Teamarbeit: Hello World. Ein Chatbot für Telegram und ConceptNet

Tabea Haverkamp, Kevin Choong, Johannes Rompe

Sommersemester 2019

Contents

1 Aufgabenstellung

Unsere Aufgabe war es, einen Chatbot zu erstellen, der dann an die semantische Datenbank ConceptNet angeschlossen wird. Auf dieser Basis soll dann ein Roboter Aufgaben ausführen. Die Konversationssprache ist Englisch, weil es für viele Natural Language Processing (NLP) Bibliotheken am besten ausgebaut ist. So ist die englische Grammatik einfacher als die deutsche. Der Chatbot soll auf einfache Fragen antworten können, wie *Hello, can you bring me a cup of coffee?*. Diese Informationen sollen extrahiert werden und mit ConceptNet soll der Roboter die Aufgaben ausführen. Für das Beispiel, eine Tasse Kaffee zu bringen braucht es viele Informationen, die teils aus ConceptNet und teils aus Informationen gewonnen werden, die aus dem früherem Chat-Verlauf extrahiert werden. Für das Bringen eines Kaffees, braucht es unter anderem folgende Informationen:

Information	Information Gewinn
me	Chat-Verlauf. Wo ist diese Person?
cup	ConceptNet. Was ist eine Tasse, wo stehen Tassen? Daraus resultierend Informationen wie Küche, Schrank...
coffee	ConceptNet. Was ist Kaffee, wo ist Kaffee zu finden?
coffee	Chat Verlauf. Wie mag der User den Kaffee?

Dies sind aber nur die ersten Informationen, die aus dem einfachen Befehl des "Bring mir Kaffee" extrahiert werden müssen. Des Weiteren muss irgendwo hinterlegt sein, wo die Person oder die Küche ist, damit der Roboter diese anfahren kann. Dann muss der Roboter Informationen darüber haben, in welchem Schrank die Tassen in dieser speziellen Küche stehen. Zwar kann er abstrahieren und sagen "meist stehen Tassen in einem Hängeschrank über der Spüle", aber das ist nicht immer der Fall.

Dadurch zeigt sich bereits, dass sich, von einem Roboter, ein Kaffee bringen zu lassen, keine triviale Aufgabe ist. In diesem Projekt haben wir uns mit dem Chatbot beschäftigt, sodass der User sich "unterhalten" kann, und persönliche Informationen wie der Standort des Schreibtisches oder die Kaffee-Präferenzen gespeichert werden können. In einer ersten Besprechung mit Stephan Opfer haben sich folgende Punkte heraus gegliedert, die wir erreichen wollten:

1. Text to Speech (TTS). Der Chatbot soll je nach Einstellung mit Text oder Audio-Dateien antworten
2. NLP. Mithilfe von NLP-Bibliotheken soll der Chatbot auf einfachste Aussagen und Fragen antworten können.
3. Informationsgewinn. Anhand des Chatverlaufes sollen Informationen gespeichert und eventuell an das Fachbereich-interne ConceptNet weitergeleitet werden

2 Technische Überlegung

In einem vorherigen Projekt wurde ein Ansatz mithilfe eines Telegram-Bots und einer NLP Bibliothek verfolgt. Jedoch wurden mehrere Sprachen (C++ und Python) verwendet, wodurch eine Bridge benötigt wurde. Dies haben wir als zu umständlich angesehen. Wir haben uns entschieden, alles in Python zu programmieren. Besonders die NLP Bibliotheken sind für Python, weswegen wir diese Entscheidung getroffen haben. Aus diesen Gründen haben wir einen neuen Telegram-Bot erzeugt. Mehr Überlegungen über Telegram sind in Abschnitt 2.1 zu finden.
Wir haben uns viele NLP Bibliotheken angeschaut. Von der vorherigen Gruppe wurde `spaCy` benutzt. `spaCy` ist eine große API für NLP Grundlagen und wird auch in der Wissenschaft und Industrie genutzt. Dies erschien uns für unser anfängliches Problem zu groß, da wir keine großen Analysen auf Texten laufen lassen wollten und zusätzlich zum Textverstehen auch Antworten passend zu den Sätzen erstellen wollten. Nach einer kurzen Recherche sind wir auf `NLTK` gestoßen. `NLTK` ist nicht so umfangreich wie `spaCy`, jedoch stellt auch diese Bibliothek grundlegende NLP Funktionen bereit. Zusätzliche enthält `NLTK` bereits eine Chatfunktion, die wir benutzen wolten. Weitere Erklärungen zu `NLTK` kommen in 2.2.
Zusätzlich sollte der Chatbot auch eine Sprachausgabe haben. Für dieses TTS Problem haben wir die Bibliothek `TTS` benutzt, welche auch in Python agiert. Weitere Überlegungen zu `TTS` sind in 2.3 zu finden.

