

**ISSAM CHAREF**

**SALEH CHAABAN**

**Dankeschön**

Irgendwas hier schreiben als Dankeschön für Herr Kurpjuweit :D

Alexa Teil

**Prof. Dr. Stephan Kurpjuweit**

**| Hochschule Worms**

**WS-2021**

**Hochschule Worms**

**Alexa Skill**

🗣

**Chatbot Rasa**

🖄

Dokumentation & Einleitung für **4Free**.

Alexa

Rasa

Nach der ausführlichen Dokumentation von Alexa kommen wir zu dem zweiten Teil unserer Dokumentation „Implementation von Sprachassistent Rasa“. Hier geht es um die Konzeption und die Implementation von einem Chatbot durch die Open-Source Software Rasa. Das wird mit dem Ziel gemacht, um die Stärke und die Schwäche von Alexa und Rasa als Sprachassistenten zu vergleichen.

Dazu werden wir folgende Plan durchgehen:

**Plan:**

1. Rasa Definition
2. Vorteile von Rasa
3. Einleitung von Rasa
4. 4Free Chatbot
5. Fazit
6. **Rasa Definition**

Rasa ist eine Conversational AI Platform, welche mit dem Rasa Stack eine Open-Source Software zur Verfügung stellt, um Contextual AI Assistants und Chatbots zu erstellen. Neben dem kostenfreien Rasa Stack, gibt es außerdem die Rasa Platform. Diese setzt ebenso auf den Rasa Stack, bietet allerdings für Enterprise-Kunden zusätzliche Features, wie ein User-Interface mit Funktionalitäten wie die Training Data, Admin, Conversations und Models API. Entwickelt wird Rasa vom gleichnamigen Start-up mit Hauptsitz in Berlin, zudem gibt es über 300 Contributors, welche Rasa im Open-Source Sinn voranbringen.

In unserer Dokumentation werden wir uns mehr um Technik kümmern als Theorie, aber ist wichtig zu wissen, worum es hier genau geht.

Unser Chatbot wird mit Rasa erstellt. Hierbei ist Rasa-NLU für das Verstehen, die Klassifizierung und die Eigennamenerkennung zuständig. Diese Arbeit nutzt diese Fähigkeiten zu verstehen. Die Verwendung von Rasa-NLU ist gut durchgedacht und wurde mit guten Beispielen auf eigene Webseite dokumentiert. Diese macht die Nutzung und die Implementierung sehr einfach.

1. Vorteile von Rasa

Rasa ist ein Open-Source-Framework, das Unternehmen hat das Ziel ihren Kunden zu helfen, Konversation und Interaktion mit den Kunden oder das Publikum einfach zu erstalten und zu verbessern. Es basiert auf natürlichem Sprachverständnis, Dialogmanagement und Interaktionen.

Mit Rasa wird es einfach, Konversations-KI aufzubauen und im Laufe der Zeit zu verbessern. Rasa stellt eine Standardinfrastruktur für KI-Chatbots bereit, damit alle Entwickler und Unternehmen einen ihrer eigenen virtuellen Assistenten leicht verstehen und entwickeln können.

Nachdem wir jetzt einen Blick auf Rasa haben, werden wir hier ein paar von vielen Grunde, warum Rasa eine gute Auswahl ist.

* **Einfach zu integrieren und anzupassen:**

Rasa ist eine Open-Source-Plattform zur Entwicklung von Konversations-KI. Open-Source-Plattformen sind Software mit Quellcode, die jeder einsehen, ändern oder erweitern kann. Da es Open Source ist, können Entwickler zusätzliche Features und Funktionalitäten gemäß Ihren Anforderungen integrieren. Die Plattform ist einfach anzupassen und flexibel und kann daher an Ihre Bedürfnisse angepasst werden. Die Konversationen dienen auch als Daten für den Chatbot. Der Chatbot mit künstlicher Intelligenz verwendet diese Daten als Anweisungen oder „Eingabe“ und lernt daraus. Dann werden die zukünftigen Ergebnisse entsprechend der Eingabe modifiziert. Dies erspart die wiederholte Eingabe von Anweisungen und spart Zeit, Geld, Energie und Ressourcen.

