Aktueller Stand

Der Bot war ständig nicht erreichbar

Der Bot läuft via docker-compose in einer VM. Immer wenn der Bot nicht erreichbar war, startete er neu sobald sich jemand in die VM einloggte und war dann auch wieder erreichbar. Der Grund dafür war, dass docker-compose so konfiguriert war, dass die Container zwar neu starten sollten, aber nicht sofort wenn sie abstürzten (sondern in diesem fall dann eben beim Login durch einen docker-user).

Der Lösung bestand entsprechend in der Änderung der Configuration nach „restart-always“.
https://github.com/beuthbot/beuthbot/pull/3

Gateway funktionierte nicht ohne Telegram-ID

Bei ersten Experimenten ist aufgefallen, dass wenn eine Nachricht an das Gateway geschickt wird und diese keine valide Telegram-ID enthielt, wurde die Nachricht ignoriert. Dieses war entgegen der Dokumentation, welche die Telegram-ID als optional definierte. Da dies für Testzwecke sehr hinderlich ist und im Projektverlauf zwei weitere Messenger (Discord und eigene Webseite) hinzugefügt werden sollen, galt dieses als eines der ersten Probleme die behoben werden sollten.

Durch eine Anpassung des Gateways bei der User-Abfrage wird eine valide Telegram-ID nicht vorausgesetzt. https://github.com/beuthbot/gateway/pull/2

Continous Deployment

Continous Integration & Deployment ist ein wichtiger Pfeiler für ein stabiles Production Environment. Durch eine CI/CD Pipeline kann sichergestellt werden, dass das Deployment nachvollziehbar, zuverlässig ausgeführt wird und bietet zugleich die Möglichkeit Qualitätssicherungs-Mechanismen in der Pipeline zu manifestieren.

Zu Beginn des Semesters wurde das Deployment manuell ausgelöst. Es gab mehrere Scripte, die diesen Vorgang unterschiedlich angingen. Die Updatestrategie ist grundsätzlich „alle Repositories updaten via git pull“ - leider gab es hier allerdings flaws, die dazu führten dass das lokale Repository „unrein“ wurde und nicht automatisch aktualisiert werden konnte.

Hinzu kam das Problem, dass die Ordnerstruktur unterschiedlichen Usern gehörte, immer denjenigen, die das Update ausgeführt haben, bei dem die Dateien erstmals im Repository auftauchten. Dadurch scheiterten Updates zusätzlich, wenn „der falsche user“ das update versuchte bzw. „die falschen dateien“ im update aktualisiert würden.

Wir haben via Github-Actions eine CI/CD Pipeline erstellt, die den Deployment Prozess in 3 Stages ausführt: Build, Test, Deploy. Die Test-Stage ist via Makefile angebunden, so dass EntwicklerInnen neue Tests simpel in eine zentrale Stelle eintragen können. Das Makefile ist nun Single Point of Truth. Die teilweise widersprüchlichen Scripte von vorher wurden aufgeräumt

Code Änderungen im Repo (Pull Request): https://github.com/beuthbot/beuthbot/pull/4

Mithilfe eines Selfhosted-Runners welcher auf dem BeuthBot-Server installiert wurde, ist es jetzt möglich diesen Prozess zu automatisieren. Sobald ein Git-Commit mit einem Versions-Tag gepusht wird, wird dies vom Runner erkannt und der Deploy-Prozess wird angestoßen. Der Runner führt die Aktualisierung auf der VM des Bot durch.

Doku zur Runner Config:
https://github.com/beuthbot/beuthbot/blob/master/.documentation/github-runner.md

Public Domain

Es gab keine Public Domain zum Telegram Gateway. Diese brauchen wir aber um a) eine Landing Page für den Bot zu hosten und b) das Gateway von Chatbots ansprechen zu können, die nicht in der VM gehosted sind. Damit dies funktioniert wurde ein Proxy-Pass für die Default-Domain von **https://beuthbot.ziemers.de/** angelegt. Damit wird jetzt folgender Curl Möglich: *$ curl https://beuthbot.ziemers.de/message -X POST -H „Content-Type: application/json“ –data „{\“text\„:\“Wie wird das Wetter morgen?\„}“*

Text 2 Speech Recherche

- Say.js
  - https://www.npmjs.com/package/say
  - https://github.com/marak/say.js/
2. Web Speech API
   - https://wiki.selfhtml.org/wiki/JavaScript/Web_Speech
3. Text2Speech
   - https://www.npmjs.com/package/text-to-speech-file

Speech 2 Text Recherche

Mozilla Voice STT (DeepSpeech)

https://github.com/mozilla/DeepSpeech https://github.com/AASHISHAG/deepspeech-german

- Opensource
- Offline nutzbar
- Viel Dokumentation
- Deutsches Modell
- WER: 15%
- Zukunft ungewiss

Kaldi

https://github.com/kaldi-asr/kaldi http://kaldi-asr.org/doc/about.html

- Opensource
- Offline nutzbar
- Deutsche Modelle

- WER: 8,44%

Wav2Letter

https://github.com/facebookresearch/wav2letter

- Opensource
- Offline nutzbar
- Deutsche Modelle
- WER: 4%

Espresso

https://github.com/freewym/espresso

- Opensource
- Offline nutzbar
- Kein deutsches Modell

Nvidea OpenSeq2Seq

https://github.com/NVIDIA/OpenSeq2Seq

- Opensource
- Offline nutzbar
- Kein Deutsches Modell

WER Vergleich 2017

- Google (8%)
- Microsoft (5.9%)
- IBM (5.5%)
- Apple (5%)
- Baidu (16%)
- Hound (5%)

Quelle:
https://askwonder.com/research/current-voice-recognition-word-error-rates-google-amazon-microsoft-ibm-apple-5b88trj0t