Jonas
FREIKNECHT

KI-Sprachassistenten mit Python entwickeln

Datenbewusst, Open Source und modular

Zahlreiche KI-Anwendungsfälle

HANSER

HANSER

Jonas Freiknecht

KI-Sprachassistenten mit Python entwickeln

Datenbewusst, Open Source und modular

Der Autor:
Jonas Freiknecht, Landau
www.jofre.de

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

Lektorat: Sylvia Hasselbach
Copy editing: Walter Saumweber, Ratingen
Coverkonzept: Marc Müller-Bremer, München, *www.rebranding.de*

Covergestaltung: Max Kostopoulos
Titelmotiv: gettyimages.de/CHRISTOPH BURGSTEDT/SCIENCE PHOTO LIBRARY
Satz: Manuela Treindl, Fürth

Print-ISBN: 978-3-446-47231-0
E-Book-ISBN: 978-3-446-47448-2
E-Pub-ISBN: 978-3-446-47659-2

Für Justus und Elisa

Inhalt

1 Der Sprachassistent in unserem täglichen Leben

Der Sprachassistent – ein meist kleiner, optisch ansprechender Computer – ist mittlerweile fester Bestandteil vieler Haushalte. Der Grund für die rasche Verbreitung ist schnell gefunden: Die Interaktion via Sprache ist um ein Vielfaches leichter, als ein kompliziertes Interface zu bedienen. *Licht Wohnzimmer* spricht sich zügiger und intuitiver aus als in einem User Interface (UI) das Wohnzimmer zu suchen und den Lichtschalter zu drücken. Außerdem muss niemand zu einem Bedienfeld laufen und eine Taste drücken oder das Handy zücken, denn unsere Stimme ist im wahrsten Sinne des Wortes *wireless*.

Auch die Erweiterung eines Sprachassistenten ist denkbar einfach. In der Regel fragt ein Gerät, ob eine bestimmte Funktion aktiviert werden soll, wenn erkannt wird, dass diese eine bestimmte Aufgabe übernehmen kann. An einem PC oder Smartphone müsste man nun ein Programm suchen, dass dieser Aufgabe gewachsen ist und es gegebenenfalls installieren und lernen, es zu bedienen. Möchten Sie aber etwa eine Einkaufsliste für die Geburtstagsfeier Ihrer Tochter anlegen und auf Ihr Smartphone synchronisieren, dann machen Sie das in der Regel

über einen einfachen Befehl, der automatisch die Funktion *Einkaufsliste* aktiviert.

Doch nicht nur für Benutzer ist der Einsatz eines Sprachassistenten eine Erleichterung. Da Sie sich dieses Buch ansehen, gehe ich davon aus, dass Sie (freiwillig oder unfreiwillig) mit dem Gedanken spielen, ein derartiges Gerät zu entwickeln, und sich auch damit auseinandersetzen müssen, gewisse Funktionen dafür zu programmieren (wir werden das in Kapitel 6 ausgiebig tun). Und auch hier kommt uns die menschliche Sprache gelegen, denn ein umständliches Design eines User Interface entfällt für die Entwickler (es sei denn, Sie entwickeln für den *Echo Show*) und zumindest mir persönlich ist das immer eine der unliebsamsten Aufgaben, muss ein Interface doch auf hunderten verschiedenen Geräten ordentlich aussehen und funktionieren. Also auch als Entwickler profitiert man von der einfachen Interaktion per Stimme. Voraussetzung ist, dass man sich ein Framework schafft, das Stimme versteht, doch auch das werden wir zusammen gemeinsam in diesem Buch in aller Ausführlichkeit tun.

Allerdings hat auch ein Sprachassistent seine Tücken. Allen voran fällt das Thema Datenschutz immer mal wieder unangenehm auf, wenn etwa Transkripte von Interaktionen mit Amazons Alexa auftauchen, die von Dienstleistern abgeschrieben und somit auch gelesen werden. Auch wenn das der Weiterentwicklung und der Verbesserung des Sprachverständnisses dient, bekommt man eventuell ein mulmiges Gefühl im Bauch, da auf den Festplatten eines IT-Giganten gespeichert ist, was im eigenen Wohnzimmer gesprochen wird. Auch diesem Thema widme ich einen Teil dieses Buches, und tatsächlich ist es sogar einer der Gründe, warum ich mich darangesetzt habe, einen eigenen Sprachassistenten zu schreiben. Ich wollte eine Offline-Variante

schaffen, die nur für die nötigsten Aufrufe eine Internetverbindung herstellt.

