Bachelorarbeit

Integration einer Sprachsteuerungsfunktion in Mobile Apps

Rubén Nuñez

Herbstsemester 2023

Bachelorarbeit an der Hochschule Luzern – Informatik

Titel: Integration einer Sprachsteuerungsfunktion in Mobile Apps

Studentin/Student: Ruben Nuñez

Studiengang: BSc Informatik

Jahr: 2023

Betreuungsperson: Dr. Florian Herzog

Expertin/Experte: xxx

Auftraggeberin/Auftraggeber: Stefan Reinhard, Bitforge AG

Codierung / Klassifizierung der Arbeit:
☒ Öffentlich (Normalfall)
☐ Vertraulich

Eidesstattliche Erklärung Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt habe, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben habe, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht habe, das Vertraulichkeitsinteresse des Auftraggebers wahren und die Urheberrechtsbestimmungen der Hochschule Luzern respektieren werde.

Ort / Datum, Unterschrift _____

Abgabe der Arbeit auf der Portfolio Datenbank:
Bestätigungsvisum Studentin/Student
Ich bestätige, dass ich die Bachelorarbeit korrekt gemäss Merkblatt auf der Portfolio Datenbank abgelegt habe. Die Verantwortlichkeit sowie die Berechtigungen habe ich abgegeben, so dass ich keine Änderungen mehr vornehmen kann oder weitere Dateien hochladen kann.

Ort / Datum, Unterschrift _____

Verdankung gibt ein separiertes Kapitel dazu

Ausschliesslich bei Abgabe in gedruckter Form: Eingangsvisum durch das Sekretariat auszufüllen

Rotkreuz, den _____ Visum: _____

Abstract

Das Problem dieser Arbeit ist im wesentlichen die Erkennung von Triggerwörtern innerhalb des Kontext einer App. Grundsätzlich ist es unüblich, dass mobile Apps eine integrierte Sprachsteuerungsfunktion anbieten.

Inhaltsverzeichnis

1 Problem, Fragestellung, Vision

Das Kernproblem dieser Bachelorarbeit ist die Erkennung von Triggerwörtern innerhalb eines App-Kontexts. Obwohl Sprachsteuerungstechnologien ein erhebliches Potenzial aufweisen und Assistenten wie Siri oder Alexa weit verbreitet sind, bieten mobile Apps selten eine integrierte Spracherkennung. Dies führt zu einer Lücke, da solche Assistenten nicht spezifisch für App-Kontexte optimiert sind. Diese Arbeit verfolgt das Ziel, diese Lücke zu schliessen und eine integrierte Spracherkennungsfunktion zu entwickeln, die Triggerwörter in einer App effektiv erkennt.

1.1 Fragestellung

Die Fragestellung dieser Arbeit lautet: *Wie kann eine integrierte Sprachsteuerung für eine Mobile Apps entwickelt werden, die speziell das Erkennen von Triggerwörtern ermöglicht, indem Methoden des Machine Learnings genutzt werden?.*

Ausgangslage und Problemstellung

Sprachsteuerungstechnologien haben ein grosses Potenzial und werden bisher vor allem als Sprachsteuerungsassistenten genutzt. Während es etablierte Sprachassistenten wie Siri gibt, fehlt es an Lösungen für eine integrierte Sprachsteuerung in Mobile Apps, insbesondere in Bezug auf das Erkennen von Triggerwörtern.

Ziel der Arbeit und erwartete Resultate

Ziel der Arbeit ist es zum einen, eine Grundlage zu schaffen, um ein Triggerwort oder eine Sequenz von Triggerwörtern in der akustischen Sprache erkennen zu können. Dabei werden Methoden und Werkzeuge aus dem Bereich des Machine Learnings verwendet. Zum anderen soll diese Erkenntnis in eine mobile Plattform wie iOS oder Android integriert werden. Für den Rahmen dieser Arbeit genügt die Integration in eine der genannten Plattformen. Weiterhin werden das Thema Datenschutz und die ethischen Aspekte berücksichtigt.

Gewünschte Methoden, Vorgehen

Das Projekt kann beispielsweise in drei Phasen durchgeführt werden: Technische Abklärungen, Datensammlung und Modelltraining, sowie die Erarbeitung eines Prototypen. Agile Vorgehensweisen sind wünschenswert.

