falsch korrigiert mit Bi-Grammen zu *she likes to you.* Anders herum wird der Satz *he fever* richtig korrigiert zu *he has fever* mit Bi-Grammen und falsch korrigiert mit Tri-Grammen *He fever.* Dieser Unterschied zwischen Bi-Gramm und Tri-Gramm ist begründet durch die Größe der jeweiligen Kontexte.

Aus den nichtfunktionalen Anforderungen wurden alle Punkte erfüllt. Das Frontend läuft auf einem Android Gerät mit der minimalen Version 4.1.1. Es konnte ebenfalls auf einem Android Gerät mit Version 6.0.1 ohne Probleme getestet werden. Das System bzw. das Backend ist unabhängig vom Frontend und stellt Schnittstellen bereit, um andere UIs von anderen Betriebssystemen anbinden zu können. Zusätzlich lässt sich das Backend auf anderen Betriebssystemen starten. Hierfür wurde Docker[4] als Virtualisierungssoftware verwendet. Auch die verwendete

[4]Docker ist eine Virtualisierung um Anwendungen in Containern zu isolieren. Siehe https://www.docker.com für weitere Informationen (abgerufen am 20.08.2017).

Redis-Datenbank, die alle wichtigen Daten beinhaltet, wird in Docker ausgeführt. Für das Python Backend sowie für die Redis-Datenbank wurde je ein Dockerfile[5] zur Verwendung eines Basis-Images erstellt. Zudem wurde eine Docker Compose[6] Datei angelegt, damit mit einem Befehl beide Container mit ihren Konfigurationen gestartet werden können. Somit ist das Backend unabhängig vom Betriebssystem und kann auf jedem Rechner, auf dem Docker installiert ist, ausgeführt werden. Die Evaluation von *Das System ist leicht bedienbar* wurde daran festgemacht, wie gut eine Person, insbesondere ein Kind, lernt mit dem System umzugehen. Die Testpersonen hatten relativ schnell den Unterschied zwischen *Wort* und *Kategorie* herausgefunden. Die Probanden haben auch die Symbole zum Abschicken des Satzes zur Satzverbesserung und das Wiederholen der Aussprache des Satzes richtig verwendet. Die meisten Testsätze waren mit einer Antwortzeit unter einer Sekunde aber maximal zwei Sekunden schneller als die nichtfunktionale Anforderung gefordert hat. Dazu muss erwähnt werden, dass die meisten Testsätze nicht mehr als 12 Wörter, durchschnittlich aber weniger, beinhalteten.

Im Großen und Ganzen konnte die Mehrzahl der Anforderungen erfüllt werden. Die einzige, aber größte Anforderung, den Satz richtig korrigieren zu lassen konnte nicht wie gewünscht erreicht werden. Hierzu bedarf es weiterer Forschung für einen optimaleren Algorithmus bzw. die Verwendung von anderen Methoden. Es könnte getestet werden, ob andere Smoothing Verfahren ein besseres Ergebnis liefern oder ob die Hilfe einer KfG zu besseren Ergebnissen führen würde. Außerdem wäre ein Vergleich verschiedener Corpora und deren Auswirkungen auf die Satzverbesserung interessant.

[5] Dockerfile ist ein Textdokument, das alle Kommandos beinhaltet, die zum Bau eines Images benötigt werden.
[6] Docker Compose ist ein Tool zum Beschreiben und anschließenden Starten einer *multi-container* Docker Anwendung

5. Fazit und Ausblick

In diesem Kapitel wird zunächst ein Fazit der Ergebnisse gezogen und anschließend wird auf mögliche Verbesserungen und Erweiterungen der Anwendung in der Zukunft eingegangen.

Kinder haben in den meisten Fällen Interesse an elektronischen Geräten und das Ziel dieses Interesse bei den Kindern mit *Talk To Me* zu nutzen, war erfolgreich. Die jüngsten Probanden hatten Spaß die Anwendung zu benutzen. Die Jüngste meinte, es sei lustig damit zu schreiben, da es wie ein Spiel sei. Die Satzverbesserung war nicht in allen Zügen positiv ausgefallen. Nomen deklinieren und Verben konjugieren verlief häufig korrekt, aber fehlende Wörter einfügen verschlechterte im Regelfall die Grammatik des Satzes. Die Verwendung von Tri-Grammen, im Gegensatz zu Bi-Grammen, fügte weniger willkürliche Wörter ein, jedoch wurde an wichtigen Stellen, zu wenig korrigiert. *Add-k-Smoothing* verbesserte das Ergebnis ebenfalls nicht. Es wurden zwar mehr Wörter an richtiger Stelle eingefügt, doch teilweise waren diese Verben falsch konjugiert. In einzelnen Fällen kam es zu Veränderungen des Sinns eines Satzes.

Eine Verbesserung könnte in Zukunft mit der Hinzunahme einer kontextfreien Grammatik erzielt werden. Mit Hilfe dieser und den N-Grammen könnte genauer bestimmt werden, ob ein Wort und welches Wort an die gegebene Stelle hinzugefügt werden sollte. Außerdem könnte es möglich sein, mit einer Grammatik eine falsche Satzstellung zu verbessern. So könnten einfache Sätze wie *I dog like* verbessert werden zu *I like the dog*. Weiterhin sollte überprüft werden, inwieweit das Resultat verändert wird, wenn ein anderes Corpus, beispielsweise eine Auswahl aus Kinderbüchern, gewählt wird.

In Hinblick auf die Zukunft der Oberfläche sollte die Anwendung individualisierbar sein und z.B. die Funktion eigene Symbole und Bilder hinzuzufügen, ermöglichen. So könnten Fotografien von eigenen Haustieren oder vom Lieblingskuscheltier in der Anwendung bereitgestellt werden. Zudem wird die Anwendung vereinfacht, wenn *zuletzt verwendete Wörter* oder *häufig verwendete Wörter* in der obersten Zeile angezeigt werden würden. Dazu müsste eine individuelle Statistik geführt werden, die bereits verwendete Wörter speichert. Diese Statistik ist auch für die allgemeine Anordnung der Symbole interessant.

A. Anhang

A.1. Tagset der POS

Tag	Beschreibung
$	currency symbol ($)
”	closing quotes (")
(	opening parentheses (()
)	closing parentheses ())
,	comma (,)
:	connecting punctuation (:, -, ...)
CC	conjunction, coordinating (and)
CD	number, cardinal (four)
DT	determiner, general (a, the, this, that)
EX	existential there
FW	foreign word (ante, de)
IN	preposition (on, of)
JJ	adjective, general (near)
JJR	adjective, comparative (nearer)
JJS	adjective, superlative (nearest)
LS	List item marker
MD	modal auxiliary (might, will)
NN	noun, common singular (action)
NNS	noun, common plural (actions)
NP	noun, proper singular (Thailand, Thatcher)
NPS	noun, proper plural (Americas, Atwells)
PDT	determiner, pre- (all, both, half)
POS	possessive particle (’, ’s)
PP	pronoun, personal (I, he)
PP$	pronoun, possessive (my, his)