2.1 Telegram

Wie schon erwähnt, kam die Idee einen Telegram-Bot als Chatgrundlage zu nehmen von dem Projekt aus dem vorherigem Semester. Mithilfe eines Telegram Bots kann man mit einer bereits bestehenden Messenger eine Chatumgebung nutzen, die open-source und sicher ist. Telegram Bots benötigen keine eigene Telefon-Nummer und können von jedem Telegram-Nutzer durch die Such-Funktion gefunden und benutzt werden. Zusätzlich können sie auch Gruppen-Chats hinzugefügt werden. Wir haben uns entschieden, das gesamte Projekt in Python zu schreiben, weswegen wir einen neuen Telegram Bot erstellt haben. Dies ist in der API von Telegram sehr gut beschrieben und bereitete uns keinerlei Schwierigkeiten. Man schreibt dafür bei Telegram dem *BotFather* eine Nachricht und folgt den Schritten. Danach erhält man eine Bot-ID, mithilfe derer man den Bot programmieren kann. Da eine der Aufgaben war, den Bot per Schrift oder Audio antworten zu lassen, haben wir noch zwei weitere Features von Telegram genutzt. Telegram unterstützt Standart-Audio Formate wie .mp3. Wenn jedoch eine mp3 Datei gesendet wird und der Nutzer diese abspielen möchte, muss er die Datei herunterladen und "Datenmüll" sammelt sich im Downloadordner. Deswegen haben wir uns für Sprachnachrichten entschieden. Sprachnachrichten haben bei Telegram eine .ogg Endung und werden nur lokal herunter geladen und nicht im Downloadordner.
Außerdem haben wir für den Bot noch ein Menü erzeugt, mit dem man ihn starten

und beenden kann, sowie die präferierte Ausgabenart einstellen kann. Schreibt man `audio` so werden die Antworten des Bots als Sprachnachrichten gesendet. Analog dazu werden bei \`text` die Antworten in Textform gesendet.

2.2 NLTK

Das Natural Language Toolkit ist eine Zusammenstellung von Bibliotheken in Python für Anwendungen der Computerlinguistik. Dabei interessierten wir uns vor allem für die Chatfunktion, welche diese Bibliothek bietet. Man kann dem Bot durch die Chatfunktion von `NLTK`, sehr schnell beibringen auf einfache Fragen zu Antworten. Allerdings muss man diese Fragen ohne Rechtschreibfehler und genau nach dem implementierten Schema stellen. Bei geringster Abweichung davon, zum Beispiel vergessenes Satzzeichen, versteht der Bot die Eingabe nicht.
Um dies zu mit Hilfe von `NLTK` zu implementieren, gibt man verschiedene Fragen, Aufforderungen oder sonstige Sätze vor, welche den gleichen Inhalt haben. Darauf folgen dann die Antworten des Bots. Auch hier kann man mehrere Antwortmöglichkeiten vorgeben. Sobald man dem Chatbot nun einen der vorne stehenden Sätze schreibt, wählt er zufällig eine der vorgegebenen Antwortmöglichkeiten aus und schickt sie in den Chat. Außerdem bietet die Chatfunktion von `NLTK` die Möglichkeit, bestimmte Wörter aus Sätzen die ihm geschickt wurden, einzulesen. Beispielsweise kann man ihm folgenden Satz schicken: "my name is Rick". Für den Namen wurde in der Implementierung ein Platzhalter verwendet, welcher in der Antwort wiederverwendet werden kann. So kann der Chatbot antworten: "Hello Rick, How are you today".
Wir haben über 20 Fragen implementiert, um auf die einfachsten Nachrichten antworten zu können. Man kann dies in der Zukunft in der Breite erweitern, aber auch ist es leicht möglich, Daten die eingelesen wurden (z.B. durch die Platzhalter) den Chat-IDs zuzuordnen und lokal zu speichern, um besser auf den Benutzer eingehen zu können.

