Robbi, mach mir einen Kaffee!

Storyboard

Im 5 Stock des Technikum Wien haben wir einen Roboter, welcher die Mitarbeiter automatisch mit Kaffee versorgt. Die Mitarbeiter geben dabei Bestellungen über eine Benutzeroberfläche auf und der Roboter fährt zum jeweiligen Tisch, um den Kaffee zuzubereiten.

Nun wollen wir die Interaktion zwischen Nutzer und Roboter natürlicher gestalten, indem wir Bestellungen mittels Sprache ermöglichen. Ähnlich zu gängigen Sprachassistenten, wie z.B. Amazon Alexa oder dem Google Assistant, soll der CoffeeBot in der Lage sein, gesprochene Bestellungen aufzunehmen und zu verarbeiten. Aber wie kann ein Roboter gesprochene Sprache so verarbeiten, dass er sie versteht?

Zur Verarbeitung von Sprache muss diese erst aufgenommen werden. Neben einfachen Mikrofonen, welche direkt an den PC angeschlossen werden, gibt es spieziell für die Anwendung der Spracherkennung ausgelegte Hardware. Solche sogenannten Microphone Arrays bestehen aus mehreren Mikrofonen, um effektiv Befehle aus verschiedenen Richtungen wahrnehmen und Hintergrundgeräusche von der gesprochenen Sprache trennen zu können. Ein Beispiel für eine offene Platform für solche Mikrofone ist das Mic Array von ReSpeaker. Nach Aufnahme der Sprache muss diese von Audio- in Textform gebracht werden. Diesen Vorgang nennt man Speech to Text oder Spracherkennung. Letztendlich wenden wir Natural Language Processing (NLP, siehe AIAV Video Sprachverarbeitung) an. NLP hat das Ziel, mittels Statistik den Sinn von Text zu verstehen. Wenn wir den Sinn des Textes verstehen können, können wir diesen als Grundlage für Roboteraktionen verwenden.

Abbildung 1 zeigt den Ablauf, wie wir von gesprochener Sprache zu einem Befehl, welcher der Roboter verarbeiten kann, kommen. Mit Natural Language Processing haben wir uns dabei schon im AIAV Usecase *Erkläre eine Robotikkonferenz... mit AI!* beschäftigt. Also fehlt uns für das volle Verständnis der Pipeline die Übersetzung von gesprochener Sprache zu Text.

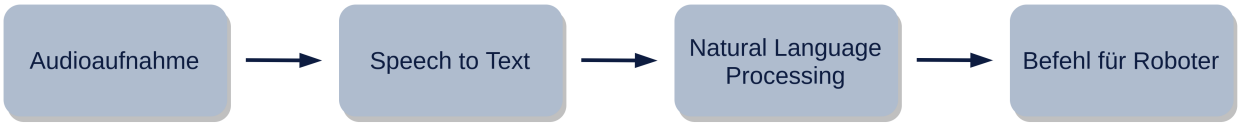

Abbildung 1: Für die Verarbeitung von gesprochener Sprache müssen wir diese zunächst aufnehmen und in Text übersetzen (Spech to Text). Anschließend wird der Text ausgewertet (Natural Language Processing) und verstanden, um einen Roboterbefehl erzeugen zu können.

Spracherkennung

Ziel der Spracherkennung ist es, gesprochene Sprache in Audioform zu Text zu übersetzen, der einen Befehl darstellt. Die Grundsätzliche Idee hinter Text to Speech ist es, das Audiosignal in kleine Ausschnitte aufzuteilen. Diese Ausschnitte werden dann mit in Frage kommenden Kombinationen von bekannten Worten verglichen, um eine Phrase zu ermitteln, welche die bestmögliche Übereinstimmung mit dem Audioausschnitt hat. Um diese Übereinstimmung zu finden, werden

Merkmale, sogenannte Features, des Audioausschnitts ermittelt. Diese Merkmale werden dann in ein Modell, z.B. einem Neuronalen Netz (siehe AIAV Video Neuronale Netze), gegeben, welches die Merkmale analysiert (siehe Abbildung 2).