RB	adverb, general (chronically, deep)
RBR	adverb, comparative (easier, sooner)
RBS	adverb, superlative (easiest, soonest)
RP	adverbial particle (back, up)
SENT	sentence-final punctuation (.)
SYM	symbol or formula (US$500, R300)
TO	infinitive marker (to)
UH	interjection (aah, oh, yes, no)
VB	be, base (be)
VBD	be, past tense (went)
VBG	be, -ing (being)
VBN	be, past participle (been)
VBP	be, plural (are)
VBZ	be, -s (is)
VH	have, base (have)
VHD	have, past tense (had)
VHG	have, -ing (having)
VHN	have, past participle (had)
VHP	have, plural (have)
VHZ	verb, -s (believes)
VV	verb, base (believe)
VVD	verb, past tense (believed)
VVG	verb, -ing (believing)
VVN	verb, past participle (believed)
VVP	verb, plural (believe)
~~VBZ~~ VVZ	verb, -s (believes)
WDT	det, wh- (what, which, whatever, whichever)
WP	pronoun, wh- (who, that)
WP$	pronoun, possessive wh- (whose)
WRB	adv, wh- (how, when, where, why)
“	opening quotes (")

Quelle: Evert u. a. (2016)
Das durchgestrichene Tag ist ein Tippfehler der Autoren und kommt doppelt vor. Es wurde mit dem nachstehenden Tag verbessert

A.2. Testsätze

Auf den folgenden Seiten befinden sich alle Testsätze, die von den Probanden in die mobile Anwendung eingegeben wurden. Alle bestehenden Verben sind gelb markiert. Verben, die hinzugekommen sind, sind rot markiert. Wörter, anhand denen Nomen dekliniert wurden, sind grün markiert. Sätze, die einen Haken haben, wurden insgesamt als richtig verbessert bewertet.

In der ersten Spalte ist der ursprünglich eingegebene Satz, in der zweiten Spalte der korrigierte Satz und in der dritten Spalte ist das jeweilige Vorgehen (*Bi-Gramm* , *Tri-Gramm* oder *Tri-Gramm mit add-k-Smoothing*) angegeben.

boy love girl bit	The Boy in love the girl in bit	bigram
but my brother hate it	But in my brother of hate because it	bigram
cat in house	The Cat in house	bigram
caterpillar and duck nice	Caterpillar is and duck is nice	bigram
dad like sun	✓ Dad likes the sun	bigram
dad love love love love love love love love pick nose	Dad for love in love in love in love in love in love in love to pick the nose	bigram
Dad play guitar	✓ Dad plays the guitar	bigram
do you want eat or drink	Do with you want to eat the or a drink	bigram
grandma cook vegetables today	Grandma cooks in vegetables today	bigram
great you nice I love you and you body and meat	The Great of you are nice because I love with you are and that you are body of and meat	bigram
he do his homework	He does in his homework	bigram
he five years old	✓ He is five years old	bigram
he like you	✓ He likes to you	bigram
he like your cat and dog	He likes to your cat in and dog	bigram
he love cheese	✓ He loves the cheese	bigram
he love you	He loves to you	bigram
he love your bottom	He loves of your bottom	bigram
he play guitar	✓ He plays the guitar	bigram
he prince I princess	He prince of I am princess	bigram
Hello my name Louisa	A Hello my name of Louisa	bigram
Hello my name Louisa and I four year	A Hello my name of Louisa in and the I am four years	bigram
Hello please help me	A Hello please help to me	bigram
I 23 year	✓ I am twenty-three years	bigram
I afraid bee	I am afraid bee	bigram
I afraid vacuum	I am afraid vacuum	bigram
I also like play with my doll	I am also the like to play with all my doll	bigram
I angry and evil	I am angry and the evil	bigram
I artist	I artist	bigram
I bird	I am bird	bigram
I cold	✓ I am cold	bigram
I do not know	I do not to know	bigram
I do not know more share	I do not to know the more to share	bigram
I do not know you	I do not to know that you	bigram
I do not want brush teeth	✓ I do not want to brush the teeth	bigram
I drink lemonade	I drink of lemonade	bigram
I eat and sleep	I eat in and to sleep	bigram
I eat sausage and drink apple juice	I eat the sausage in and the drink the apple juice	bigram
I hate cucumber and peanut but I love melon	I hate a cucumber and peanut is but the I love with melon	bigram
I hate doctor	✓ I hate a doctor	bigram
I hate pirate	✓ I hate a pirate	bigram
I hope I play after I finish work	I hope that I play after the I finish the work	bigram
I like artist	✓ I like the artist	bigram
I like August	I like in August	bigram
I like baby	✓ I like a baby	bigram
I like cookie and sweets and lollipop	I like a cookie is and the sweets and lollipop	bigram
I like dog	✓ I like the dog	bigram
I like dog and cat but I afraid spider	I like the dog in and cat is but the I am afraid spider	bigram
I like have apple	✓ I like to have the apple	bigram
I like my brother	I like to my brother	bigram
I like my sister	I like to my sister	bigram
I like nobody but you because they mean	I like that nobody is but that you are because they are mean	bigram
I like phone	✓ I like the phone	bigram
I like phone and tablet and printer	I like the phone to and tablet of and printer	bigram
I like play doll but my brother annoying	I like to play the doll is but in my brother of annoying	bigram
I like swing and slide	I like a swing to and slide	bigram
I like tea please	I like the tea to please	bigram
I like television	✓ I like the television	bigram
I like two onion therefore I eat onion	I like the two onion therefore the I eat the onion	bigram
I like you	I like that you	bigram
I like your beard	I like to your beard	bigram
I like your tshirt	I like to your tshirt	bigram
I love apple juice	I love of apple juice	bigram
I love christmas livingroom	I love christmas livingroom	bigram
I love horse	I love of horse	bigram
I love horse and bunny and dog and cat	I love of horse in and bunny and dog in and cat	bigram
I love pick nose	✓ I love to pick the nose	bigram
I love play guitar	✓ I love to play the guitar	bigram
I love princess	✓ I love the princess	bigram
I love red and peach	I love with red and peach	bigram
I love tablet phone cd dvd keyboard television film camera	I love the tablet phone cd dvd keyboard television film camera	bigram
I love you	I love with you	bigram
I need bathroom	✓ I need a bathroom	bigram
I need toilet	✓ I need a toilet	bigram
I no like thunder	I am no like the thunder	bigram
I play accordion and trumpet and triangle	I play the accordion is and trumpet in and triangle	bigram
I play socker	✓ I play socker	bigram
I play socker and I love ride bicycle	I play socker and the I love to ride a bicycle	bigram
I play violin	✓ I play the violin	bigram
I really see big shark	I am really the see a big shark	bigram
I stomachache	I stomachache	bigram
I twenty three year	I am twenty in three years	bigram
I twenty year	✓ I am twenty years	bigram
I want more socks	I want to more socks	bigram
I want more socks my socks bad	I want to more socks with my socks as bad	bigram
I work like bee	I work to like a bee	bigram
ice creme delicious	Ice creme delicious	bigram
let us play with dolls	The Let us play with the dolls	bigram
mom	✓ Mom	bigram
mom and my sister and my grandma muscle	Mom in and in my sister of and in my grandma muscle	bigram