2.3 TTS

Wie bereits zuvor erwähnt haben Nutzer die Möglichkeit über die Chat-Operationen `audio` und `text` Antworten als Text- oder Audio-Nachrichten zu erhalten. Die Audio-Option wird mithilfe von Text-to-Speech realisiert. Wir haben uns hier für gTTS entschieden, eine Python Bibliothek, welche die TTS API von Google verwendet. gTTS hatte den Vorteil, direkt Audio-Dateien für unsere Sprachnachrichten zu liefern, wobei andere Bibliotheken nur direkte Audio-Ausgabe ermöglichten. Zwar waren diese Dateien immer noch im .mp3-Format, jedoch ermöglichte die Bibliothek pydub eine einfache Konvertierung zu unserem benötigten .ogg-Format und konnten den Bot diese, als Sprachnachricht, an den Nutzer senden lassen.

3 Persönliche Rolleneinschätzungen

3.1 Tabea Haverkamp

In dem Vorbereitungsworkshop habe ich mich als *Implementor* nach Basadur gesehen. In der Team Rollen Verteilung nach Belbin habe ich mich als Vermittler zwischen kommunikationsorientierter Personen und wissensorientierter Personen gesehen. Ich habe mich als Koordinatorin bis zu Spezialistin, und Erfinderin gesehen, da ich schon Erfahrungen mit Robot Operating Systems (ROS) in früheren Kursen gesammelt habe. Meine Selbsteinschätzung hat sich nur zur Hälfte bewahrheitet. Kevin war in unserer Gruppe der Implementor. Ich habe eher nach Ideen und Ansätzen gesucht, die funktionieren könnten. Anbei habe ich versucht, unsere Gruppe voran zu bringen und überlegt, wie neue Ideen in unsere Überlegungen eingebunden werden könnten.
Ich denke, ich konnte die Ideen von uns dreien gut zusammenführen und zusammenfassen.

Meine Rolle als Spezialist konnte ich durch mein früheres Cognitive Science Studium gut einnehmen. Mit Grundlagen in (Computer) Linguistik und (semantischen) Datenbanken hatte ich schon vor Beginn des Projekts einen groben Überblick über die fachlichen Grundlagen die dem Chatbot zugrunde liegen. So hatte ich zum Beispiel schon von NLP Problemen gehört und einfachste NLP Bibliotheken programmiert.

Erfahrungen mit Projektarbeiten hatte ich vorher noch nicht wirklich sammeln können. In früheren Kursen waren Programmier-Aufgaben höchstens mit 2 Personen zu erledigen. Dementsprechend unsicher war ich, wie Projekt-Treffen ablaufen, wie man Meilensteine setzt, oder wie die Aufgabenteilung insgesamt in so einem Projekt abläuft. Unter diesen Umständen finde ich, haben wir, und auch ich im speziellen eine gute Arbeit geleistet.

Rückblickend kann ich sagen, dass ich früher mehr eine Organisatorische Rolle hätte einnehmen müssen, um ein besseres Resultat zu erhalten. So hätten wir vielleicht einen gezielten Projekt-Verlauf erreichen können.

3.2 Kevin Choong

Zum Auftakt-treffen für die Veranstaltung "Teamarbeit" wurden uns zwei Modelle zur Rolleneinschätzung in einer Teamarbeit vorgestellt. Nach dem Modell von Basadur habe ich mich persönlich größtenteils als Implementer gesehen, aber teilweise auch als Optimizer. Für das zweite Modell von Belbin fiel es mir etwas schwerer mich einzuschätzen, da das Rollenspektrum, um einiges größer und feiner war. Jedoch habe ich mich zu Beginn als Macher, Umsetzer, Erfinder und Spezialist für Gruppenprojekte gesehen. Diese Einschätzungen von meiner persönlichen Teamrolle basiert ich zwar auf alle meinen Teamarbeiten, aber sehr stark auf die Veranstaltung "Software Engineering I", welche Pflicht ist für das Informatik-Studium in Kassel.