Die praktische Umsetzung dieser Idee wird aber durch mehrere Faktoren erschwert. Einerseits ist es nicht möglich, alle Kombinationen aller bekannten Wörter zu prüfen, da der benötigte Rechenaufwand viel zu groß wäre. Deswegen wird neben dem Ausschnitt des Audiosignals auch noch der Kontext berücksichtigt. Dieser Kontext kann von bereits erkannten Phrasen, aber auch von manuell angegebenen Wörtern kommen. Mittels des Kontexts wird die Suche eingegrenzt, um die Spracherkennung zu beschleunigen.

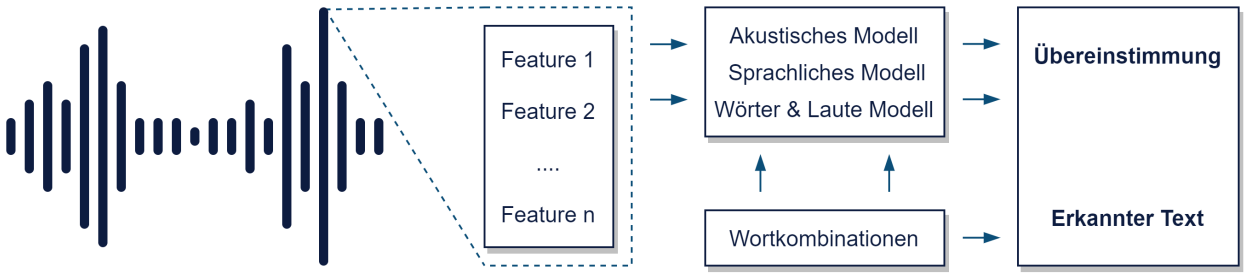

Abbildung 2: Für die Durchführung der Spracherkennung werden Ausschnitte aus der Audiaufnahme anhand bestimmter Merkmale (Features) untersucht. Dabei werden die Merkmale zusammen mit aus dem sprachlichen Kontext erzeugten Wortkombinationen in drei Modelle gegeben, um die Übereinstimmung zwischen Wortkombination und Audio zu erhalten. Die Wortkombination mit der besten Übereinstimmung wird dann als erkannter Text zurückgegeben.

Neben der Geschwindigkeit, mit der Text aus Sprache erkannt wird, ist auch ein Problem, wie die Übereinstimmung zwischen Ausschnitten der Audiodatei und Wortkombinationen überhaupt gefunden wird. Die Dokumentation von CMUSphinx, einer offene Platform zur Spracherkennung, beinhaltet eine detaillierte Beschreibung, wie diese Übereinstimmung gefunden wird. Die dort vorgestellte Lösung basiert auf drei Modellen: einem akustischen Modell, einem sprachlichen Modell und einem Modell, welches sich mit dem Zusammenhang zwischen Wörtern und ihren Lauten beschäftigt. Das akustische Modell beinhaltet dabei die Merkmale (Features) der Ausschnitte, während das sprachliche Modell aus dem Kontext vorgibt, welche Wörter abgesucht werden. Die Ergebnisse aus allen drei Modellen werden kombiniert, um jeden überprüften Ausschnitt zu bewerten und so die gesprochene Sprache zu erkennen. Das Resultat ist eine Liste von Kandidaten mit absteigender Bewertung.

Praktische Implementierung

Für die praktische Implementierung des Usecases wollen wir Sprachbefehle aufnehmen, in Text umwandeln und so auswerten, dass eine Bestellung automatisch aufgegeben werden kann. Dabei sollte es möglich sein, mehrere Kaffee auf einmal zu bestellen und zwischen den beiden möglichen Kaffeearten (Espresso und verlängerter Kaffee) zu unterscheiden.

Um Sprache aufzunehmen und in Text umzuwandeln wurde das SpeechRecognition Modul für Python verwendet. Dieses Modul bietet Verbindungen zu gängigen Schnittstellen zur Spracherkennung. Damit kann z.B. das oben erwähnte CMUSphinx verwendet werden, um lokal Spracherkennung durchzuführen. Es besteht auch die Möglichkeit, die Spracherkennung online, durch Anbieter wie z.B. Google oder Microsoft, durchführen zu lassen.

