Speech To Text

In diesem Projekt sollte es ermöglicht werden, dass Nutzer ebenfalls Sprachnachrichten an den BeuthBot schicken können um mit diesem zu interagieren. Um dieses umzusetzen ist eine sogenanntes Speech-To-Text-Programm erforderlich, welche Sprachnachrichten in Text umwandelt. Diese umgewandelten Nachrichten können dann wie normale Textnachrichten vom BeuthBot weiterverabeitet werden. Da es sich hierbei um ein äußerst kompliziertes technisches Problem handelt, bei dem Ansätze mit statischen Algorithmen nicht anwendbar sind, werden ausschließlich Ansätze des DeepLearning angewendet. Neben vielen Cloud-Lösungen von namenhaften Anbietern wie Amazon und Google gibt es ebenfalls eine Reihe von OpenSource-Lösungen, welche privat gehostet werden. Dieses bietet mehrere Vorteile. Zum einen, fallen keine Gebühren für die Nutzung an, da alle Berechnungen lokal auf dem BeuthBot-Server ausgeführt werden. Zum anderen ist Datenschutz leichter umzusetzen, da alles lokal verarbeitet wird und keine Daten an externe Services weitergegeben werden. Der Recherche ergab eine Vielzahl an Lösungen, jedoch sind nur drei für das Projekt geeignet, da nur für diese ein vortrainiertes Modell für die deutschen Sprache verfügbar ist. Diese drei wurden im Zuge des Projektes getestet und werden hier kurz vorgestellt.

Kaldi

Als erstes wurde das Python-Framework „Kaldi“ betrachtet. Hierzu wurden zu Testzwecken das „kaldi_decode_wav.py“-Skript von Zamia (https://goofy.zamia.org/zamia-speech/misc/kaldi_decode_wav.py) herangezogen und mit den von Zamia zur Verfügung gestellten, tortrainierten deutschsprachigen Modellen (http://zamia-speech.org/asr/) getestet. Hierbei hat sich jedoch heraus gestellt, dass sowohl die Verarbeitungszeit als auch die Qualität der Übersetzung nicht akzeptabel ist. So dauert die Verarbeitung von WAV-Audioateien (16kHz, 16 Bit) mit einer Länge von 3 Sekunden mehr als 20 Sekunden, was bei einer Anwendung in einem Chatbot unzureichend ist. Des Weiteren war es mit wenigen Ausnahmen nicht möglich eine annähernd korrekte Übersetzung von Sprachaufnahmen zu erhalten. Der Fehlerhaftigkeit erwies sich erstaunlicher weise als sehr konstant. So wurden mehrere Aufnahmen von „Wie wird das Wetter morgen“, welche mit unterschiedlichen Betonungen und Redegeschwindigkeiten aufgenommen wurden, konstant als „Die Aussprache ist geschlossen“ übersetzt. Aufgrund dieser Faktoren wurde sich gegen Kaldi als STT-Lösung entschieden.

Wav2Letter

Wav2Letter als Lösung von FaceBook Research erschien zunächst als ideale Lösung. Zamia Audio bietet ein Modell an, welches eine WER (Word Error Rate) von lediglich 3.98% besitzen soll. Die aktive Benutzung des Frameworks erwies sich jedoch als äußerst kompliziert. Im Internet lies sich nur bedingt Dokumentation diesbezüglich finden und alle gefundene Tutorials setzten eine große Menge an Vorwissen voraus. Da die Bearbeitungszeit dieses Projekts stark begrenzt war und noch Mozilla DeepSpeech als weitere Lösung offen stand, wurde der Versuch Wav2Letter zu verwenden gestoppt.

Mozilla DeepSpeech (Mozilla Voice STT)

Mozilla DeepSpeech stellte sich im Zuge des Testens als am geeignetsten heraus. Hierbei wurde das deutschsprachige Modell (v0.9.0) von Aashish Agarwal und Torsten Zesch(https://github.com/AASHISHAG/deepspeech-german) verwendet. Sowohl die Qualität der Übersetzung als auch de Bearbeitungszeit ist im Vergleich zu Kaldi deutlich besser. Nahezu jede getestete Sprachaufnahmen wurde richtig übersetzt. Die Verarbeitungszeit beträgt ca die doppelte Länge der verarbeiteten Datei. Beispielsweise eine 3 Sekunden lange WAV-Audioateien (16kHz, 16 Bit) benötigt ca 6 Sekunden. Dieses ist nicht ideal für die Verwendung in einem Chatbot, jedoch das beste Ergebnis welches bisher erzielt werden konnte und damit zunächst akzeptabel. Dieser Prozess könnte jedoch deutlich beschleunigt werden, indem eine CUDA-Grafikkarte von NVidea im BeutBot-Server verwendet werden würde. Ein weiterer Vorteil von DeepSpeech gegenüber Kaldi ist, dass DeepSpeech eine natives NodeJS-Paket anbietet. Da bereits fast alle Services des BeuthBots auf NodeJS aufbauen, ist dieses für die Einheitlichkeit des Codes im Projekt ein nicht zu unterschätzender Vorteil. Aufgrund dieser Faktoren, wurde sich dafür entschlossen das Mozilla DeepSpeech zur Umsetzung des Speech To Text Services des BeuthBots verwendet werden soll.

Umsetzung

Als Grundlage für die Umsetzung als MicroService im BeuthBot wurde der Cat-Microservice (https://github.com/beuthbot/beuthbot/tree/master/cat_microservice) verwendet, welcher auf dem „bhtbotservice“-Framework . Zur Umsetzung von DeepSpeech wurde eines der offizielle Beispiele (https://github.com/mozilla/DeepSpeech-examples/tree/r0.9/nodejs_wav) verwendet. Aufgrund dieser beider Vorlagen, erwies sich die Umsetzung größtenteils als sehr intuitiv. Dateien werden im Request-Body als Form-Data für den Key „audio“ abgelegt. Innerhalb des Services werden die empfangenen Dateien dann mittels des Programms „Sox“ in ein für DeepSpeech kompatibles Format gebracht. Der fertige Service ermöglicht es somit, Sprachnachrichten in unterschiedlichsten Audio-Formaten (WAV, OGG und MP3) an den Service via HTTP-Request zu schicken und innerhalb einiger Sekunden eine Textversion der Sprachnachricht zu erhalten.

Request

```
curl --request POST 'localhost:3000/stt' --form 'audio=@"/test.ogg"'
```

Response

```
{
    "answer": {
        "cacheable": false,
        "history": [
            "file_upload",
            "stt"
        ],
        "content": "wie wird das wetter morgen"
    }
}
```