HINWEIS: Es gibt sehr viele Sprachassistenten – Amazon Alexa, Google Home, Mycroft als Open-Source-Variante, Siri von Apple, Samsung Bixby oder Microsoft Cortana. Ich hoffe, Sie sehen es mir nach, dass ich nicht jedes Gerät im Haus aufgestellt habe, um darüber berichten zu können – die fertigen Produkte sollen uns ja nur als Orientierungshilfe dienen. Demnach werden meine Beispiele in der Regel an Amazons Alexa dargestellt, wobei das keinerlei Qualitätsmerkmal dieser einen Ausführung ist. Sie war einfach zuerst da.

Des Weiteren ist der Zugriff auf einen Sprachassistenten in der Regel nicht eingeschränkt. Die Einkaufsfunktion von Alexa beispielsweise ist ab Werk nicht deaktiviert. Das bedeutet, dass jeder, der in Rufweite des Geräts steht, Einkäufe tätigen kann. Diese Erfahrung habe ich bereits am eigenen Leib gemacht, denn bis ich das entsprechende Häkchen in den Optionen der App gefunden hatte, hatten meine Kinder schon zwei Monate lang eine zusätzliche Musikoption gebucht. Eine andere Anekdote hat ein ehemaliger Arbeitskollege von IBM erzählt, dessen Töchter sich wohl seit einiger Zeit die Mathehausaufgaben vom heimischen Assistenten erledigen ließen, um mehr Zeit zum Spielen zu haben. Von den Kollegen, die im Büro die Lautstärke auf 10 stellen und DJ Bobo spielen lassen, fange ich gar nicht erst an.

Es zeichnet sich also ab, dass Kontrollmechanismen, die nicht nur stur Befehle interpretieren und ausführen, einen wirklichen Mehrwert bringen würden. In einigen Fällen kann es gewünscht sein, dass nur eine bestimmte Person eine bestimmte Funktion ausführen darf, etwa das bereits angesprochene Einkaufen oder auch die Administration des Sprachassistenten. Aber auch im Bereich der Hausautomatisierung, in dem der räumliche Aspekt eine Rolle spielt, kann es hilfreich sein, z. B. zwischen Stimmen unterscheiden zu können, sodass nicht durch Fehlinterpretation eines Funktionsaufrufs im Kinderzimmer nachts der Rollladen hochgefahren oder das Licht eingeschaltet wird. Besagte Kontrollmechanismen werden wir in Abschnitt 6.6.1 besprechen und auch praktisch implementieren.

Letzteres ist ein gutes Stichwort, um darauf hinzuweisen, wie wir in diesem Buch arbeiten. Ich möchte mit Ihnen die Themen nicht nur trocken durchsprechen, sondern nach und nach implementieren, um herauszufinden, wie wir bestimmte Funktionen bestmöglich umsetzen können. Besonders im Bereich der künstlichen Intelligenz (KI; ich mag diesen Begriff nicht besonders, werde ihn aber dennoch verwenden) möchte ich durch die praktische Anwendung versuchen, für Klarheit zu sorgen und zu zeigen, was sich hinter dem Begriff *Intelligenz* im Kontext von Sprachassistenten oder Assistenzsystemen im Allgemeinen verbirgt, inwiefern man von einem eigenständig denkenden System sprechen kann und wie nah wir einer allgemeinen KI (AGI; Artificial general intelligence) mit unserem Projekt kommen können.

HINWEIS: Zu diesem Buch gibt es ein *Github* Repository mit der Code-Basis für die einzelnen Kapitel, sodass Sie auch ohne Schreibarbeit den Fortschritt jedes Abschnitts nachvollziehen können.