mom diarrhea	Mom diarrhea	bigram
Mom love cat	Mom in love of cat	bigram
Mom play guitar	Mom at play the guitar	bigram
mom work dressmaker and dad work filmmaker	Mom at work the dressmaker and dad for work of filmmaker	bigram
my brother like play with tablet	In My brother of like to play with tablet	bigram
my favorite color red	In My favorite color of red	bigram
my favorite color yellow	In My favorite color of yellow	bigram
my mum dressmaker	In My mum dressmaker	bigram
my name baby	In My name baby	bigram
my name celine	In My name celine	bigram
my name Cinderella	In My name Cinderella	bigram
my name Claire and I two year	In My name of Claire in and the I am two years	bigram
my name is celine	In My name is celine	bigram
my nose running	In My nose is running	bigram
my nose running I cold	In My nose is running the I am cold	bigram
my sister also like play with my doll	In My sister is also the like to play with all my doll	bigram
my socks bad	In My socks as bad	bigram
no Ouch! Ouch! Ouch!	✓ No Ouch! Ouch! Ouch!	bigram
Ouch!	✓ Ouch!	bigram
Ouch! I stomachache	Ouch! I stomachache	bigram
She do not want brush teeth	✓ She does not want to brush the teeth	bigram
she do washing	✓ She does the washing	bigram
she kind	✓ She is kind	bigram
she like Cinderella	She likes of Cinderella	bigram
she like monster	✓ She likes the monster	bigram
she like play ball	✓ She likes to play the ball	bigram
she need wash your breast	She needs to wash out your breast	bigram
she play guitar with Susi	✓ She plays the guitar with Susi	bigram
she pregnant	✓ She is pregnant	bigram
sweets and yoghurt delicious	Sweets and yoghurt is delicious	bigram
thank you	✓ Thank you	bigram
that scary	That scary	bigram
The cat in house	The cat in house	bigram
they eat pineapple and raspberry	They eat the pineapple to and raspberry	bigram
they love potty	They love for potty	bigram
today I wear blue jeans and yellow tshirt	The Today that I wear a blue jeans in and the yellow tshirt	bigram
Today my Mum and I go pet shop and buy new puppy	The Today in my Mum and the I go to pet shop in and to buy a new puppy	bigram
tomorrow I brush teeth and blow nose	The Tomorrow is I brush the teeth in and the blow the nose	bigram
we like brush teeth	We like a brush the teeth	bigram
what your name	✓ What is your name	bigram
when I ride my bicycle I wear bicycle helmet	When the I ride in my bicycle that I wear a bicycle helmet	bigram
when I ride my bicycle I wear helmet	When the I ride in my bicycle that I wear a helmet	bigram
yes I love your bottom	Beyond Yes I love with your bottom	bigram
you are welcome	✓ You are welcome	bigram
You eat sausage and drink apple juice	You eat the sausage in and the drink the apple juice	bigram
you hate banana	✓ You hate the banana	bigram
you like sausage	✓ You like a sausage	bigram
you love pick nose	✓ You love to pick the nose	bigram
boy love girl bit	Boy loves girl bit	trigram
but my brother hate it	✓ But my brother hates it	trigram
cat in house	The cat in house	trigram
caterpillar and duck nice	Caterpillar and duck nice	trigram
dad like sun	✓ Dad likes sun	trigram
dad love love love love love love love love pick nose	Dad loves love love love love love love love pick nose	trigram
Dad play guitar	✓ Dad plays guitar	trigram
do you want eat or drink	✓ Do you want to eat or drink	trigram
grandma cook vegetables today	✓ Grandma cooks vegetables today	trigram
great you nice I love you and you body and meat	Great you nice I love you and you body and meat	trigram
he do his homework	✓ He does his homework	trigram
he five years old	He five years old	trigram
he like you	✓ He likes you	trigram
he like your cat and dog	✓ He likes your cat and dog	trigram
he love cheese	✓ He loves cheese	trigram
he love you	✓ He loves you	trigram
he love your bottom	✓ He loves your bottom	trigram
he play guitar	✓ He plays guitar	trigram
he prince I princess	He prince I princess	trigram
Hello my name Louisa	Hello my name Louisa	trigram
Hello my name Louisa and I four year	Hello my name Louisa and I four year	trigram
Hello please help me	Hello please help me	trigram
I 23 year	I twenty-three year	trigram
I afraid bee	I afraid bee	trigram
I afraid vacuum	I afraid vacuum	trigram
I also like play with my doll	✓ I also like to play with my doll	trigram
I angry and evil	I angry and evil	trigram
I artist	I artist	trigram
I bird	I bird	trigram
I cold	I cold	trigram
I do not know	✓ I do not know	trigram
I do not know more share	I do not to know more share	trigram
I do not know you	✓ I do not know you	trigram
I do not want brush teeth	I do not want brush teeth	trigram
I drink lemonade	✓ I drink lemonade	trigram
I eat and sleep	✓ I eat and sleep	trigram
I eat sausage and drink apple juice	I eat sausage and drinks apple juice	trigram
I hate cucumber and peanut but I love melon	✓ I hate cucumber and peanut but I love melon	trigram
I hate doctor	I hate doctor	trigram
I hate pirate	I hate pirate	trigram

I hope I play after I finish work	I hope I play after I finish work	trigram
I like artist	I like artist	trigram
I like August	I like August	trigram
I like baby	I like baby	trigram
I like cookie and sweets and lollipop	I like cookie and sweets and lollipop	trigram
I like dog	I like dog	trigram
I like dog and cat but I afraid spider	I like the dog and cat but I afraid spider	trigram
I like have apple	I like have apple	trigram
I like my brother	I like my brother	trigram
I like my sister	I like my sister	trigram
I like nobody but you because they mean	I like nobody but you because they mean	trigram
I like phone	I like phone	trigram
I like phone and tablet and printer	I like the phone and tablet and printer	trigram
I like play doll but my brother annoying	I like play doll but my brother annoying	trigram
I like swing and slide	I like swing and slides	trigram
I like tea please	I like tea please	trigram
I like television	I like television	trigram
I like two onion therefore I eat onion	I like two onion therefore I eat onion	trigram
I like you	I like you	trigram
I like your beard	I like your beard	trigram
I like your tshirt	I like your tshirt	trigram
I love apple juice	I love apple juice	trigram
I love christmas livingroom	I love christmas livingroom	trigram
I love horse	I love horse	trigram
I love horse and bunny and dog and cat	I love horse and bunny and dog and cat	trigram
I love pick nose	I love pick nose	trigram
I love play guitar	I love to play guitar	trigram
I love princess	I love princess	trigram
I love red and peach	I love red and peach	trigram
I love tablet phone cd dvd keyboard television film camera	I love tablet phone cd dvd keyboard television film camera	trigram
I love you	I love you	trigram
I need bathroom	I need bathroom	trigram
I need toilet	I need toilet	trigram
I no like thunder	I no like thunder	trigram
I play accordion and trumpet and triangle	I play accordion and trumpet and triangle	trigram
I play socker	I play socker	trigram
I play socker and I love ride bicycle	I play socker and I love ride bicycle	trigram
I play violin	I play violin	trigram
I really see big shark	I really see big shark	trigram
I stomachache	I stomachache	trigram
I twenty three year	I twenty three year	trigram
I twenty year	I twenty year	trigram
I want more socks	I want more socks	trigram
I want more socks my socks bad	I want more socks my socks bad	trigram
I work like bee	I work like bee	trigram
ice creme delicious	Ice creme delicious	trigram
let us play with dolls	Let us play with dolls	trigram
mom	Mom	trigram
mom and my sister and my grandma muscle	Mom and my sister and my grandma muscle	trigram
mom diarrhea	Mom diarrhea	trigram
Mom love cat	Mom loves cat	trigram
Mom play guitar	Mom plays guitar	trigram
mom work dressmaker and dad work filmmaker	Mom works dressmaker and dad works filmmaker	trigram
my brother like play with tablet	My brother likes play with tablet	trigram
my favorite color red	My favorite colors red	trigram
my favorite color yellow	My favorite colors yellow	trigram
my mum dressmaker	My mum dressmaker	trigram
my name baby	My name baby	trigram
my name celine	My name celine	trigram
my name Cinderella	My name Cinderella	trigram
my name Claire and I two year	My name Claire and I two year	trigram
my name is celine	My name is celine	trigram
my nose running	My nose running	trigram
my nose running I cold	My nose running I cold	trigram
my sister also like play with my doll	My sister also likes to play with my doll	trigram
my socks bad	My socks bad	trigram
no Ouch! Ouch! Ouch!	No Ouch! Ouch! Ouch!	trigram
Ouch!	Ouch!	trigram
Ouch! I stomachache	Ouch! I stomachache	trigram
She do not want brush teeth	She does not want brush teeth	trigram
she do washing	She does washing	trigram
she kind	She kind	trigram
she like Cinderella	She likes Cinderella	trigram
she like monster	She likes monster	trigram
she like play ball	She likes to play ball	trigram
she need wash your breast	She needs wash your breast	trigram
she play guitar with Susi	She plays the guitar with Susi	trigram
she pregnant	She pregnant	trigram
sweets and yoghurt delicious	Sweets and yoghurt delicious	trigram
thank you	Thank you	trigram
that scary	That scary	trigram
The cat in house	The cat in house	trigram
they eat pineapple and raspberry	They eat pineapple and raspberry	trigram
they love potty	They love potty	trigram
today I wear blue jeans and yellow tshirt	Today I wear blue jeans and yellow tshirt	trigram
Today my Mum and I go pet shop and buy new puppy	Today my Mum and I go pet shop and buys new puppy	trigram
tomorrow I brush teeth and blow nose	Tomorrow I brush teeth and blow nose	trigram
we like brush teeth	We like brush teeth	trigram
what your name	What your name	trigram