* **Integration in bestehende Systeme:**

Ein weiterer Vorteil von Open-Source-Plattformen ist, dass sie problemlos in bestehende Systeme integriert werden können. Mit dieser Funktion können auch alle Vorteile verschiedener Backend-Systeme genutzt werden. Wenn bereits APIs oder RPAs vorhanden sind, die Integration vom entwickelten KI-Chatbot mit der Open-Source-Plattform von Rasa ist möglich. Wenn solche Technologien in Ihrer Konversations-KI verwendet werden, können verschiedene Prozesse einfach automatisiert werden.

* **Analysen und Berichten:**

Rasa bietet die Möglichkeit, verschiedene Analysen und Daten des Chatbots mit künstlicher Intelligenz zu überprüfen. Die Open-Source-Plattform ermöglicht es Ihnen auch, Berichte zu erstellen, die Ihnen helfen zu verstehen, wie der Benutzer mit dem KI-Chatbot interagieren und umgekehrt.

* **Auf vielen Plattformen ausführbar:**

Chatbot-Anwendung kann problemlos auf eigener Webseite (sehe unten) oder in der Cloud bereitstellen. Wenn man eine andere Plattform hat, ist es möglich, den Chatbot auf eigenen Platform einzusetzen. Rasa ermöglicht auch den Zugriffskontrolle zu gewähren usw..

Wenn wir hier weiterschreiben wollen, werden wir nicht zu Ende kommen. Rasa hat viele Vorteile.

1. **Einleitung von RASA**

In diesem Kapitel werden wir eine komplette Einleitung von Rasa darstellen für jeden Anfänger, der einen Chatbot mit Rasa entwickeln möchte. Das könnte für alle aktuelle und zukünftige Kommilitonen eine große Hilfe sein.

Da es in der Vorlesung meistens Probleme auf Windows gab, werden wir die Einleitung auf Windows darstellen.

Die Installation wird in folgenden Punkten unterteilen:

1. Anaconda installieren
2. Microsoft Visual C++ Redistributable
3. Arbeitsverzeichnis erstellen „MyFirstBot“
4. In der Kommandozeile
   * Cd zu unseren Arbeitsvereichnis
   * Conda create –name chatbotvenv python==3.7.6
   * Conde activate chatbotvenv
   * Conda install ujson==2.1.3
   * conda install tensorflow=2.0 python=3.7
   * Pip install rasa==1.10.0
   * Rasa init
5. **Anaconda Installation**

Anaconda ist ein freies Installationsprogramm (Open Source) zur einfachen Installation eines Linux-Betriebssystems. basiert auf einer Open-Source-Distribution für die Programmiersprachen Python und R. Um Anaconda zu installieren kann man hier direkt auf die offizielle Webseite gehen [Anaconda](https://www.anaconda.com/products/individual) Die Installation läuft ganz einfach.

1. **Microsoft Visual C++ Redistributable**

Die „[Visual C++ Redistributable](https://docs.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170)“-Versionen installieren Laufzeitkomponenten., sie dienen zur einwandfreien Ausführung von Programmen, die mit der Programmiersprache C oder C++ entwickelt wurden. Diese sind empfohlen, um Probleme zu vermeiden auszuführen.

1. **Rasa Installieren**

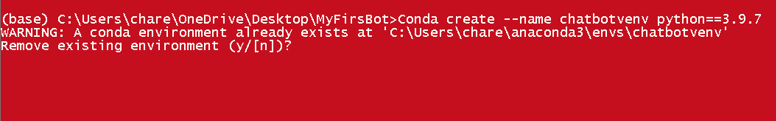
Nachdem wir die 2 Programme installiert haben, werden wir jetzt anfangen, Rasa zu installieren.

Zuerst werden wir Anaconda ausführen (als Admin) und dann werden wir ein neues Verzeichnis erstellen, wo unser Chatbot gebaut wird. In unserem Fall haben wir einen neuen Folder (MyFirstBot) im Desktop angelegt .

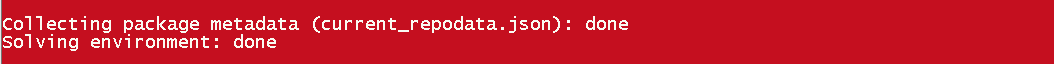


Danach werden wir unsere virtuelle Umgebung erstellen (Virtual Environment) mit dem Namen „chatbotvenv“ und das alles machen wir mit dem Kommando:

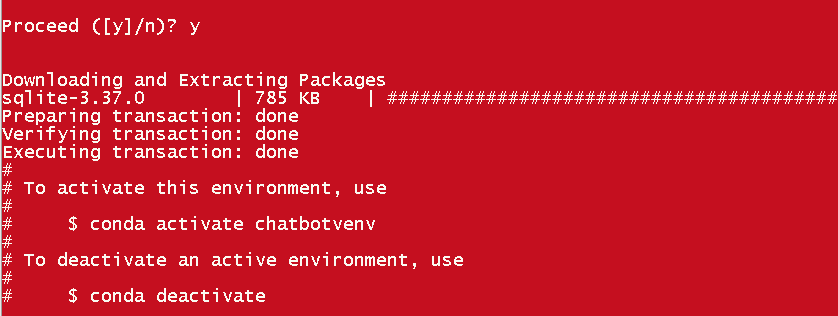
Conda create –name chatbotvenv python==3.9.7 (Python Version anpassen)



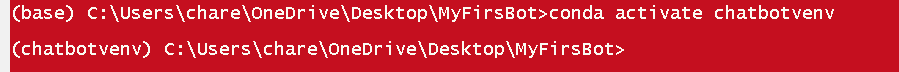
Hier fragt uns, ob er alle bereits installierte Umgebungen löschen kann (Sie können hier mit Y or N antworten, wie Sie möchten). Danach liefert uns das Terminal ein paar Informationen Bspw. Alles Package, die er installiert hat. Das kann man alles ignorieren Was für uns wichtig ist, dass er die Umgebung installiert hat und das kann mit dieser Meldung erkennen. (Wenn hier Probleme auftauchen, dann gabst Probleme bei der Installation von „Visual C++ Redistributable – ist nicht richtig installiert).



Danach machen wir einfach Y um weiterzumachen dann wird die Umgebung fertig installiert und zeigt uns ein paar Kommando, die wir in der Zukunft benutzen können. (Die Kommando finde Sie extra unserem Projekt)



Es steht jetzt klar, dass wir unsere Umgebung aktivieren sollen und das machen wir mit dem Kommando: Conda activate (chatbotvenv) – (chatbotvenv habe ich es genannt, Sie können es nennen, wie Sie möchten).



Umgebung ist aktiviert, Jetzt wird die Base Umgebung mit dem Namen unserer Umgebungsname geändert und das kann man am Anfang der Zeile sehen.

Demnächst brauchen wir noch 2 Package, die uns die Arbeit mit Rasa ermöglichen:

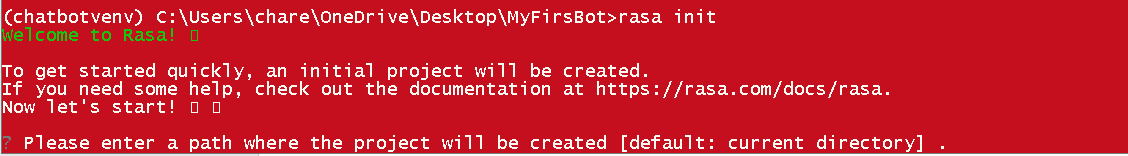
* Ujson : conda install ujson==2.1.3 : Hier die Version 2.1.3 (Installieren, um Probleme zu vermeiden)
* Tensorflow : conda install tensorflow=2.0 python=3.7

Der nächste Schritt werden wir Rasa installieren und das machen wir mit dem Kommando:



Die Version kann selbstverständlich angepasst werden oder nach der Installation ein Rasa Update machen.

Nachdem Rasa installiert wurde, wollen wir mit der Arbeit loslegen und das machen wir mit der Kommandozeile:



Welcome to Rasa! Also es hat alles gut geklappt und wir können unseren Chatbot bauen. In dem oberen Screenschot wird uns nach dem Projektverzeichnis, wo wir arbeiten möchten, da ich schon in meinem Verzeichnis bin, werde ich einfach Enter drücken.



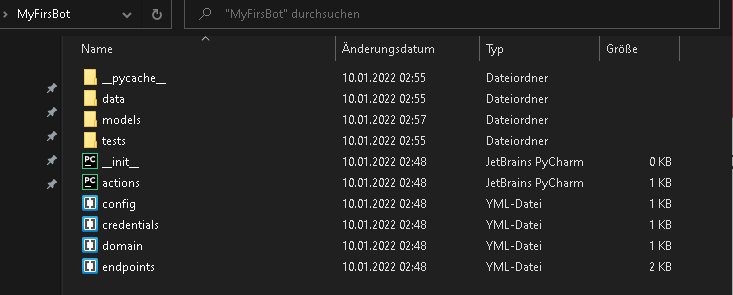
Letzte Frage, bevor wir anfangen mit der Anwendung von unseren Chatbot ist : wollen wir mit dem Kommandezeile sprechen- Yes das würde ich gern machen.