Sie finden es hier:
https://github.com/padmalcom/AISpeechAssistant

Ziel des praktischen Teils soll sein, dass Sie am Ende des Tages einen eigenen Sprachassistenten implementiert haben, den Sie mit Ihrer Stimme steuern und beliebig um Funktionen erweitern können.

1.1 Warum ein eigener Sprachassistent?

Nun könnten Sie natürlich auch einfach in den nächstbesten Elektronikgroßhandel gehen und sich einen der diversen Geräte aus dem Hause Google, Amazon, Samsung, Apple etc. kaufen. Warum also sollten Sie sich selbst die Mühe machen und einen Assistenten entwerfen? Hierfür gibt es einige gute Gründe.

Dieses Buch ist in seiner Gesamtheit ein Werk, das Sie von Beginn an anhand eines praktischen Anwendungsfalls durch die Applikationsentwicklung in Python führt und bis auf wenige Ausnahmen jedes Thema behandelt, mit dem Sie u. a. im Kontext Data Analytics, Natural Language Processing (NLP), Audio Processing, Deep Learning, Machine Learning, Kompilierung,

Logging, Error Handling etc. in Berührung kommen. Kurzum: Es bietet Ihnen die Möglichkeit, Ihre Entwicklerfähigkeiten auszubauen und zu festigen.

Natürlich werden wir uns auch, wie bereits gesagt, dem Thema KI widmen und die Unterschiede zwischen regelbasierten Entscheidungen und maschinellem Lernen kennenlernen, ganz konkret, wenn es darum geht die Befehle der Sprecher zu interpretieren. Ebenso schauen wir uns den Bereich Deep Learning im Kontext Sprachsynthese, NLP und Zeitreihenvorhersagen an. Wir werden in Kapitel 6 auf verschiedene Architekturen (*Recurrent Neural Networks* (RNN), *Long Short-Term Memory* (LSTM) und *Transformer*) eingehen und lernen, wie diese funktionieren, wie sie aufeinander aufbauen und wie wir sie für unsere Zwecke nutzbar machen können. Eine Einschränkung muss ich jedoch machen: Den generellen Aufbau von neuronalen Netzen und die diesen zugrunde liegenden Mechanismen, wie etwa Optimierungsfunktionen à la *Gradient Descent*, werden wir nicht durchsprechen, sondern als gegeben hinnehmen. Da gibt es bessere Bücher und Onlinekurse, die sich diesen Themen in der notwendigen Ausführlichkeit widmen. Jedoch werden wir uns anschauen, wie wir Trainingsdaten erzeugen, bereinigen und das Training mehrerer neuronaler Netze durchführen, um zu lernen, wie wir die Güte und den Fortschritt des Trainingsvorgangs beurteilen können und wie wir die fertig trainierten Modelle in unseren Sprachassistenten einbinden.

Ein weiterer Grund, einen eigenen Sprachassistenten zu schreiben, sollte sein, dass Sie die Hoheit über Ihre eigenen Daten haben; Datensouveränität hat sich dafür als Begriff in den letzten Jahren durchgesetzt. Ich möchte hier ausdrücklich betonen, dass wir in diesem Buch keinerlei Cloud-Dienste

nutzen, sondern alle Berechnungen *on-premises* durchführen. On-premises bedeutet, dass Vorgänge lokal auf dem von Ihnen genutzten Gerät stattfinden und nicht an entfernte Geräte ausgelagert werden. Natürlich gibt es Ausnahmen, z. B. wenn wir eine Funktion schreiben, die das Wetter abfragt, unseren Standort feststellt oder Streams abspielt. Aber dann senden wir nur die nötigsten Informationen. Und Ihr Sprachassistent funktioniert auch dann, wenn keine Internetverbindung besteht.