Abbildung 2 zeit den Programmablauf. Zunächst wird der Befehl aufgezeichnet und von Sprache in Text umgewandelt. Die Umwandlung erfolgt mittels der Google Webspeech API, da diese keine Installation außerhalb der Python Komponenten erfordert und ohne Registrierung gratis verwendet werden kann. Da hier die Spracherkennung in der Cloud durchgeführt wird, ist allerdings eine Internetverbindung notwendig. Nach Ermittlung des Textes führen wir Natural Language Processing, ähnlich wie im AIAV Usecase *Erkläre eine Robotik Konferenz... mit AI!*, durch. Dabei wenden wir Regular Expressions an um zu zählen, wie oft verschiedene Wörter wie z.B. Espresso oder Kaffee im Text vorkommen. Anhand der Anzahl der vorkommenden Wörter wird dann entschieden, welcher Kaffee bestellt wird. Zusätzlich zur Kaffeeart wird auf die gleiche Weise entschieden, wie viele Kaffee bestellt werden.

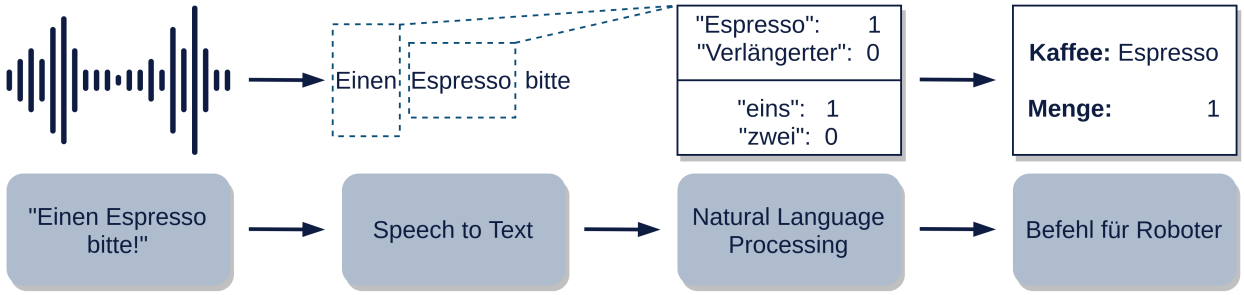

Abbildung 3: Um gesprochene Bestellungen zu verarbeiten, wird der aufgenommene Ton zunächst in Textform gebracht. Danach wird im Text gezählt, wie oft bestimmte Schlüsselwörter für die beiden Kaffeearten und Zahlen vorkommen. Die Kaffeeart die am öftesten vorkommt, wird so oft bestellt, wie die Zahl die am öftesten vorkommt.

Fazit

Spracherkennung erlaubt es uns, benutzerfreundliche Steuerungen von Robotern mittels Sprache zu realisieren. Bereits vorgefertigte Modelle wie die Google Webspeech API ermöglichen performante Lösungen, welche wenig Implementierungsaufwand benötigen. Leider sind die meisten gängigen Spracherkennungsmodelle proprietär und kostenpflichtig (z.B. Microsoft Bing Speech, Google Cloud Speech oder IBM Speech to Text). Es gibt aber auch offene Lösungen, wie CMUSphinx oder Mozilla's Project DeepSpeech die lokal ausgeführt werden.

Diese Implementierung von Natural Language Processing resultiert in einigen Grenzen der Applikation. Die Google Webspeech API unterstützt mehrere Sprachen. Die zu erkennende Sprache muss aber im Vorhinein festgelegt werden; man kann also bei diesem Use Case ohne den Code zu ändern nur auf Deutsch bestellen. Zusätzlich dazu kann pro Bestellung nur eine der möglichen Kaffeearten bestellt werden. Diese Limitierung kommt daher, dass mittels Schlagwörtern nach den Kaffeearten gesucht wird. Die Kaffeeart mit den am öftesten vorkommenden Schlagwörtern wird dann bestellt.