Danach unser initiale Chatbot ist bereit. Ich habe ihm ein paar Fragen gestellt, damit man sieht – Das alles funktioniert.



Wenn wir jetzt unser Verzeichnis aufmachen, wo wir Rasa installiert haben dann können wir sehen, das hier voll viele Folder und Dateien, die von Rasa installiert hat. In dem nächsten Schritt wird ein Texteditor benötigt, um unseren Chatbot Code zu editieren. In meinem Fall werde ich Atom benutzen, da ich es einfach super finde.

Wenn wir jetzt unser Verzeichnis öffnen, wo wir Rasa installiert haben dann können wir sehen, das hier ein paar Verzeichnisse und Dateien sind, die von Rasa angelegt wurden. In dem nächsten Schritt wird ein Texteditor benötigt, um unseren Chatbot Code zu editieren. In meinem Fall werde ich Atom benutzen, da ich es einfach super finde.



Das ganze Folder werden wir mit Atom oder anderen Texteditor aufmachen, um unsere Chatbot zur unseren Gebraut zu machen.

Damit sind wir fertig mit der Einleitung, jetzt kann einfach in Rasa geändert werden, um eigenen Chatbot zu entwickeln. Es bleiben uns die wichtigsten Dateien in Rasa zu beschreiben und das machen wir in unserem eigenen Projekt, um das Mehr zu geben.

1. 4 Free Chatbot

Da hier um Sprachassistenten geht und Während die Entwicklung von Alexa Skill ist uns ein paar Probleme begegnet und von Rat unserem Professor Herr Kurpjuweit sind wir auf die Idee gekommen, einen weiter Sprachassistenten zu entwickeln, der dieses Mal anderes ist, um beide Sprachassistenten zu vergleichen und schauen was, welche Vorteile und Nachteile hat.

4 Free als Alexa Skill ist vollständig und um es auf den Amazon Platform zu veröffentlichen, uns war klar dass eine Webseite mit einem Chatbot zu entwickeln würde helfen, unser Projekt zu vervollständigen.

Der Chatbot sollte als Hilfe für alle Stakeholder, die auf unsere Webseite zustoßen und Fragen haben, diese zu beantworten. Er sollte einfach benutzbar und spricht die mentale Modele unserem Kunden. Dazu was hier auch wichtig ist, die Intent Erkennung zu optimieren, damit dass für unseren Kunden als ein Mensch vorkommt und nicht als Machine.

Wir gehen jetzt alle Komponenten unserem Projekt durch und für die Optimierung kommen wir nochmal am Ende der Dokumentation dazu.

1. **Allgemeiner Überblick über die App-Struktur**

|  |  |
| --- | --- |
|  | Nach der Installation von Rasa, werden unterschiedliche Verzeichnisse und Dateien angelegt, was eigentlich ein Vorteil ist, da hier keine extra Plattform gebraucht werden, um den Chatbot im Laufe zu bringen.  Die App ist in unterschiedliche Verzeichnisse strukturiert, wie zum Beispiel einem Verzeichnis *data* für die *nlu* und *stories* (dazu mehr in den nächsten Seiten).  Die 2 Verzeichnisse Webseite und ngrok haben wir selber angelegt, um den Chatbot auf eigener Webseite laufen zu lassen. Im weiteren Verlauf werden nochmal alle wichtigen Dateien im Detail aufgegriffen, erläutern und Zusammenhänge dargestellt. |

1. **NLU-Modell (NLU.md)**

|  |  |
| --- | --- |
|  | Nachdem wir einen Blick auf der allgemeinen Struktur von Rasa geworfen haben, kommen zu einer der wichtigsten Dateien in der Entwicklung von einen Rasa Chatbot. Das NLU-Modell ist im File nlu.md (analog zum Alexa Interaction Model).  Das Ziel von NLU (Natural Language Understanding) ist es, strukturierte Informationen aus unseren Nachrichten zu extrahieren. Dies beinhaltet normalerweise die Absicht des Benutzers und alle Entitäten, die seine Nachricht enthält. Hier haben wir alle unsere benötigten Intent definiert. Zu Jedem Intent soll einen Namen gegeben haben und die verschiedenen Aussagen/Fragen die von dem erkannt werden sollen. In dem NLU-Modell können verschiedenen Elemente verwendet werden, z.B Synonyms, Entity Roles und Regular Expressions. ([Mehr Elemente](https://rasa.com/docs/rasa/training-data-format)).  Wie auf dem Bild zu sehen ist, haben wir da verschiedene Intents definiert, zu jedem Intent haben für einen Namen gegeben und die verschiedenen Aussagen oder die Fragen, die dazu geordnet sind. |