Bei der Verarbeitung der Audioeingabe bleibt es also Ihnen überlassen, ob Sie Texte mitschneiden möchten, etwa um die Befehlsverarbeitung zu verbessern. Wir werden das in diesem Buch nicht tun, da wir von sogenannten Aktivierungswörtern (*Wake Words*) Gebrauch machen werden, die die Interpretation der Sprache erst aktivieren, wenn ein bestimmtes Wort gesprochen wird. Weiterhin machen wir den Zustand des Assistenten zu jeder Zeit durch ein Icon sichtbar, sodass Sie und alle, die Ihren Assistenten später verwenden, sicher sein können, wann er wartet, lauscht, denkt oder spricht. Die Offline-Fähigkeit hat noch einen anderen Vorteil: Der Assistent wird portabel. Natürlich muss für die entsprechende Stromversorgung gesorgt werden, jedoch sind wir nicht davon abhängig, dass eine Internetverbindung besteht. Eine weitere Herausforderung, die auch für mich als Motivation diente, ist, den Assistenten auf deutscher oder besser nicht englischer Sprache zu implementieren. Viele fertige Bibliotheken, die mit Sprache oder Text arbeiten, sind auf das Englische ausgerichtet und funktionieren überragend. Doch wenn es dazu kommt, eine Umsetzung in Spanisch, Französisch oder wie in unserem Falle Deutsch anzugehen, stoßen wir bei vielen fertigen Bausteinen schnell an die Grenzen. Und diese Grenzen gilt es zu evaluieren und zu durchbrechen, entweder mit Workarounds oder mit Fleiß

und Schreibarbeit. Sollten Sie nicht hauptberuflich Softwareentwicklung betreiben, lernen Sie hier, mit welchen Problemen man in diesem Berufsfeld häufig konfrontiert wird und wie diese zu lösen sind.

Der letzte Punkt, den ich hier anführen möchte, ist Spaß. Mir hat es Freude gemacht herauszufinden, wie die Entwickler im Silicon Valley bestimmte Probleme gelöst und welche Fehler sie sicher zu Beginn gemacht haben. Auch bei mir war der Lerneffekt hoch, denn meistens ist man ja als Spezialist auf eine Domäne ausgerichtet (Data Science, Web, Backend, Data Engineering …) und hier schauen wir kontinuierlich über den Tellerrand. Durch die modulare Erweiterbarkeit des Assistenten können Sie in beliebige Gebiete reinschauen, die Sie interessieren und entsprechende Funktionen an das System andocken lassen.

1.2 Immer präsent: Das Thema Datenschutz

Ich habe einen Freund, der Sicherheitsprüfungen für Firmen durchführt und Zertifizierungen ausstellt. Auch wenn es in seiner täglichen Arbeit mehr um Sicherheitsaspekte von Arbeitsausrüstung und Maschinen geht, war seine Reaktion auf die Alexa (richtigerweise *Amazon Echo*; *Alexa* bezeichnet nur den Assistenten, nicht das Device) in unserem Wohnzimmer, freundlich ausgedrückt, eher verhalten. Wieso rufen Sprachassistenten bei vielen Menschen eine derartige Reaktion hervor?

Sucht man im Internet nach *Alexa Voice Transcripts*, wird man schnell fündig und erfährt, was Amazon von den Benutzereingaben für die Verbesserung seiner Dienste verwertet und was nicht. So wurde 2019 bekannt, dass sich zwar über die Konfiguration einstellen lässt, dass eigene Stimmaufnahmen gelöscht werden, jedoch die übersetzten Befehle weiter gespeichert werden (Kelly & Statt, 2019). Zwar sind diese häufig harmlos, aber der Teufel steckt bekanntlich im Detail und wenn nur eine von 100.000 Transkripten eine Kontonummer, Geschäftsgeheimnisse oder Krankendaten beinhaltet, ist die Speicherung in Frage zu stellen.

Und haben Sie schon mal erlebt, dass Ihr Assistent angesprungen ist, obwohl Sie das Aktivierungswort nicht gesprochen haben? Der Klassiker bei uns ist die Frage *War lecker?*, die als *Alexa* interpretiert wird (mit Kindern sagt man diesen Satz öfter, als man denkt) und schon beginnt das Gerät aufzuzeichnen und sendet Daten, auch wenn es das in dem Moment nicht soll und die Nutzer auch nicht damit rechnen, dass es das tut.