when I ride my bicycle I wear bicycle helmet	When I ride my bicycle I wear bicycle helmet	trigram
when I ride my bicycle I wear helmet	When I ride my bicycle I wear helmet	trigram
yes I love your bottom	Yes I love your bottom	trigram
you are welcome	You are welcome	trigram
You eat sausage and drink apple juice	You eat sausage and drinks apple juice	trigram
you hate banana	You hate banana	trigram
you like sausage	You like sausage	trigram
you love pick nose	You love pick nose	trigram
boy love girl bit	Boy loves girl bit	trigram_smoothed
but my brother hate it	But that my brother hates it	trigram_smoothed
cat in house	The cat in house	trigram_smoothed
caterpillar and duck nice	Caterpillar and duck nice	trigram_smoothed
dad like sun	Dad likes sun	trigram_smoothed
dad love love love love love love love love pick nose	Dad loves love love love love love love love love pick nose	trigram_smoothed
Dad play guitar	Dad plays guitar	trigram_smoothed
do you want eat or drink	Do you want to eat or drink	trigram_smoothed
grandma cook vegetables today	Grandma cooks vegetables today	trigram_smoothed
great you nice I love you and you body and meat	Great you nice I love you and you body and meat	trigram_smoothed
he do his homework	He does his homework	trigram_smoothed
he five years old	He is the five years old	trigram_smoothed
he like you	He likes you	trigram_smoothed
he like your cat and dog	He likes your cat and dog	trigram_smoothed
he love cheese	He loves cheese	trigram_smoothed
he love you	He loves you	trigram_smoothed
he love your bottom	He loves your bottom	trigram_smoothed
he play guitar	He plays guitar	trigram_smoothed
he prince I princess	He prince I princess	trigram_smoothed
Hello my name Louisa	Hello my name Louisa	trigram_smoothed
Hello my name Louisa and I four year	Hello my name Louisa and I are a four years	trigram_smoothed
Hello please help me	Hello please help me	trigram_smoothed
I 23 year	I twenty-three years	trigram_smoothed
I afraid bee	I afraid bee	trigram_smoothed
I afraid vacuum	I afraid vacuum	trigram_smoothed
I also like play with my doll	I is also like to play with my doll	trigram_smoothed
I angry and evil	I is the angry and evil	trigram_smoothed
I artist	I artist	trigram_smoothed
I bird	I bird	trigram_smoothed
I cold	I cold	trigram_smoothed
I do not know	I do not know	trigram_smoothed
I do not know more share	I do not know more share	trigram_smoothed
I do not know you	I do not know you	trigram_smoothed
I do not want brush teeth	I do not want brush teeth	trigram_smoothed
I drink lemonade	I drink lemonade	trigram_smoothed
I eat and sleep	I eat and sleep	trigram_smoothed
I eat sausage and drink apple juice	I eat sausage and drinks in apple juice	trigram_smoothed
I hate cucumber and peanut but I love melon	I hate cucumber and peanut but I love melon	trigram_smoothed
I hate doctor	I hate doctor	trigram_smoothed
I hate pirate	I hate pirate	trigram_smoothed
I hope I play after I finish work	I hope I play after I finish work	trigram_smoothed
I like artist	I like artist	trigram_smoothed
I like August	I like August	trigram_smoothed
I like baby	I like baby	trigram_smoothed
I like cookie and sweets and lollipop	I like cookie and sweets and lollipop	trigram_smoothed
I like dog	I like dog	trigram_smoothed
I like dog and cat but I afraid spider	I like the dog and cat but I afraid spider	trigram_smoothed
I like have apple	I like have apple	trigram_smoothed
I like my brother	I like to my brother	trigram_smoothed
I like my sister	I like that my sister	trigram_smoothed
I like nobody but you because they mean	I like nobody but you because they mean	trigram_smoothed
I like phone	I like phone	trigram_smoothed
I like phone and tablet and printer	I like the phone and tablet and printer	trigram_smoothed
I like play doll but my brother annoying	I like play doll but my brother annoying	trigram_smoothed
I like swing and slide	I like the swing and slides	trigram_smoothed
I like tea please	I like tea please	trigram_smoothed
I like television	I like television	trigram_smoothed
I like two onion therefore I eat onion	I like two onions therefore I eat onion	trigram_smoothed
I like you	I like you	trigram_smoothed
I like your beard	I like your beard	trigram_smoothed
I like your tshirt	I like your tshirt	trigram_smoothed
I love apple juice	I love apple juice	trigram_smoothed
I love christmas livingroom	I love christmas livingroom	trigram_smoothed
I love horse	I love horse	trigram_smoothed
I love horse and bunny and dog and cat	I love to a horse and bunny and dog and cat	trigram_smoothed
I love pick nose	I love pick nose	trigram_smoothed
I love play guitar	I love to play guitar	trigram_smoothed
I love princess	I love princess	trigram_smoothed
I love red and peach	I love to a red and peach	trigram_smoothed
I love tablet phone cd dvd keyboard television film camera	I love tablet phone cd dvd keyboard television film camera	trigram_smoothed
I love you	I love you	trigram_smoothed
I need bathroom	I need bathroom	trigram_smoothed
I need toilet	I need toilet	trigram_smoothed
I no like thunder	I no like thunder	trigram_smoothed
I play accordion and trumpet and triangle	I play accordion and trumpet and triangle	trigram_smoothed
I play socker	I play socker	trigram_smoothed
I play socker and I love ride bicycle	I play socker and I love ride bicycle	trigram_smoothed
I play violin	I play violin	trigram_smoothed
I really see big shark	I really see big shark	trigram_smoothed
I stomachache	I stomachache	trigram_smoothed