1. **Domain (Domain.yml)**

Die Domäne definiert das Universum, in dem unser Assistent tätig ist. Hier werden Absichten, Entitäten, Slots, Antworten, Formulare und Aktionen angegeben. Viele Informationen, über die unseren Bot Bescheid wissen sollte. Es definiert auch Konfiguration für die Konversationssitzung. Der Teil werden wir in 3 Abschnitten unterteilten, um die wichtigsten Elemente besser zu erläutern.

|  |  |
| --- | --- |
|  | Am Anfang des Dokuments werden alle Intent, die wir in unserem NLU Modell definiert haben, nochmal geschrieben, somit kann das Bot erkennen, dass diese überhaupt definiert sind. Der Unterschied hier von NLU Modell, dass wir keine Detaillierten Informationen dazu geben, sondern nur den Namen den Intents. |
|  | im zweiten Teil werden alle Antworten/ Responses eingegeben. Bei jeder Antwort wird einen Namen zugeordnet und den Text und/oder das Bild dazu gegeben. Achtung: hier sollte pro Antwort nur einen einzelnen Text gegeben werden. Wenn man mehrere Texte angegeben werden, wird den Chatbot bei der Ausführung laufen, aber wenn das Intent aufgerufen wird, sucht er nur einen einzelnen Text aus und gibt er das aus (meistens der erste Text). |
|  | Im dritten Teil kann entschieden werden, wie lange die Session (Setzung) im Laufen bleiben soll. (Eingabe in Minuten) |

1. **Stories (Stories.md)**

|  |  |
| --- | --- |
|  | Stories (Geschichten) sind eine Art von Trainingsdaten, die verwendet werden, um das Dialogmanagementmodell des Assistenten zu trainieren. Stories können verwendet werden, um Module zu trainieren, die auf unsichtbare Konversationspfade verallgemeinern zu können. Einfacher gesagt, hier wird die Schnittstelle zwischen den vorherigen Dateien Domain.yml und nlu.md zusammengestellt. Beispiel:   * Zu der Begrüßung haben wir 3 Intents definiert. {hallo, schlechtFühlen und gutFühlen}. Wenn einer der Elemente von Intent gutFühlen erkannt wird, sollen die 2 Antworten {utter\_gutFühlen, utter\_frageStellen} ausgegeben werden. |

1. **Konfiguration (config.yml)**

|  |  |
| --- | --- |
|  | Eins der wichtigsten Dateien in Rasa ist die Config Seite, hier werden die Piplineschlüssel, Richtlinien und Sprache definiert werden. Alles was an Konfiguration geht und was unser Modell verwendet wird hier definiert.  Die Sprach- und Piplineschlüssel geben die Komponenten an, die vom Modell verwendet werden, um NLU-Vorhersagen zu treffen. Die Rechtlinienschlüssel definiert die Rechtlinien, die vom Modell verwendet werden, um die nächste Aktion vorherzusagen. (Sobald die Konfiguration auskommentiert wird, wird eine Standardkonfiguration vorgeschlagen.) |
|  |  |

1. **Verbindung zu eigener Webseite**

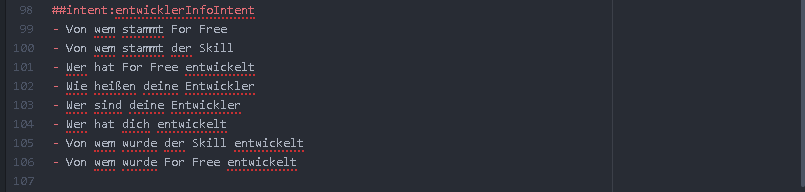
|  |  |
| --- | --- |
|  | Um unseren Chatbot auf einer unseren localhost auszuführen, haben wir hier eine Html Seite erstellt: Die Seite besteht aus ganz normalen HTML-Code und den JavaScript Code zum Einfügen von unserem Chatbot zur Webseite. *(Code im Projekt).*  Der nächste Schritt: wir werden ein paar Daten in der credentials.yml in unserem Projekt ändern.  Mit Rasa kann eine Verbindung mit verschiedenen Plattformen erstellt werden (z.B Facebook – Slack…), können wir eine Verbindung zwischen unseren Chatbot und …  Wir werden im unseren Fall Socketio benutzen.  *session\_persistence = true : um unsere Unterhaltung zu speichern und beim nächsten Mal wieder anzeigen.* |