Ein noch viel einfacherer Fall passiert in unserem Haushalt recht häufig. Bestellt man bei Amazon und erhält in Kürze eine Lieferung, hat Alexa die Angewohnheit, gelb zu leuchten, bis man per *Benachrichtigung* abfragt, welcher Hinweis denn vorliegt. Das Team rund um Amazons Sprachassistenten ist so schlau, dass es diese Funktion über Weihnachten deaktiviert, jedoch stehen ja die restlichen elf Monate jede Menge Geburtstage und Feierlichkeiten an. Und so ist hier und da schon im Vorfeld bekannt geworden, was eine bestimmte Person geschenkt bekommt. Das Thema mag man jetzt mit einem Schmunzeln abtun, ist es doch kein großes Geheimnis, wo die Geburtstagsgeschenke herkommen. Doch haben Sie mal darüber nachgedacht, dass Ihrem Assistenten gegebenenfalls Ihre

Einkaufshistorie bekannt ist? So passiert es seit Kurzem, dass eine Benachrichtigung angezeigt wird, die mich fragt, ob ich einen Artikel bewerten möchte, den ich kürzlich gekauft habe. Hier wird unmissverständlich klar, dass Sprachassistenten auch Daten und somit im entferntesten Sinne Umsatz generieren sollen und nicht nur das tägliche Leben erleichtern.

Die eben angesprochene Tatsache, dass Amazon Audioaufnahmen von uns vorliegen, sagt einiges über den Verarbeitungsprozess der Daten aus. Es hat den Anschein, dass die Logik zur Verarbeitung des gesprochenen Befehls nicht auf der Hardware selbst liegt, sondern in der Cloud (oder für Realisten: in einem der Rechenzentren von Google, Microsoft, Amazon etc., siehe Bild 1.1). Das schicke Gerät, das wir in unseren Räumen stehen haben, ist in der Regel also nur ein Datentransporteur oder Proxy, der zwar die Aktivierungswörter wahrzunehmen versteht, den Mitschnitt aber an einen Cloud-Service streamt, um ihn dort auch zu prozessieren.

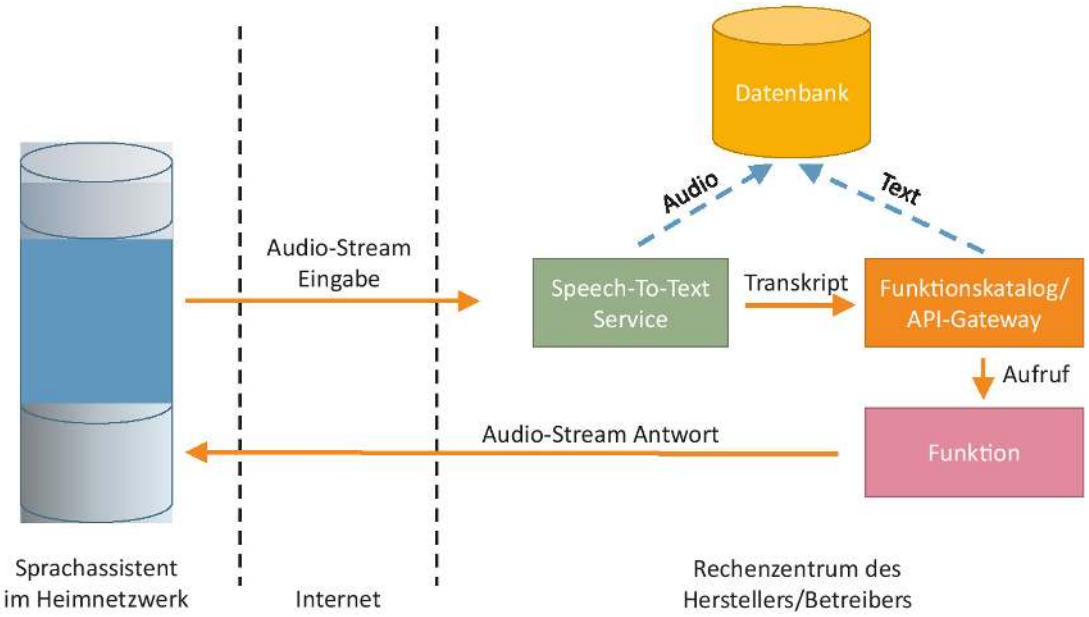

Bild 1.1 Datenfluss bei Aufruf eines Sprachassistenten mit Backend in der Cloud des Herstellers