I twenty three year	I twenty three years	trigram_smoothed
I twenty year	I twenty years	trigram_smoothed
I want more socks	I want more socks	trigram_smoothed
I want more socks my socks bad	I want more socks my socks bad	trigram_smoothed
I work like bee	I work like bee	trigram_smoothed
ice creme delicious	Ice creme delicious	trigram_smoothed
let us play with dolls	Let us play with dolls	trigram_smoothed
mom	Mom	trigram_smoothed
mom and my sister and my grandma muscle	Mom and my sister and my grandma muscle	trigram_smoothed
mom diarrhea	Mom diarrhea	trigram_smoothed
Mom love cat	Mom loves cat	trigram_smoothed
Mom play guitar	Mom plays guitar	trigram_smoothed
mom work dressmaker and dad work filmmaker	Mom works dressmaker and dad works filmmaker	trigram_smoothed
my brother like play with tablet	My brother likes play with tablet	trigram_smoothed
my favorite color red	My favorite colors red	trigram_smoothed
my favorite color yellow	My favorite colors yellow	trigram_smoothed
my mum dressmaker	My mum dressmaker	trigram_smoothed
my name baby	My name baby	trigram_smoothed
my name celine	My name celine	trigram_smoothed
my name Cinderella	My name Cinderella	trigram_smoothed
my name Claire and I two year	My name Claire and I two years	trigram_smoothed
my name is celine	My name is celine	trigram_smoothed
my nose running	My nose running	trigram_smoothed
my nose running I cold	My nose running I cold	trigram_smoothed
my sister also like play with my doll	My sister also likes to play with my doll	trigram_smoothed
my socks bad	My socks bad	trigram_smoothed
no Ouch! Ouch! Ouch!	No Ouch! Ouch! Ouch!	trigram_smoothed
Ouch!	Ouch!	trigram_smoothed
Ouch! I stomachache	Ouch! I stomachache	trigram_smoothed
She do not want brush teeth	She does not want brush teeth	trigram_smoothed
she do washing	She does washing	trigram_smoothed
she kind	She kind	trigram_smoothed
she like Cinderella	She likes Cinderella	trigram_smoothed
she like monster	She likes monster	trigram_smoothed
she like play ball	She likes to play ball	trigram_smoothed
she need wash your breast	She needs to wash your breast	trigram_smoothed
she play guitar with Susi	She plays the guitar with Susi	trigram_smoothed
she pregnant	She pregnant	trigram_smoothed
sweets and yoghurt delicious	Sweets and yoghurt delicious	trigram_smoothed
thank you	Thank you	trigram_smoothed
that scary	That scary	trigram_smoothed
The cat in house	The cat in house	trigram_smoothed
they eat pineapple and raspberry	They eat the pineapple and raspberry	trigram_smoothed
they love potty	They love potty	trigram_smoothed
today I wear blue jeans and yellow tshirt	Today I wear blue jeans and yellow tshirt	trigram_smoothed
Today my Mum and I go pet shop and buy new puppy	Today my Mum and I go pet shop and buys new puppy	trigram_smoothed
tomorrow I brush teeth and blow nose	Tomorrow I brush teeth and blow nose	trigram_smoothed
we like brush teeth	We like brush teeth	trigram_smoothed
what your name	What is your name	trigram_smoothed
when I ride my bicycle I wear bicycle helmet	When I ride my bicycle I wear bicycle helmet	trigram_smoothed
when I ride my bicycle I wear helmet	When I ride my bicycle I wear helmet	trigram_smoothed
yes I love your bottom	Yes I love your bottom	trigram_smoothed
you are welcome	You are welcome	trigram_smoothed
You eat sausage and drink apple juice	You eat sausage and drinks in apple juice	trigram_smoothed
you hate banana	You hate banana	trigram_smoothed
you like sausage	You like sausage	trigram_smoothed
you love pick nose	You love pick nose	trigram_smoothed