Zum Ende der Teamarbeit nachdem wir unseren Chat-Bot fertig gestellt haben, finde ich immer noch, dass meine initiale Rolleneinschätzung zu meiner generellen Aufgabe in Teams passend war. Nach dem Basadur war ich der Implementer, welches dadurch zustande kam, dass der Programmier- bzw. der Umsetzungsteil dieses Projektes nicht überragend groß war für drei Personen, sondern der Teil einen Plan zu entwickeln, was wir genau erreichen wollen für die Veranstaltung in einer größeren Thematik. Als Optimizer sah ich mich nur leicht, da ich diese Rolle eher als Programmierer angesehen habe und mir hier dachte, dass es um optimale Lösungen geht. Jedoch ist nach Baldur diese Rolle eher für Personen, die abstrakte Ideen in Konkretes umsetzen, welches für mich nicht der Fall war. Ähnlich wie beim Modell von Basadur passen zum Ende der Teamarbeit auch beim Belbin der machende Teil zu mir. Der Erfinder bzw. die Erfinderin in der Gruppe waren hier Tabea und Johannes, wobei ich nur wenige Ideen zum Projekt eingebracht habe und stattdessen deren Pläne implementiert habe. Demnach war die Einschätzung vom Macher und Umsetzer passend, aber auch der Spezialist zum Teil aufgrund meiner Erfahrung mit Python und Linux.

Meine konkreten Aufgaben im Team waren zum Ende das Implementieren, Verwaltung des Repository und teils Struktur im Quellcode und Projekt zu erhalten. Insgesamt bin ich zufrieden mit unserer Teamarbeit, da jeder von uns seine eigene Rolle schnell gefunden hatte und wir gut als Team zusammenarbeiten konnten. Zusätzlich bin ich froh, dass meine eigene Einschätzung meines Beitrags in einer Teamarbeit nicht komplett falsch lag und ich diese in der Projektarbeit erfüllen konnte.

3.3 Johannes Rompe

Zu dem Seminar am Anfang des Kurses, habe ich mich nach dem "Basadur Profile" in der Rolle des Conceptualizers gesehen, da ich in anderen Projekten meist der Vermittler war, wenn es mehrere Ansätze zur Lösung eines Problems gegeben hat. In dieser Situation muss man das ganze von außen betrachten und den Rest des Teams überzeugen, dass der vorgeschlagene Weg ein guter ist, um das Problem möglichst effizient zu lösen. In unserer Teamarbeit allerdings, waren wir uns meistens einig bei Entscheidungen, daher wurde diese Rolle nicht sehr oft benötigt. Es gab höchstens mal Probleme in der Kommunikation, welche ich versucht habe zu lösen. Ich würde mich am Ende dieser Teamarbeit, aber immer noch als Conceptualizer sehen, da ich selbst gemerkt habe dass ich diese Rolle bei jeder Gelegenheit einnehme und somit dem Team geholfen habe, effektiv zu arbeiten.

Nach den "Belbin Teamrollen" habe ich mir selbst den Weichensteller, Teamarbeiter und Beobachter zugewiesen. Diese Rollen haben ebenfalls sehr gut zu mir gepasst, wie ich im Laufe dieser Teamarbeit feststellen sollte. Der Weichensteller passt aus den gleichen Gründen wie der Conceptualizer aus dem "Basadur Profile". Außerdem habe ich in kritischen Situationen versucht, die Wogen innerhalb des Teams zu glätten, auch wenn es nur wenige davon gab. Auch habe ich stetig versucht das Arbeitsklima innerhalb des Teams, durch kleine Aufheiterungen zu verbessern. Dadurch konnten wir uns auch an Tagen motivieren, an denen wir schlecht gestartet sind oder einfach nicht gut drauf waren. Dies macht mich zu einem Teamarbeiter, der stetig versucht das Team zusammenzuhalten und das Arbeitsklima aufzulockern, ohne dabei die Produktivität einzuschränken. Die Rolle des Beobachters steckt meiner Meinung nach auch in der des Conceptualizers. Selbst geschrieben habe ich nicht sehr viele Zeilen Code, aber ich habe immer versucht einen Überblick zu bewahren, wodurch mir konzeptionelle oder offensichtliche Fehler schneller aufgefallen sind. So konnte ich beispielsweise Kevin öfter helfen, obwohl er das meiste implementiert hat und ich dabei oft nur in der Rolle des Beobachters stand.