Vergleich

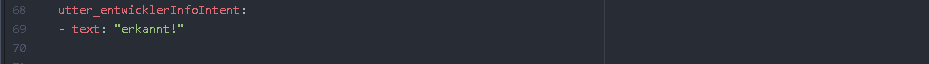
Vergleich zwischen Alexa und Rasa:

* **Intent : EntwicklerInfoIntent**

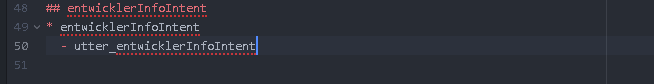
In Rasa : NLU.md

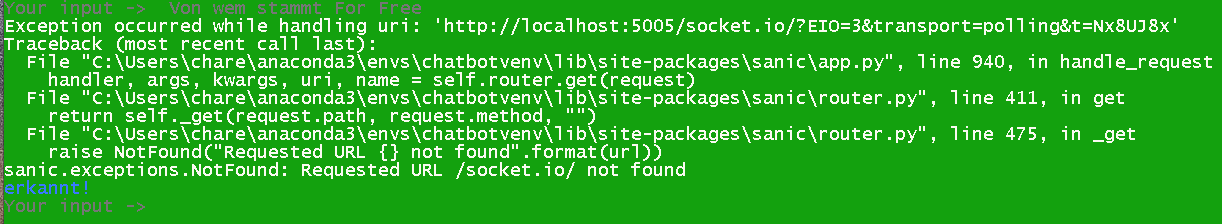


Domain.yml

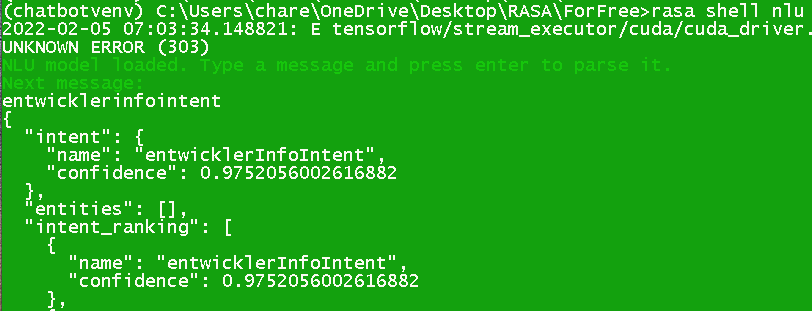


Stories.md

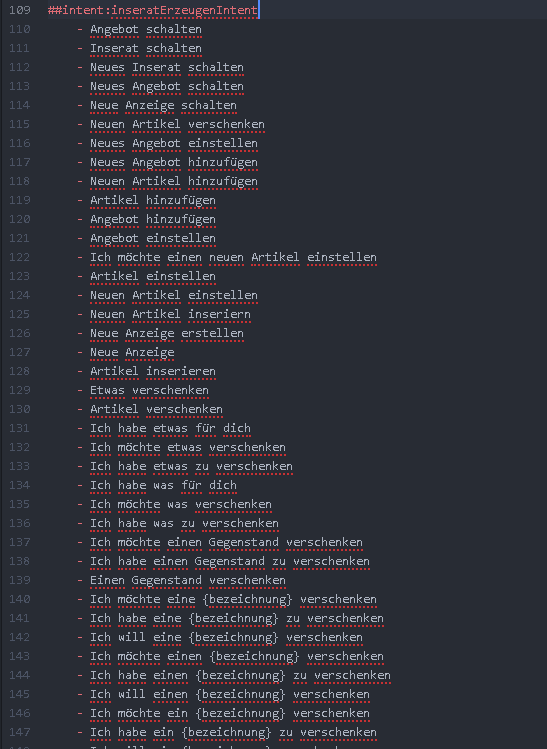