A.3. Screenshots der Andorid Applikation

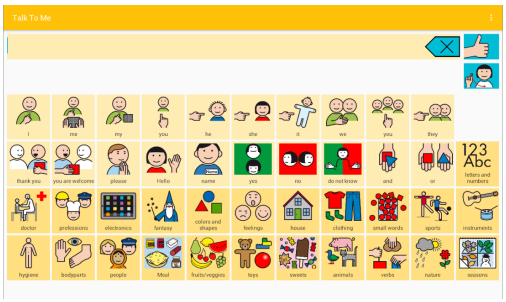

Abbildung A.1.: Startbildschirm

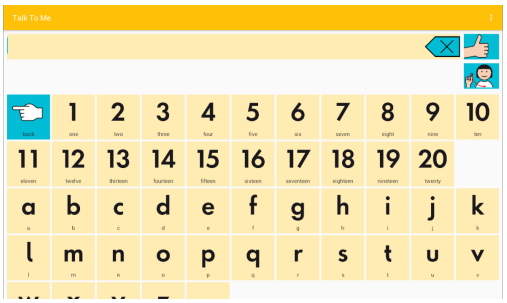

Abbildung A.2.: Kategorie Zahlen und Buchstaben

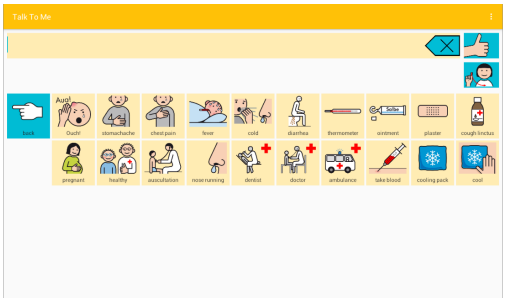

Abbildung A.3.: Kategorie Arzt

Abbildung A.4.: Kategorie Berufe

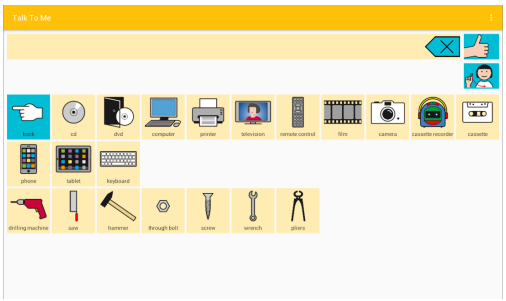

Abbildung A.5.: Kategorie Elektronik

Abbildung A.6.: Kategorie Fantasie

Abbildung A.7.: Kategorie Farben und Formen

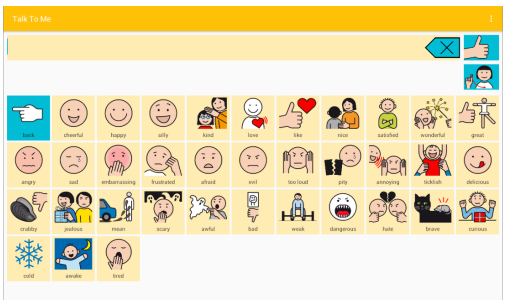

Abbildung A.8.: Kategorie Gefühle

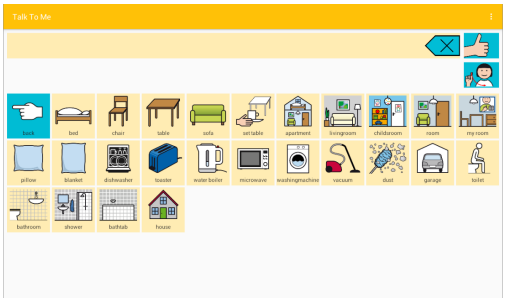

Abbildung A.9.: Kategorie Haus

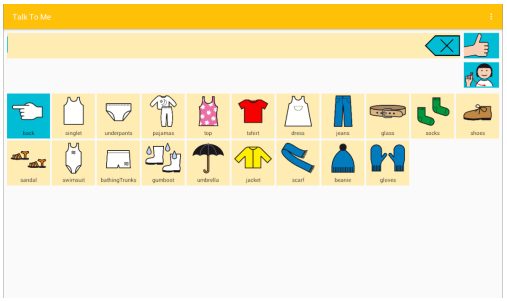

Abbildung A.10.: Kategorie Kleidung

Abbildung A.11.: Kategorie Kleine Wörter

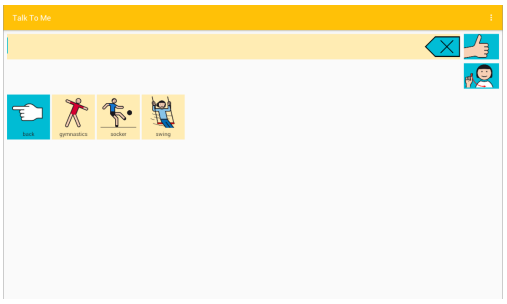

Abbildung A.12.: Kategorie Sport

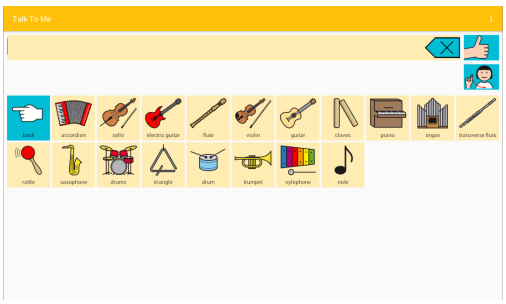

Abbildung A.13.: Kategorie Instrumente

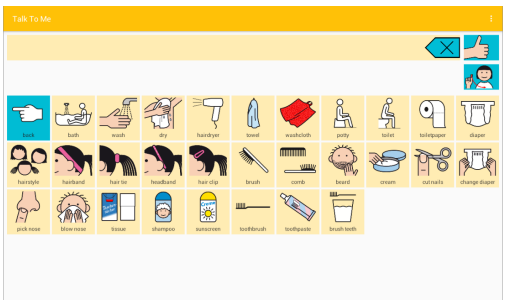

Abbildung A.14.: Kategorie Hygiene

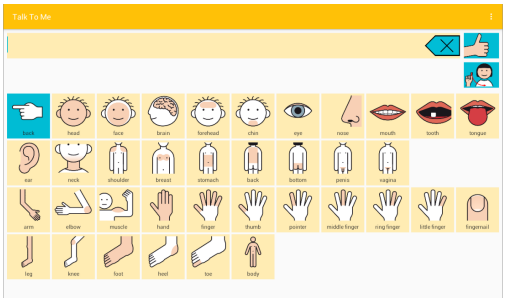

Abbildung A.15.: Kategorie Körperteile

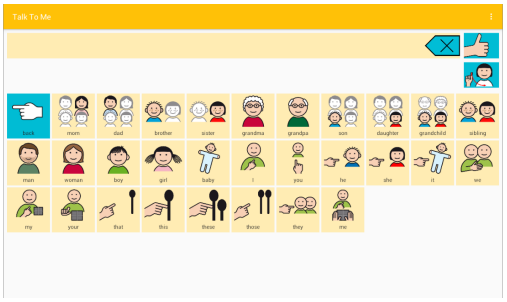

Abbildung A.16.: Kategorie Personen

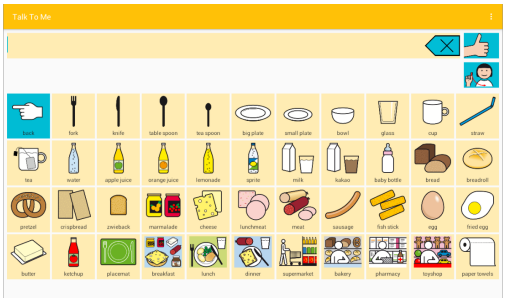

Abbildung A.17.: Kategorie Essen

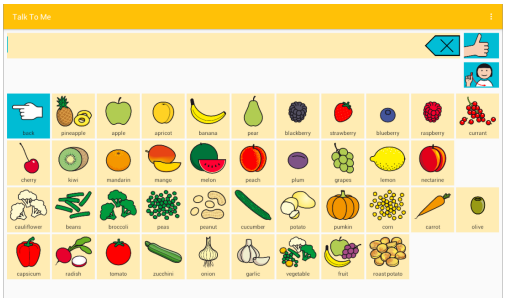

Abbildung A.18.: Kategorie Obst und Gemüse

Abbildung A.19.: Kategorie Spielzeug

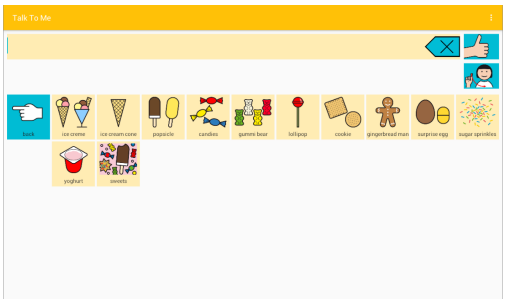

Abbildung A.20.: Kategorie Süßigkeiten

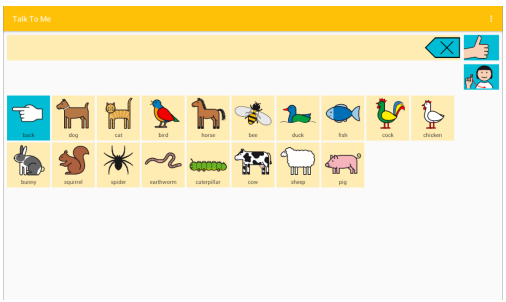

Abbildung A.21.: Kategorie Tiere

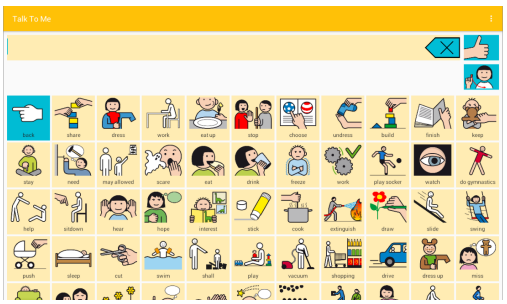

Abbildung A.22.: Kategorie Verben

Abbildung A.23.: Kategorie Natur

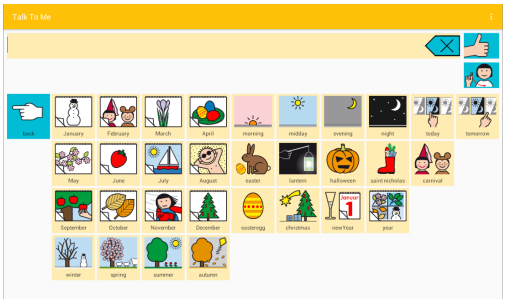

Abbildung A.24.: Kategorie Saison

A.4. Klassifizierung der ICD-10

Klassifizierung der ICD-10 für Sprach- und Sprechstörungen von Krollner und Krollner (2017). Zu den Entwicklungsstörungen gehören

Klasse	Name	Beschreibung
F80.0	Artikulationsstörung	Eine umschriebene Entwicklungsstörung, bei der die Artikulation des Kindes unterhalb des seinem Intelligenzalter angemessenen Niveaus liegt, seine sprachlichen Fähigkeiten jedoch im Normbereich liegen. Inkl. Dyslalie; Entwicklungsbedingte Artikulationsstörung; Funktionelle Artikulationsstörung; Lallen; Phonologische Entwicklungsstörung Exkl. Artikulationsschwäche (bei): Aphasie o.n.A.; Artikulationsschwäche (bei): Apraxie; Artikulationsschwäche (bei): mit einer Entwicklungsstörung der Sprache: expressiv; Artikulationsschwäche (bei): mit einer Entwicklungsstörung der Sprache: rezeptiv; Artikulationsschwäche (bei): Hörverlust; Artikulationsschwäche (bei): Intelligenzstörung
F80.1	Expressive Sprachstörung	Eine umschriebene Entwicklungsstörung, bei der die Fähigkeit des Kindes, die expressiv gesprochene Sprache zu gebrauchen, deutlich unterhalb des seinem Intelligenzalter angemessenen Niveaus liegt, das Sprachverständnis liegt jedoch im Normbereich. Störungen der Artikulation können vorkommen. Inkl. Entwicklungsbedingte Dysphasie oder Aphasie, expressiver Typ Exkl. Dysphasie und Aphasie: entwicklungsbedingt, rezeptiver Typ; Dysphasie und Aphasie: o.n.A.; Elektiver Mutismus; Erworbene Aphasie mit Epilepsie [Landau-Kleffner-Syndrom]; Intelligenzstörung; Tiefgreifende Entwicklungsstörungen