Im Nachhinein denke ich, dass die persönliche Rolleneinschätzung am Anfang des Projektes ziemlich gut gepasst hat, weil ich im Laufe des Studiums schon mehrere Projektarbeiten mit größeren Teams hatte. Auch ist mir aufgefallen, dass ich während der Teamtreffen öfter an diese Rollen denken musste, wodurch ich noch stärker darauf fokussiert war, die jeweilige Rolle einzunehmen.

4 Probleme und Lösungsstrategien

Unser größtes Anfangsproblem war die Zeitsuche. Durch unterschiedliche Stundenpläne waren Zeitslots, wo wir alle Zeit hatten rar gesät. Durch ein Doodle haben wir nach einigem hin und her jedoch einen wöchentlichen Termin finden können, aber auch einen weiteren Ausweichtermin, den wir eventuell in Erwägung ziehen konnten, wenn jemand mal keine Zeit hatte und wir uns trotzdem nochmal alle zusammensetzten wollten.

Das darauf folgende Problem war es, uns Ziele, so genannte Meilensteine, für unsere Projektarbeit zu setzen, da uns zu Beginn der Veranstaltung ein sehr großes Aufgabenfeld gegeben wurde, wo wir als Studenten uns selbst Gedanken machen mussten, was wir genau erreichen wollten. Hier hatten wir unterschiedliche Meinungen, wie wir den Chat-Bot gestalten, wie wir ConceptNet einbringen, was der Chat-Bot überhaupt können sollte oder sogar in welcher Programmiersprache wir ihn entwickeln würden und vieles weitere. Hier wurde uns von unserem Betreuer Stefan ein Diagramm aufgezeichnet und von dort fingen wir erst wirklich an produktiv zu werden. Wir haben uns zunächst entschlossen uns von einer anderen Implementierung eines Telegram-Bots, welcher im Fachgebiet entwickelt wurde zu lösen. Auch wenn das Fachgebiet stärker C als Programmiersprache bevorzugte, haben wir uns hier für Python entschieden, da es bessere Bibliotheken für NLP besitzte und es nicht problematisch ist zwischen Python und C Programmen zu kommunizieren über JSON-Nachrichten.

Unsere Meilensteine waren darauf ausgelegt einen Chat-Bot von Grund auf aufzubauen und haben uns dafür entschieden das ConceptNet eher als Ausblick für unsere Projektarbeit zu betrachten, statt mit der gegebenen Telegram-Bot Implementierung zu arbeiten und uns auf die Einbindung von ConceptNet zu fokussieren. Jedoch kamen wir dazu wichtige Schlüsselwörter abzuspeichern, welche zum einen als Grundstein für die Nutzung von ConceptNet benötigt wird, aber auch sinnvoll ist, wenn man die Funktionalität des Bots nicht nur beim Chatten belässt und auch tatsächlich mit dem Nutzer agieren soll.