Kreativität, Methoden, Innovation

Bisher sind Sprachsteuerungsfunktionen fast ausschliesslich grossen Akteuren wie Siri vorbehalten. Der innovative Ansatz dieser Arbeit zielt darauf ab, einen Anreiz zu setzen, um diese Funktionen auch in herkömmlichen Apps einzusetzen. Die handfreie Bedienung durch Sprachsteuerung hat das Potenzial, das Benutzererlebnis erheblich zu verbessern.

2 Stand der Forschung

Um diese Arbeit fundiert anzugehen, ist ein Verständnis der Grundlagen in den Bereichen Audioverarbeitung und Machine Learning essenziell. Daher wird in diesem Kapitel ein Überblick über die wichtigsten Themen gegeben. Zudem wird das Kapitel sich mit der Implementation von Sprachsteuerungstechnologien befassen. Darunter fallen die Sprachassistenten wie Siri, Alexa oder Google Assistant. Was natürlich auch nicht fehlen darf, ist eine kleine Einleitung in die Fourier-Transformation. Die Fourier-Transformation ist ein wichtiges Konzept in der Signalverarbeitung und wird in dieser Arbeit verwendet.

2.1 Audio

In der digitalen Welt repräsentiert Audio Schallwellen, die durch eine Reihe von numerischen Werten dargestellt werden Somberg et al., n. d., p.9. beschreibt Audio als: „Fundamentally, audio can be modeled as waves in an elastic medium. In our normal everyday experience, the elastic medium is air, and the waves are air pressure waves.“ Audiosignale werden durch die Funktion $A(t)$ repräsentiert, wobei t die Zeit und $A(t)$ die Amplitude zum Zeitpunkt t angibt. Die Amplitude ist die Stärke des Signals und die Zeit repräsentiert die Position des Signals in der Zeit. Diese Betrachtung ist vor allem in der Elektrotechnik von Bedeutung, da die Amplitude als Spannung angesehen werden kann. Grundsätzlich ist Audio ein kontinuierliches Signal. In der digitalen Welt können wir jedoch nur diskrete Werte darstellen. Daher wird das kontinuierliche Signal in diskrete Werte umgewandelt. Dieser Vorgang wird als *Sampling* bezeichnet (Tarr, n. d., Chapter 3.1).

2.1.1 Sampling

Ein früher Ansatz zur digitalen Darstellung von analogen Signalen war die Pulse-Code-Modulation (PCM). Dieses Verfahren wurde bereits in den 1930er Jahren von Alec H. Reeves entwickelt, parallel zum Aufkommen der digitalen Telekommunikation (Deloraine und Reeves, n. d., p. 57). Im Grundsatz wird es heute noch in modernen Computersystemen nach dem gleichen Verfahren angewendet.

Es folgt eine formelle Definition von Sampling. Ein kontinuierliches Signal $A(t)$ wird in bestimmten Zeitintervallen T_s gesampelt. Diese Zeitintervalle werden auch als Sampling-Periode bezeichnet. Die Sampling-Rate $F_s = \frac{1}{T_s}$ gibt die Anzahl der Samples pro Sekunde an. Angenommen wir haben ein Signal mit einer Sampling-Periode von $T_s = 0.001$. Um nun die Sampling-Rate zu berechnen, müssen wir den Kehrwert der Sampling-Periode berechnen. $F_s = \frac{1}{0.001} = 1000$. Somit erhalten wir eine Sampling-Rate von 1000 Samples pro Sekunde. Nun typische Sampling-Raten sind 44100 Hz oder 48000 Hz. Bei Sampling-Raten wird die Einheit *Hertz* verwendet. Ein Hertz entspricht einer Frequenz von einem Sample pro Sekunde. Ein weiter wichtiger Begriff ist die *Nyquist-Frequenz*. Die Nyquist-Frequenz F_n ist die Hälfte der Sampling-Rate. Also $F_n = \frac{F_s}{2}$. Die Idee hinter der Nyquist-Frequenz ist, dass die Sampling-Rate mindestens doppelt so hoch sein muss wie die höchste Frequenz des Signals. Wenn diese Eigenschaft erfüllt ist, kann das Signal ohne Informationsverlust rekonstruiert werden (Tarr, n. d., Chapter 3.1). Mehr dazu folgt im Unterkapitel *Fourier-Transformation*.

