Computerlinguistik-Projekt: Simple audio recognition of speech commands for moonlanding game

Dennis Binz, Nathalie Elsässer, Julia Karst,
Sarah Ondraszek, Till Preidt

Computerlinguistik – Projektseminar

Sommersemester 2021
08.06.2021

Gliederung

- Datensammlung und Training des Modells
- Test-Audios erstellen
- Test der Modellprediction
- Spiel „Moonlanding“
- Audio-Input generieren
- Keyboard-Mapping
- Ausblick

Datensammlung und Training des Modells

Vorgehen:

1. Wahl des Datensatzes: Zunächst „mini speech command set“, für bessere Ergebnisse die große Version
2. Wahl und Erstellung des Scripts (siehe Artikel von Manash) für das Preprocessing der Daten und Training des Modells
3. Erstellung eines weiteren Scripts für die Pipeline – Prediction für Sample Sets
4. Zwischenstand: Modell funktioniert mit 97% Akuratesse für alle speech commands, die wir brauchen
5. Modell ist zwischengespeichert und kann nun aus dem Speicher abgerufen werden $\dashrightarrow$ nützlich für die Prediction im folgenden Teil

Datensammlung und Training des Modells

Befehlssatz:

- left
- right
- up
- down
- go
- stop
- yes
- no

Test-Audios erstellen

- Erstellen von eigenen Aufnahmen (mit Hilfe von Audacity)
- optimales Audioformat:
  - 16-bit-WAVE-Dateien
  - mono
  - Abtastrate: 16 kHz, Bitrate: 256 kBit/s

Test-Audios erstellen

- Erstellen von eigenen Aufnahmen (mit Hilfe von Audacity)
- optimales Audioformat:
  - 16-bit-WAVE-Dateien
  - mono
  - Abtastrate: 16 kHz, Bitrate: 256 kBit/s
- drei Testsets + ein Datensatz explizit nur „left“

Test-Audios erstellen

- Erstellen von eigenen Aufnahmen (mit Hilfe von Audacity)
- optimales Audioformat:
  - 16-bit-WAVE-Dateien
  - mono
  - Abtastrate: 16 kHz, Bitrate: 256 kBit/s
- drei Testsets + ein Datensatz explizit nur „left“
- **Wichtig: Audios dürfen max. 1 Sekunde lang sein**
  $\rightarrow$ Training erfolgte ebenfalls auf 1-sekündigen Dateien

Test der Modellprediction

Testen mit verschiedenen Sample Sets:

1. Mit dem Trainingsset als Input
2. Mit eigens erstellten Aufnahmen

Trainingsset funktioniert sehr gut, die eigenen Aufnahmen funktionieren auch gut, hier muss nur besonders auf die Länge der Datei geachtet werden

Spiel „Moonlanding“

Über Pfeiltasten und Sprachsteuerung spielbar:

- links/rechts: Winkel der Rakete korrigieren
- oben/unten: Schub bzw. keinen Schub geben
- Leertaste aktiviert Sprachsteuerung

Um das Spiel zu gewinnen muss man

- mit der richtigen Geschwindigkeit und
- im richtigen Neigungswinkel landen
- dem Meteor ausweichen
- auf den Treibstoff achten

Sonstiges: Verwendung eigener Bilder und Spielerauswahl

Spiel „Moonlanding“

Menüs im Spiel (chronologisch):

- Spieler-Auswahl:
  → Rakete mit Tasten auswählen
  (→ noch keine Sprachsteuerung)
- Start-Bildschirm:
  → Enter drücken / „go“ sagen, um loszulegen
- End-Bildschirm:
  → Gewonnen oder nicht und wenn nicht, warum?
  → ESC oder R drücken / „no“ oder „yes“ sagen,
  um das Spiel zu beenden oder neu zu starten
  → Neustart führt wieder in die Spieler-Auswahl

Spiel „Moonlanding“

Abbruch-Steuerung: (ESC drücken / „stop“ sagen)

- Im Spiel:
  → Spiel-Abbruch, Einblenden des End-Bildschirms
- In den Menüs:
  → Programm-Abbruch

→ im End-Bildschirm zusätzlich die Option, „no“ zu sagen

Audio-Input generieren

- Aufzeichnung nach Druck der Leertaste: wird als WAVE-Datei abgespeichert
  $\rightarrow$ 3 Sekunden, theoretisch beliebig anpassbar
  $\rightarrow$ Eigenschaften identisch mit denen der Testdaten
  (16 kHz, 16-bit, Mono)
- aus Input-Datei wird 1-sekündiger, lautester Part herausgeschnitten
  $\rightarrow$ auf diesem wird Prediction durchgeführt

Audio-Input generieren

- Aufzeichnung nach Druck der Leertaste: wird als WAVE-Datei abgespeichert
  $\rightarrow$ 3 Sekunden, theoretisch beliebig anpassbar
  $\rightarrow$ Eigenschaften identisch mit denen der Testdaten
  (16 kHz, 16-bit, Mono)
- aus Input-Datei wird 1-sekündiger, lautester Part herausgeschnitten
  $\rightarrow$ auf diesem wird Prediction durchgeführt
- Programmdauer: ca. 0,5 s

Prediction zu Tastaturbelegung umwandeln

- Importierung von pynput.keyboard
  $\rightarrow$ Simulierung von Tastendruck und Tastenlösung
- Tastenabfrage momentan nur innerhalb des Spiels mittels pygame-Events (z.B. KEYDOWN)
- Nach Leertastendruck gibt Aufnahmefunktion den Befehl als String zurück

Prediction zu Tastaturbelegung umwandeln

- Analyse des String-Befehls im Spiel
  $\rightarrow$ entweder Zugriff auf Spielfunktionen zum direkten Aufruf von Spiel-Aktionen
  $\rightarrow$ oder Rückübergabe des Befehls an input-Modul zum Simulieren eines Tastendrucks, welcher im Spiel erkannt wird
  (Tastendruck muss nach Ausführung der Spiel-Aktion wieder im input-Modul released werden)

Ausblick

- einzelne Teile des Programms zusammengesetzt
  → einzelne Module noch anpassbar
- Dokumentation zu Programmteilen erstellen
  → Erste Ansätze mit Sphinx-Autodoc, hier entstand bis jetzt eine HTML für das Preprocessing und Training des Modells
- Spiel „verbessern“
  → ggf. verschiedene Level hinzufügen & andere Details
- eventuell continuous speech
- eventuell Akuratesse weiter erhöhen