5 Ablauf der Teamarbeit

Vor dem Workshop am Anfang des Semesters kannten wir uns noch nicht. Daher war die erste Hürde in dieser Teamarbeit sich gegenseitig kennenzulernen. Für diese Forming-Phase verwendeten wir ein Teamtreffen, weil wir der Meinung waren, dass es sich wesentlich besser arbeiten lässt, wenn man die Stärken und Schwächen jedes einzelnen Teammitgliedes kennt. Da unser Team nur aus drei Personen bestand, lernte man sich schnell kennen und konnte sich gegenseitig gut einschätzen. So konnten wir im zweiten Teamtreffen direkt mit der Storming-Phase und auch der Norming-Phase beginnen und entwickelten erste Ideen. Dabei entstanden innerhalb des Teams keine großen Reibungen und wir konnten uns immer auf eine gemeinsame Lösung einigen. Die Teamtreffen fanden bei uns wöchentlich statt, wobei wir teilweise auch zu Hause kleinere Aufgaben erledigt haben, wie zum Beispiel das Einlesen in Bibliotheken, Recherche oder ähnliches. Wenn ein Teammitglied es versäumt hat, sich diesen "Hausaufgaben" anzunehmen, gab es nachvollziehbare Gründe und konnte im Team dadurch geklärt werden, dass es die anderen kurz für ihn zusammenfassten. So blieben alle Teammitglieder immer auf dem selben Stand, wodurch jeder in der Lage war eigene Vorschläge einzubringen.
Bereits nach wenigen Wochen waren wir, durch die oben geschilderten Maßnahmen, ein eingespieltes Team. Wir erkannten schnell dass Kevin mit seiner Erfahrung im Bereich Software Engineering die Rolle des Implementers einnehmen würde. Tabea und Johannes dagegen haben für ihn recherchiert und versuchten genau zu verstehen wie die benutzten Bibliotheken funktionierten, um den Implementor des Teams zu unterstützen.
Am Ende dieser Teamarbeit waren wir mitten in der Performing-Phase, daher fiel es uns schwer zum Ende zu kommen. Innerhalb der Gruppe wurde sich darauf geeinigt den Bericht möglichst früh zu schreiben, da ein Teammitglied wegen eines Studiums im Auslands später verhindert sein würde. Auch hier gab es keine Probleme und wir konnten uns innerhalb unserer Treffen wieder gut organisieren, um den Bericht möglichst früh fertig zu stellen.

6 Fazit

Ziel dieser Teamarbeit war es, einen Chatbot zu entwickeln, der Mithilfe von einer NLP-Bibliothek auf einfachste Fragen antworten kann. Außerdem soll er je nach Einstellung mit Text oder Sprachnotiz antworten.
Diese Aufgaben haben wir innerhalb unseres Teams gelöst, allerdings hätte man das ganze umfangreicher, zum Beispiel durch mehrere Frage-/Antwort-Szenarien, implementieren können. Hierzu wäre es nötig gewesen, an zwei Geräten gleichzeitig zu programmieren. Da wir allerdings alles immer auf einem Bildschirm programmiert haben und die anderen Computer nur zum recherchieren genutzt haben, ging uns an dieser Stelle wertvolle Zeit verloren.
Weiterhin hätten wir uns deutlich mehr Meilensteine setzen sollen. Unsere einzigen Meilensteine waren NLP und TTS. An dieser Stelle wäre es ratsam gewesen, NLP in mehrere Teilziele zu unterteilen, um Aufgaben besser aufzuteilen und genauer zu erledigen.
Die Teamarbeit selbst lief jedoch reibungslos ab und unsere Treffen konnten das ganze Semester ohne Probleme stattfinden. Hierbei haben sich alle Mitglieder der Arbeitsgruppe schnell wohl gefühlt und sind während diesen Projektes zu einem Team herangewachsen.
Unser Projektarbeit hätte immer weiter gehen können, da es zu diesem Bereich sehr viele Funktionalitäten gibt, die man für einen Chat-Bot implementieren kann. So könnte man mehr Informationen, die ein Nutzer oft und gerne abfragt einbinden, wie nicht nur Zeit, sondern auch Wetter und Termine. Informationen nicht nur lokal speichern, sondern auf einem Server oder sogar aussenden und mit anderen Bots in Form von JSON-Nachrichten kommunizieren. Zusätzlich war unser Chat-Bot in der Lage Schlüsselwörter zu nehmen und diese potenziell und ConceptNet direkt anzubinden. Auch wenn ConceptNet sich lediglich gut mit der englischen Sprache widmet, hätte es hier durch dessen Implementierung zu interessanten Dialogen führen können oder auch einen Bot der einfach Fakten ausgibt.

7 Abkürzungen

NLP Natural Language Processing

NLTK Natural Language Toolkit

TTS Text to Speech

ROS Robot Operating Systems