Weiter ist es wichtig zu verstehen, dass ein Sample ein diskreter Wert ist. Und dieser wird in digitalen Systemen durch eine bestimmte Anzahl von Bits dargestellt. Die Anzahl der Bits wird als *Bit-Depth* bezeichnet. Die Bit-Depth bestimmt die Auflösung des Signals. Typische Bit-Depth Werte sind 16 oder 24 Bit (Somberg et al., n. d., p.10).

2.1.2 Frames, Channels, Buffers

Ebendfalls wichtig ist das Verständnis von Frames, Channels und Buffers. Da diese Arbeit sich mit Audio-Systemen beschäftigt, ist es wichtig, die Begriffe *Frame*, *Channel* und *Buffer* zu verstehen. Fangen wir mit dem Begriff *Channel* an. Ein Channel kann als ein einzelnes Audio-Signal angesehen

werden. Ein Mono-Signal hat genau nur einen Channel. Ein Stereo-Signal hat zwei Channels. Ein Surround-Signal hat mehr als zwei Channels. usw. Nun zum Begriff *Frame*. Ein Frame entspricht einem Sample pro Channel. Weiter sind Frames in Buffers organisiert. Ein Buffer ist eine Sammlung von Frames. Typischerweise werden Buffers in Grössen von 64, 128, 256, 512 oder 1024 Frames organisiert. Die Abbildung 1 zeigt die Beziehung zwischen Frames, Channels und Buffers. Die Abbildung wurde basierend auf (Somberg et al., n. d., p.10) erstellt und verdeutlicht die Beziehung zwischen Frames, Channels und Buffers.

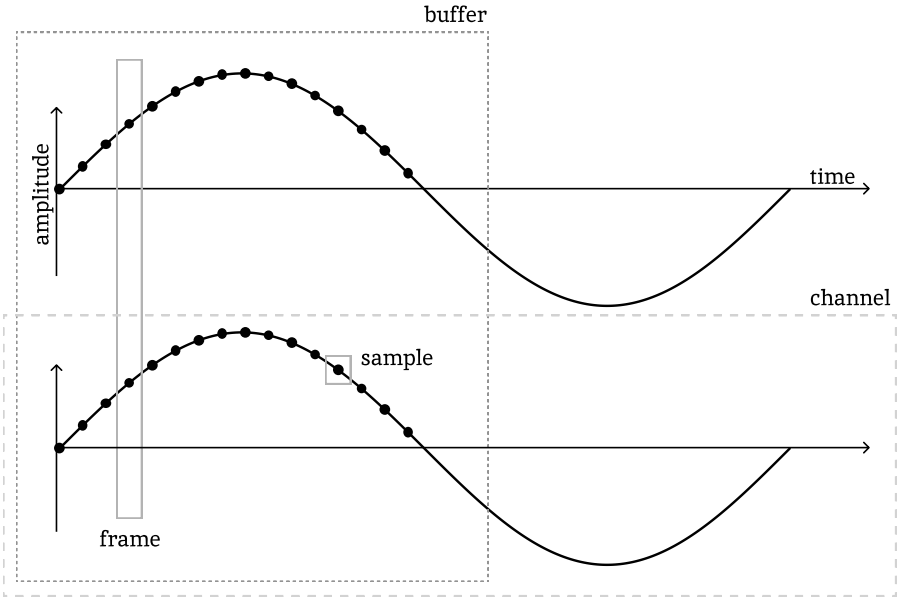

Abbildung 1: Frames, Channels und Buffers

2.1.3 Buffers im Detail

Ein Buffer im Kontext von Audio ist eine aufeinanderfolgende Sammlung von Frames. Die bereits angesprochene Grösse eines Buffers bestimmt im wesentlichen die Latenzzeit des Systems. Kleine Buffer-Grössen haben eine geringe Latenzzeit, während grosse Buffer-Grössen eine hohe Latenzzeit haben (Somberg et al., n. d., p.10). Der Trade-Off ist dass kleine Buffer-Grössen zu einer höheren CPU-Auslastung führen, während bei grossen Buffer-Grössen das nicht der Fall ist. Das liegt daran, dass bei kleinen Buffer-Grössen die CPU häufiger aufgerufen wird, um die Buffers zu verarbeiten.