Die Begründung für dieses Vorgehen liegt aber nicht nur im so oft bekundeten Datenhunger der größeren Anbieter. Wenn Sie den Betrieb eines Geräts, das beim Kunden steht, so minimalistisch wie möglich aufbauen, haben Sie dort auch weniger Aufwand zu leisten. Ganz konkret reduzieren Sie die Anzahl der möglichen Fehlerquellen auf dem Gerät und verlagern diese stattdessen in Ihr eigenes Rechenzentrum, in dem Sie auf alle Daten, Protokolle und Anwendungslogiken Zugriff haben. So ist es viel leichter, Fehler zu beheben und Updates einzuspielen oder experimentelle Features an einer kleinen Benutzergruppe zu testen, als wenn Sie möglicherweise durch ein einzelnes Update eine Reihe Geräte beschädigen, sodass die Benutzer sie erst manuell zurücksetzen müssen.

Nun wechseln wir mal die Seiten. Wir werden schließlich in den nächsten Tagen zu den Entwicklern, die ebenfalls einen

Assistenten entwickeln wollen, und kommen somit nicht darum herum, Verständnis für das Speichern von Befehlen aufzubringen. Schließlich möchten wir ja wissen, wie mit unserem Assistenten interagiert wird. Die Herausforderungen dabei sind vielfältig.

Unsere Sprache, Intonation und Sprechgeschwindigkeit ist sehr divers, wir haben alleine in Deutschland schon zwanzig Dialekte und diese haben ein ganz eigenes Vokabular, das selbst ein Mensch erst über Jahre erlernen muss (ich bin als Hannoveraner in die Pfalz gezogen; ich weiß, wovon ich spreche) und selbst wenn es keine Dialekte gäbe, sind die Sprechweisen ganz anders. So muss *Wetter Landau* dasselbe Ergebnis liefern wie *Wie ist das Wetter in Landau?*, *Gummibärchen einkaufen* wie *Erinnere mich daran, Gummibärchen zu kaufen* und *Still* dasselbe wie *Stopp*, *Ruhe* oder womöglich *Schweigefuchs*. Sie sehen also, dass Daten unerlässlich sind, um das Verständnis der Intentionen der Benutzer zu verbessern und aktuell zu halten. In Kapitel 6, wenn wir kleine Modelle anlernen, die verschiedene Formulierungen von Befehlen verstehen, und Sie grübelnd vor dem PC sitzen und überlegen, wie ein Benutzer eine bestimmte Funktion aufrufen könnte, würden Sie sich eine Tabelle mit den zwanzig häufigsten Formulierungen wünschen, das verspreche ich Ihnen. Und genau das ist ein Verwendungszweck für die gesammelten Daten.

Natürlich ist das kein Freibrief und auch keine gute Begründung dafür, persönliche Gespräche mitzuschreiben, doch vielleicht zumindest eine Erklärung. Historisch ist der Umstand wohl so zu begründen, dass in den USA erst die Innovation kommt, dann die ethischen und datenschutzrechtlichen Bedenken und Regeln. In Deutschland ist es umgekehrt, wir machen erst die Regeln und schränken uns dadurch häufig sehr ein – und erwarten das

natürlich auch von den Ländern, die uns ihre Technologie liefern. Ein klassischer *Clash of Cultures*.

1.3 Für wen ist dieses Buch gedacht?

Die Frage ist gar nicht so leicht zu beantworten, denn im Prinzip ist dieses Buch für all diejenigen hilfreich, die einfache Aufgaben in ihrer Umgebung per Stimme automatisieren möchten. Mancher möchte vielleicht durch die Wetteransage geweckt werden, ein anderer möchte sein Garagentor per Sprachbefehl öffnen, eine Dritte möchte das WLAN im Auditorium abschalten, weil ihre Vorlesung beginnt.

Falls Sie die Herausforderung etwas generalistischer betrachten, ist es für Sie vielleicht auch nur interessant, den Entwicklungsprozess eines eigenen Sprachassistenten mit mir durchzugehen, die Funktionsweise zu verstehen und zu lernen, wie sich etwas Ähnliches in Python umsetzen lässt.