F80.2	Rezeptive Sprachstörung	Eine umschriebene Entwicklungsstörung, bei der das Sprachverständnis des Kindes unterhalb des seinem Intelligenzalter angemessenen Niveaus liegt. In praktisch allen Fällen ist auch die expressive Sprache deutlich beeinflusst, Störungen in der Wort-Laut-Produktion sind häufig. Inkl. Angeborene fehlende akustische Wahrnehmung; Entwicklungsbedingt: Dysphasie oder Aphasie, rezeptiver Typ; Entwicklungsbedingt: Wernicke-Aphasie; Worttaubheit Exkl. Autismus; Dysphasie und Aphasie: entwicklungsbedingt, expressiver Typ; Dysphasie und Aphasie: o.n.A.; Elektiver Mutismus; Erworbene Aphasie mit Epilepsie [Landau-Kleffner-Syndrom]; Intelligenzstörung; Sprachentwicklungsverzögerung infolge von Schwerhörigkeit oder Taubheit
F80.20	Auditive Verarbeitungs- und Wahrnehmungsstörung [AVWS]	-
F80.28	Sonstige rezeptive Sprachstörung	-

F80.3	Erworbene Aphasie mit Epilepsie [Landau-Kleffner-Syndrom]	Eine Störung, bei der ein Kind, welches vorher normale Fortschritte in der Sprachentwicklung gemacht hatte, sowohl rezeptive als auch expressive Sprachfertigkeiten verliert, die allgemeine Intelligenz aber erhalten bleibt. Der Beginn der Störung wird von paroxysmalen Auffälligkeiten im EEG begleitet und in der Mehrzahl der Fälle auch von epileptischen Anfällen. Typischerweise liegt der Beginn im Alter von 3-7 Jahren mit einem Verlust der Sprachfertigkeiten innerhalb von Tagen oder Wochen. Der zeitliche Zusammenhang zwischen dem Beginn der Krampfanfälle und dem Verlust der Sprache ist variabel, wobei das eine oder das andere um ein paar Monate bis zu zwei Jahren vorausgehen kann. Als möglicher Grund für diese Störung ist ein entzündlicher enzephalitischer Prozess zu vermuten. Etwa zwei Drittel der Patienten behalten einen mehr oder weniger rezeptiven Sprachdefekt. Exkl. Aphasie bei anderen desintegrativen Störungen des Kindesalters; Aphasie bei Autismus; Aphasie o.n.A.
80.8	Sonstige Entwicklungsstörungen des Sprechens oder der Sprache	Inkl.: Lispeln
F80.9	Entwicklungsstörung des Sprechens oder der Sprache, nicht näher bezeichnet	Inkl.: Sprachstörung o.n.A.

Abkürzungsverzeichnis

KfG	kontextfreie Grammatik
LM	Language Model
MLE	Maximum Likelihood Estimation
POS	Part of Speech
TTS	Text-to-Speech
WaCky	Web-As-Corpus Kool Yinitiative

Glossar

Corpora

Plural Form von Corpus

Corpus

Sammlung von Texten mit Sätzen

Count

Anzahl der Vorkommen eines Wortes oder N-Gramm in einem Corpus

Logopäden

Arzt der Logopädie für die Behandlung von Sprach- bzw. Sprechstörungen

Logopädin

Weibliche Form von Logopäden

Part of Speech

Deutsch: Wortart. Zum Beispiel wäre „Noun“(N) und „Verb“(V) eine Part-of-speech

REST

REST steht für *Representational State Transfer* und ist ein Programmierparadigma für APIs. Es definiert eine Status unabhängige Client-Server Kommunikation

Sprachstörung

Störung der gedanklichen Erzeugung von Sprache

Sprechstörung

Störung der motorischen Erzeugung von Lauten

Talker

Elektronische Kommunikationshilfe für Menschen mit Sprach- oder Sprechstörungen. Auch bekannt unter dem Namen *Sprachcomputer*

Text-to-Speech

Ein Service, der einen Text in eine synthetisch erstellte Sprache umwandelt

Trainingsdaten

Daten bzw. Sätze, die zum Aufbau eines Language Model verwendet werden

Literaturverzeichnis

[Babst 2016] Babst, Jürgen: *NuVoice Kommunikationssoftware Bedienungsanleitung*. 2.05. Goethestr. 31, D - 34119 Kassel: Prentke Romich GmbH (Veranst.), Juni 2016

[Blank-Mathieu 2001] Blank-Mathieu, Margarete: *Medienerziehung im Kindergarten*. 2001. – URL http://www.kindergartenpaedagogik.de/1291.html. – Zugriffsdatum: 05.08.2016

[Blanken u. a. 2015] Blanken, Gerhard ; Schmidt, Claudia u. a.: *Besondere Kommunikation und Linguistik*. Mainz : HochschulVerlag Mainz, 2015

[Cavnar und Trenkle 1994] Cavnar, William B. ; Trenkle, John M.: N-Gram-Based Text Categorization. In: *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994, S. 161–175

[Chen und Goodman 1999] Chen, Stanley F. ; Goodman, Joshua: An empirical study of smoothing techniques for language modeling. In: *Computer Speech and Language* 13 (1999), Nr. 4, S. 359 – 394. – URL http://www.sciencedirect.com/science/article/pii/S0885230899901286. – Zugriffsdatum: 22.08.2017. – ISSN 0885-2308

[CLiPS Research Center 2017] CLiPS Research Center: *CLiPS*. 2017. – URL http://www.clips.ua.ac.be/pages/pattern-en. – Zugriffsdatum: 27.05.2017

[Cockburn 2000] Cockburn, Alistair: *Writing effective use cases*. 2000. – Vorveröffentlichung Enwurf Nr. 3, Bearbeitungsdatum : 21.02.2000. Veröffentlicht von Addison-Wesley, c. 2001.

[Delogu u. a. 1998] Delogu, Cristina ; Conte, Stella ; Sementina, Ciro: Cognitive factors in the evaluation of synthetic speech. In: *Speech Communication* 24 (1998), Nr. 2, S. 153 – 168. – URL http://www.sciencedirect.com/science/article/pii/S0167639398000090. – Zugriffsdatum: 22.08.2017. – ISSN 0167-6393

[Evert u. a. 2016] Evert, Stefan ; Attardi, Giuseppe ; Baroni, Marco ; Bernardini, Silvia ; Eckart, Kerstin u. a.: *WaCKy Corpora*. 2016. – URL http://wacky.sslmit.unibo.it/doku.php?id=start. – Zugriffsdatum: 07.12.2016

[Han u. a. 2011] Han, Jing ; E, Haihong ; Le, Guan ; Du, Jian: Survey on NoSQL database. In: *2011 6th International Conference on Pervasive Computing and Applications*, Oktober 2011, S. 363–366

[Hautumm u. a. 2012] Hautumm, Annette ; Heller, Dr. E. ; Wagner, Petra: *Hamburger Bildungsempfehlungen*. 2012. – URL http://www.hamburg.de/contentblob/118066/e7057abd11e427bd2985fe7956029dbb/data/bildungsempfehlungen.pdf. – Zugriffsdatum: 05.08.2017