Nun betrachten wir die mögliche Anordnung eines Buffers, wie in den folgenden Abbildungen dargestellt. Es gibt zwei Möglichkeiten, wie Buffers angeordnet werden können: *Interleaved* und *Non-Interleaved*. Bei der *Interleaved*-Anordnung werden die Samples der einzelnen Channels nacheinander in sequentieller Reihenfolge in den Buffer geschrieben. Im Gegensatz dazu werden bei der *Non-Interleaved*-Variante die Samples eines Channels nacheinander in den Buffer geschrieben, bevor die Samples des nächsten Channels hinzugefügt werden. Dieser Vorgang wird für jeden Channel wiederholt. Die Abbildung 2 zeigt die Unterschiede zwischen den beiden Anordnungen. Jede Zelle der Tabelle entspricht einem Sample. L und R stehen exemplarisch für die Channels Left und Right. Die erste Zeile entspricht der *Interleaved*-Anordnung und die zweite Zeile der *Non-Interleaved*-Anordnung. Die Abbildung wurde basierend auf (Somberg et al., n. d., p.11) erstellt.

L	R	L	R	L	R	L	R
L	L	L	L	R	R	R	R

Abbildung 2: Frames in Interleaved und Non-interleaved Buffers

Mit diesem Wissen kennen wir nun die Unterschiede zwischen den beiden Anordnungen. Für die

Anwendung ist es wichtig zu verstehen, mit welcher Anordnung die verwendete API arbeitet.

2.2 Fourier-Transformation

Die Fourier-Transformation ist ein zentrales Werkzeug der Fourier-Analyse, einem Teilgebiet der Mathematik, das sich mit der Zerlegung von Funktionen in Frequenzkomponenten beschäftigt. Im Kern handelt es sich bei der Fourier-Analyse um die Approximation einer Funktion durch Überlagerung von Schwingungen mit unterschiedlichen Frequenzen. Dieses Konzept wird auch von Prof. Dr. Weitz in seinem Youtube Video erläutert (Weitz, n. d., 2:20). Mathematisch ausgedrückt kann die kontinuierliche Fourier-Transformation eines Signals $f(t)$ wie folgt definiert werden:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t}\, dt$$

$F(\omega)$ ist die kontinuierliche Fourier-Transformation von $f(t)$ (Weitz, n. d., 49:27). Als kleines Rechenbeispiel betrachten wir die kontinuierliche Fourier-Transformation der einfachen periodischen Funktion $f(t) = sin(t) + \frac{1}{2} sin(2t)$.

Um die Fourier-Transformation für $f(t) = \sin(t) + \frac{1}{2}\sin(2t)$ zu berechnen, gehen wir schrittweise vor:

1. Zerlegung des Signals: Zunächst zerlegen wir $f(t)$ in seine beiden Komponenten: $\sin(t)$ und $\frac{1}{2}\sin(2t)$. Die Fourier-Transformation wird für jede dieser Komponenten separat berechnet und die Ergebnisse werden dann zusammengefasst.

2. Fourier-Transformation von $\sin(t)$: Die Fourier-Transformation von $\sin(t)$ ergibt zwei Delta-Funktionen in der Frequenzdomäne, jeweils bei $\omega = 1$ und $\omega = -1$. Die Fourier-Transformation für $\sin(t)$ lautet:

$$\mathcal{F}\{\sin(t)\} = i\,(\delta(\omega + 1) - \delta(\omega - 1))$$

3. Fourier-Transformation von $\frac{1}{2}\sin(2t)$: Ähnlich wie im vorherigen Schritt, ergibt die Fourier-Transformation von $\frac{1}{2}\sin(2t)$ zwei Delta-Funktionen bei $\omega = 2$ und $\omega = -2$. Die Fourier-Transformation für $\frac{1}{2}\sin(2t)$ lautet:

$$\mathcal{F}\{\frac{1}{2}\sin(2t)\} = \frac{i}{2}\,(\delta(\omega + 2) - \delta(\omega - 2))$$