Vielleicht kommen Sie aber auch aus einem kleinen Start-Up und haben eine absolut einzigartige Geschäftsidee, die es erfordert, dass Sie ein kleines IoT-Device (Internet of Things) per Sprache steuern können.

Schließlich hoffe ich, dass dieses Buch auch von dem ein oder anderen zur Unterhaltung gelesen wird, denn theoretische Bücher gibt es genug und zumindest mich inspirieren technisch praktische Werke oft mehr, als solche, die nur Luftschlösser bauen. Mein Bestreben ist, besonders die herausfordernden Themen, dazu zählen ich vor allem die KI-Architekturen, so zu erklären, dass sie verständlich sind und – viel wichtiger – klar

wird, wozu diese dienen und wie sie uns in unserem ganz konkreten Fall weiterhelfen. In dem Kontext sollen auch die Neuerungen des maschinellen Lernens der letzten Jahre angesprochen werden, weswegen wir u. a. auch das Thema Quantum Machine Learning kurz betrachten und ausprobieren – es aber nicht in unseren Assistenten einfließen lassen.

Wichtig ist, dass Sie einige Dinge mitbringen. Sie sollten bereits ein wenig Programmiererfahrung haben, denn ich werde nicht bei Variablen und Schleifen beginnen, sondern die Seiten nutzen, um tiefer einzusteigen. Ich werde für alle Beispiele einen Windows-PC verwenden, jedoch ist Python dankenswerterweise in hohem Maße betriebssystemunabhängig, sodass der Assistent auch auf macOS, Ubuntu oder CentOS laufen sollte. Sollten Sie planen, das Training der TTS-Engine selber durchzuführen, benötigen Sie weiterhin eine NVIDIA-Grafikkarte mit mindestens 8 GB Speicher. Alternativ können Sie für das Training auch einen Cloud-Dienst nutzen, etwa *Google Colab*. Dieser stellt GPUs (Graphics Processing Units) oder TPUs (Tensor Processing Units) für das Training neuronaler Netze kostenlos zu Verfügung. Da dieser Teil jedoch optional ist, kann er auch getrost übersprungen und auf andere Frameworks oder meine vortrainierten Modelle zurückgegriffen werden.

1.4 Aufbau des Buches

Bisher habe ich meine Bücher immer so aufgebaut, dass ich die darin besprochenen Themen aufbereitet und sequenziell besprochen habe. Das hatte den Vorteil, dass ich pro Thema beliebig in die Tiefe gehen konnte, ohne den Faden zu verlieren –

es gab ja auch keinen. Diesmal möchte ich es etwas anders machen und mich mit Ihnen dem dann doch sehr technischen Inhalt von der fachlichen Seite nähern, sprich wir entwickeln einen Sprachassistenten und Sie lernen dabei entweder in den Hauptkapiteln oder in ergänzenden Abschnitten vertieft die Techniken kennen, die einfach anmutenden Bibliotheken zugrunde liegen. Damit haben wir ein klares Ziel vor Augen, nämlich den Assistenten fertigzustellen, und lernen aber gleichzeitig noch einiges über neuronale Netze, die maschinelle Verarbeitung von Sprache, Trainingsdaten für KI-Modelle usw.

Besagter roter Faden gestaltet sich nun so: In Kapitel 2 lernen Sie, wie man eine solide Entwicklungsumgebung in Python aufsetzt und in einer kurzen Auffrischung das Grundgerüst unserer Applikation erstellt. In Kapitel 3 widmen wir uns der Sprachsynthese, also der Wiedergabe von Sprache über den Computer. Kapitel 4 behandelt die Spracherkennung sowie die Verwendung von Aktivierungswörtern. Ebenso werden Sie verstehen lernen, wie der Fingerabdruck einer Stimme zu erkennen und für die Benutzerauthentifizierung verwendet werden kann. Darauf folgt die Interpretation von Text und das Erkennen der Intention der Sprecher in Kapitel 5. Wir werden hier weiterhin die Unterschiede zwischen regelbasierten und Machine-Learning-Ansätzen ermitteln und schauen, wann welches Verfahren zum Einsatz kommen sollte. In Kapitel 6 beginnt das Handwerk. Wir haben bereits ein robustes Framework implementiert und entwickeln nun dafür die entsprechenden Funktionen. Da Funktion ein fester Begriff aus der Programmierung ist und ich Missverständnisse vermeiden möchte, nennen wir diese ab hier *Intents*. In einigen dieser Intents werden wir das Framework unseres Sprachassistenten erweitern, etwa um eine Lautstärkeregelung, die Verwendung