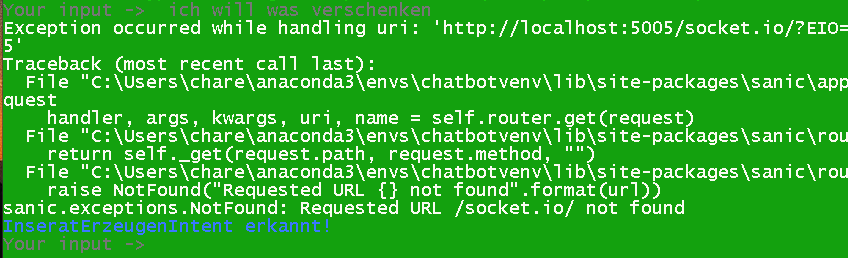


Confidence Wert:

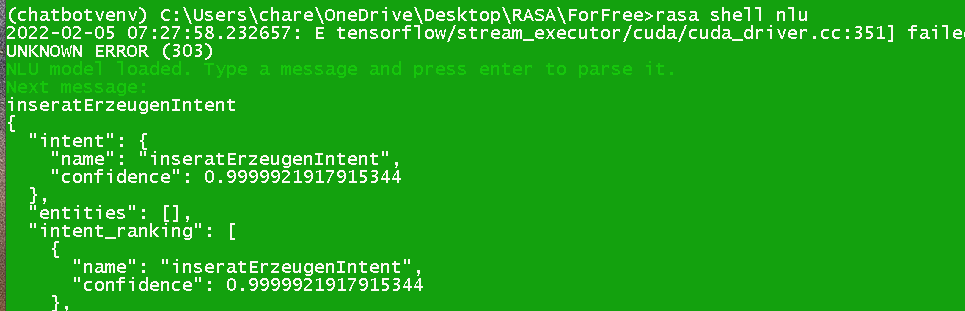


* **Intent : InseratErzeugenIntent**

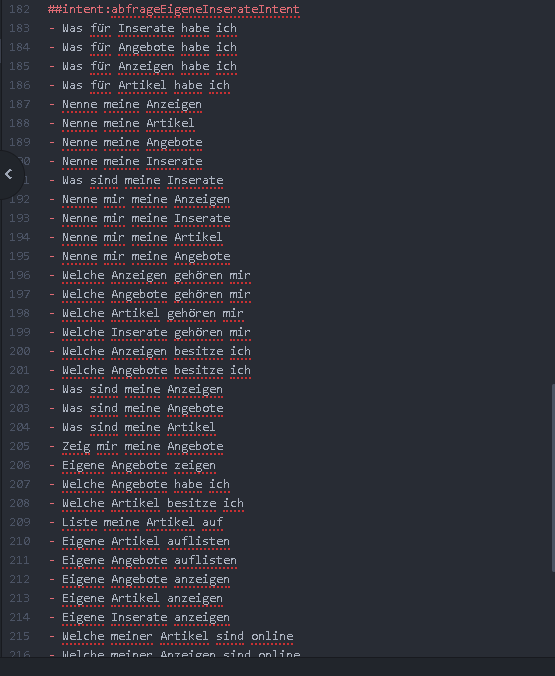


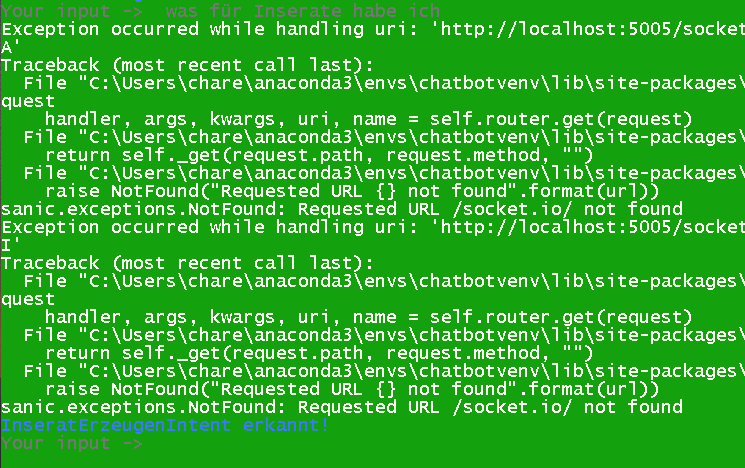


Confidence Wert :



* **Intent : AbfrageEigeneInserateIntent**





Confidence Wert :