Das Zusammenfassen der beiden Ergebnisse ergibt die endgültige Fourier-Transformierte für $f(t) = \sin(t) + \frac{1}{2}\sin(2t)$:

$$F(\omega) = \frac{i}{2}\,(\delta(\omega + 1) - \delta(\omega - 1)) + \frac{i}{4}\,(\delta(\omega + 2) - \delta(\omega - 2))$$

Die diskrete Fourier-Transformation (DFT) ist eine diskrete Version der Fourier-Transformation. Welche die Fourier-Transformation auf diskrete Signale anwendet. Die DFT ist näher an der Anwendung, da Signale in digitalen Systemen diskret sind.

3 Ideen und Konzepte

Das Problem dieser Arbeit ist im wesentlichen die Erkennung von Triggerwörtern innerhalb des Kontext einer App. Grundsätzlich ist es unüblich, dass mobile Apps eine integrierte Sprachsteuerungsfunktion anbieten.

4 Methoden

Das Problem dieser Arbeit ist im wesentlichen die Erkennung von Triggerwörtern innerhalb des Kontext einer App. Grundsätzlich ist es unüblich, dass mobile Apps eine integrierte Sprachsteuerungsfunktion anbieten.

5 Realisierung

Das Problem dieser Arbeit ist im wesentlichen die Erkennung von Triggerwörtern innerhalb des Kontext einer App. Grundsätzlich ist es unüblich, dass mobile Apps eine integrierte Sprachsteuerungsfunktion anbieten.

6 Evaluation und Validation

Das Problem dieser Arbeit ist im wesentlichen die Erkennung von Triggerwörtern innerhalb des Kontext einer App. Grundsätzlich ist es unüblich, dass mobile Apps eine integrierte Sprachsteuerungsfunktion anbieten.

7 Ausblick

Das Problem dieser Arbeit ist im wesentlichen die Erkennung von Triggerwörtern innerhalb des Kontext einer App. Grundsätzlich ist es unüblich, dass mobile Apps eine integrierte Sprachsteuerungsfunktion anbieten.

8 Anhang

8.1 Projektmanagement

Das Projektmanagement spielt eine zentrale Rolle in der Vorbereitungsphase meiner Bachelorarbeit und bildet die Grundlage für den Erfolg des gesamten Vorhabens. Dabei geht es nicht nur um die reine Planung, sondern auch um eine effiziente Steuerung und kontinuierliche Kontrolle aller Arbeitspakete und derer Ergebnisse. Die besondere Herausforderung meiner Arbeit liegt darin, das umfangreiche Themengebiet, das für diese Bachelorarbeit relevant ist, innerhalb des engen Zeitrahmens von 14 Wochen sinnvoll und fundiert zu bearbeiten. Das Themengebiet umfasst diverse Bereiche der Informatik. Darunter fallen Audioverarbeitung, maschinelles Lernen, Softwareentwicklung und auch einiges an mathematischem Hintergrundwissen. Daher wurde ein agiles Vorgehensmodell gewählt. Dies bedeutet, dass sowohl die Planung als auch die Umsetzung in iterative Zyklen unterteilt sind. Während es zu Beginn eine grobe Struktur und Zielsetzung gibt, ermöglicht diese Herangehensweise Flexibilität in der Durchführung. Dadurch können Veränderungen oder unerwartete Ereignisse leichter integriert und die Bachelorarbeit fortlaufend optimiert werden.

8.2 Grobplanung

Die Grobplanung zeigt die wichtigsten Meilensteine des Projekts auf. Ausserdem werden die Themengebiete, die für die Bachelorarbeit relevant sind, aufgezeigt.