von Callbacks in Intents, Dialoge über mehrere Fragen und Antworten hinweg und vieles mehr. In Kapitel 7 implementieren wir eine minimale UI, um von der Konsolenanwendung wegzukommen und Sie lernen, wie Toasts und Benachrichtigungen in Windows zu verwenden. Abschließend möchte ich in Kapitel 8 zeigen, wie eine Python-Anwendung kompiliert und paketiert werden kann, sodass Sie mit einem fertigen Installer zu Ihren Freunden und Kollegen gehen und Ihren eigenen Assistenten präsentieren können. Kapitel 9 beinhaltet dann die obligatorische, tränenreiche Verabschiedung und der Ausblick auf die Perspektiven, Studien- und Berufsfelder, denen Sie sich nach dieser Lektüre widmen können.

2 Die Entwicklungsumgebung

Eine Entwicklungsumgebung enthält prinzipiell erst einmal eine IDE (Integrated Developer Environment), die den Entwicklungsprozess durch einen Editor samt Syntaxhervorhebung, Build- und Versionsverwaltungswerkzeuge sowie die Codeanalyse unterstützt. Heutzutage verfügt jede gute IDE weiterhin über Werkzeuge zur Codeoptimierung und -Vervollständigung, sodass den Entwicklern möglichst viel Arbeit abgenommen wird. Dabei werden Typüberprüfungen automatisch durchgeführt, veraltete Pakete gesucht oder potenzielle Ausnahmefehler (*Exceptions*) aufgezeigt. Kurzum, dem Entwickler wird jegliche Arbeit abgenommen, sodass er sich bestmöglich auf die Anwendungslogik konzentrieren kann.

Zusätzlich machen uns in einigen Programmiersprachen Umgebungs- und Paketmanager das Leben leichter. Sie helfen uns, mehrere Projekte logisch voneinander zu trennen und bieten Funktionen an, um Abhängigkeiten per Mausklick oder Befehl für eine ganz bestimmte Laufzeitumgebung zu installieren, sodass wir auch hier nicht selber mühselig auf die

Suche nach den richtigen Paketen gehen und diese herunterladen müssen.

Um genau diese beiden Bestandteile wollen wir uns in diesem Kapitel kümmern. Ist das geschehen, schauen wir uns an, wie ein Environment in Python aufgesetzt wird und wie wir die Anwendungsarchitektur des Sprachassistenten gestalten wollen. Zu guter Letzt wiederholen wir kurz ein paar Basics, um ein einfaches Grundgerüst für den Assistenten aufbauen zu können.

2.1 Technologieauswahl

Lassen Sie uns kurz in die Abgründe der Glaubenskriege um die beste Programmiersprache hinabsteigen und ganz rational überlegen, warum wir für unser Projekt Python einsetzen. Schließlich hätten wir auch C#, Go, Kotlin oder Java verwenden können, oder? Wenn man bereit ist, etwas Mehraufwand in Kauf zu nehmen oder die Architektur des Sprachassistenten z. B. auf einen Microservice-basierten Ansatz (siehe Abschnitt 5.5) umstellt, ja. Da wir aber alle analytischen Funktionen in einer einzelnen Anwendung kapseln wollen, führt kaum ein Weg an Python vorbei. Die Sprache hat sich einfach zu sehr im analytischen Bereich etabliert, als dass man sie durch eine beliebige andere ersetzen könnte (ich selber komme aus der Java-Welt und musste mich auch erst mal umstellen). Es geht hier weniger um die Syntax. Vielmehr sind es die Pakete, die andere Entwickler bereitstellen und die wir für die Vielzahl von Aufgaben heranziehen, die wir während der Umsetzung bewältigen müssen. Eine Umfrage von Kaggle aus dem Jahre