[Hein 1998] Hein, Anna S.: A Chart-Based Framework for Grammar Checking Initial Studies, Januar 1998, S. 68–80

[Hopcroft und Ullman 2000] Hopcroft, John E. ; Ullman, Jeffrey D.: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. 4. München : Oldenbourg Wissenschaftsverlag GmbH, 2000

[Hubwieser u. a. 2010] Hubwieser, Peter ; Löffler, Patrick ; Schwaigner, Petra ; Spohrer, Matthias ; Steinert, Markus ; Voss, Siglinde ; Winhard, Ferdinand: *Informatik*. Stuttgart : Ernst Klett Verlag GmbH, 2010. – URL http://www2.klett.de/sixcms/media.php/71/873277/731068_Informatik_5_KI_netz.pdf. – Zugriffsdatum: 05.08.2017

[Jurafsky und Martin 2008] Jurafsky, Daniel ; Martin, James H.: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. 2. ed., [Pearson International Edition]. Prentice Hall, Pearson Education International, 2008 (Prentice Hall series in artificial intelligence). – ISBN 0-13-504196-1 ; 978-0-13-504196-3

[Jurafsky und Martin 2014] Jurafsky, Daniel ; Martin, James H.: *Speech and Language Processing*. 2014. – Entwurf, Kapitel 4 N-Grams

[Kitzinger 2016] Kitzinger, Annette: *METACOM Symbole*. 2016. – URL http://www.metacom-symbole.de/index.html. – Zugriffsdatum: 20.05.2017

[Krollner und Krollner 2017] Krollner, Dr. Björn ; Krollner, Dr. med. Dirk M.: *ICD-Code F80-F89 Entwicklungsstörungen.* 2017. – URL http://www.icd-code.de/icd/code/F80.1.html. – Zugriffsdatum: 05.08.2017

[Landesinstitut für Lehrerbildung und Schulentwicklung 2011] Landesinstitut für Lehrerbildung und Schulentwicklung: *Bildungsplan Grunschule Deutsch.* 2011. – URL http://www.hamburg.de/contentblob/2481792/d180d66decd915caf50391ab07bdc51d/data/deutsch-gs.pdf. – Zugriffsdatum: 28.07.2017

[Lee und Lee 2014] Lee, Kyusong ; Lee, Gary G.: Sentence completion task using web-scale data. In: *2014 International Conference on Big Data and Smart Computing (BIGCOMP)*, Januar 2014, S. 173–176. – ISSN 2375-933X

[Ludewig und Lichter 2010] Ludewig, Jochen ; Lichter, Horst: *Konzepte und Komponenten der Spezifikation.* Kap. 16.7. In: *Software Engineering*, dpunk.verlag GmbH, 2010. – 387-389

[Nonnenmacher 2016] Nonnenmacher, Dr. med. Facharzt für innere M.: *Sprachstörung.* 2016. – URL http://symptomat.de/Sprachst%C3%B6rungen. – Zugriffsdatum: 05.08.2016

[Oxford University Press 2017] Oxford University Press: *How many words are there in the English language?* 2017

[Prentke Romich GmbH] Prentke Romich GmbH: *Accent 1400.* – URL http://shop.prentke-romich.de/Komplexe-Kommunikationshilfen/Accent-1400::226.html. – Zugriffsdatum: 16.07.2017

[RehaMedia GmbH] RehaMedia GmbH: *MyCORE.* – URL http://www.my-core.de/mycore/mycore_grundlagen/. – Zugriffsdatum: 16.07.2017

[Sachse und Boenisch 2013] Sachse, Stefanie K. ; Boenisch, Jens: *Kern- und Randvokabular in der Unterstützten Kommunikation.* März 2013. – URL http://www.inklusion-lexikon.de/KernundRandvokabular_SachseBoenisch.pdf. – Zugriffsdatum: 19.07.2017

[Sachse u. a. 2013] Sachse, Stefanie K. ; Wagter, Jacqueline ; Schmidt, Lena: Das Kölner Vokabular. In: Hallbauer, Angela (Hrsg.) ; Hallbauer, Thomas (Hrsg.) ; Hüning-Meier, Monika (Hrsg.): *UK kreativ!* isaac (Veranst.), von Loeper Literaturverlag, 2013, S. 35–53. – ISBN 978-3-86059-148-2

[Samanta und Chaudhuri 2013] Samanta, Pratip ; Chaudhuri, Bidyut B.: A simple real-word error detection and correction using local word bigram and trigram. In: *ROCLING*, Association for Computational Linguistics and Chinese Language Processing (ACLCLP), Taiwan, 2013 (Twenty-Fifth Conference on Computational Linguistics and Speech Processing), S. 211–220. – ISBN 978-957-30792-6-2

[Schlosser u. a. 2007] Schlosser, Ralf W. ; Sigafoos, Jeff ; Luiselli, James K. ; Angermeier, Katie ; Harasymowyz, Ulana ; Schooley, Katherine ; Belfiore, Phil J.: Effects of synthetic speech output on requesting and natural speech production in children with autism: A preliminary study. In: *Research in Autism Spectrum Disorders* 1 (2007), Nr. 2, S. 139 – 163. – URL http://www.sciencedirect.com/science/article/pii/S1750946706000134. – Zugriffsdatum: 22.08.2017. – ISSN 1750-9467

[Senatsverwaltung für Bildung, Jugend und Wissenschaft Berlin 2014] Senatsverwaltung für Bildung, Jugend und Wissenschaft Berlin: *Berliner Bildungsprogramm*. 2014. – URL https://www.berlin.de/sen/jugend/familie-und-kinder/kindertagesbetreuung/berliner_bildungsprogramm_2014.pdf. – Zugriffsdatum: 22.08.2017

[Shieber 1985] Shieber, Stuart M.: Evidence against the context-freeness of natural language. In: *Linguistics and Philosophy* 8 (1985), Nr. 3, S. 333–343. – ISBN 978-1-55608-047-0

[Spiccia u. a. 2015] Spiccia, C. ; Augello, A. ; Pilato, G. ; Vassallo, G.: A word prediction methodology for automatic sentence completion. In: *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, Februar 2015, S. 240–243

[Statista GmbH 2017] Statista GmbH: *Anteile der verschiedenen Android-Versionen an allen Geräten mit Android OS weltweit im Zeitraum 30. Juni bis 06. Juli 2017*. 2017. – URL https://de.statista.com/statistik/daten/studie/180113/umfrage/anteil-der-verschiedenen-android-versionen-auf-geraeten-mit-android-os/. – Zugriffsdatum: 09.08.2017

[Statistisches Bundesamt [Destatis] 2017] Statistisches Bundesamt [Destatis]: *Ausstattung privater Haushalte mit Informations- und Kommunikationstechnik - Deutschland*. 2017. – URL https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/AusstattungGebrauchsguetern/Tabellen/Infotechnik_D.html. – Zugriffsdatum: 22.08.2017

[Wendlandt und Niebuhr-Siebert 2000] Wendlandt, Wolfgang ; Niebuhr-Siebert, Sandra: *Sprachstörungen im Kindesalter: Materialien zur Früherkennung und Beratung*. 4. Stuttgart : Georg Thieme Verlag KG, 2000 (Forum Logopädie). – ISBN 9783137785057

[Wikipedia 2016] Wikipedia: *Simple English Wikipedia Dumps*. November 2016. – URL https://dumps.wikimedia.org/. – Zugriffsdatum: 22.08.2017

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 23.08.2017 Louisa Spahl