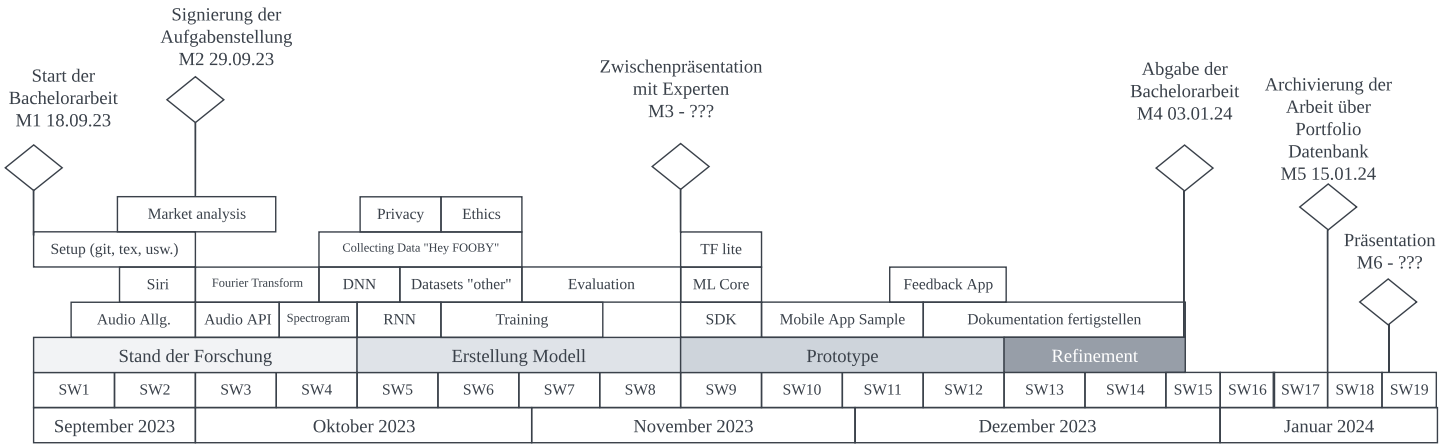

Abbildung 3: Grobplanung

8.2.1 Produkt Backlog

In der Vorbereitungsphase kann ein anfängliches Produkt Backlog als einfache Tabelle dargestellt werden. Ein Beispiel für eine solche Tabelle ist in Abbildung 5 dargestellt.

8.2.2 Risikomanagement

Risikomanagement dient dem Zweck, mögliche Probleme vorwegzunehmen. Die Verwendung von Checklisten, Brainstorming mit den Anspruchsgruppen und die von Erfahrungen aus früheren Projekten sind mögliche Strategien zur Identifikation möglicher Risiken.

Abbildung 4: Tabelle für das anfängliche Product Backlog

Tabelle 1: Beispiel-Tabelle für Risikomanagement

Kopf 1	Kopf 2	Kopf 3
Wert 1	Wert 2	Wert 3
Wert 4	Wert 5	Wert 6

Tabelle 2: Eine einfache Tabelle

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Deloraine, E. M., & Reeves, A. H. (n. d.). The 25th anniversary of pulse code modulation. *IEEE Spectrum*, *2*(5), 56–63. https://doi.org/10.1109/MSPEC.1965.5212943

Somberg, G., Davidson, G., & Doumler, T. (n. d.). A Standard Audio API for C++: Motivation, Scope, and Basic Design ["C++ is there to deal with hardware at a low level, and to abstract away from it with zero overhead." – Bjarne Stroustrup, Cpp.chat Episode #44]. *Programming Language C++*.

Tarr, E. (n. d.). *Hack audio : : an introduction to computer programming and digital signal processing in MATLAB* (1st edition). Routledge.

Weitz, P. D. E. (n. d.). *Fourier-Analysis in 100 Minuten* [Zugriff am: 06.10.2023]. YouTube. https://www.youtube.com/watch?v=zXd743X6I0w

Aufgabenstellung

Integration von Sprachsteuerungstechnologien in Mobile Apps, insbesondere zur Erkennung von Triggerwörtern.

Projektteam

- Student:in: Rubén Nuñez
- Betreuer:in: Herzog
- Firma: Bitforge AG

Auftraggeber

- Firma: Bitforge AG
- Ansprechperson: Stefan Reinhard
- Funktion: Head of Mobile
- Adresse: Zeughausstrasse 39, 8004 Zürich
- Telefon: +41 55 211 02 41
- E-Mail: stefan.reinhard@bitforge.ch
- Website: www.bitforge.ch

Sonstige Bemerkungen

Grundkenntnisse in Machine Learning, speziell im Bereich der Spracherkennung, sowie Erfahrung mit entsprechenden APIs sind erforderlich.