FOM Hochschule für Oekonomie & Management

Hochschulzentrum Siegen

Seminararbeit

im Rahmen der Lehrveranstaltung
IT-Infrastruktur

über das Thema

Dialogbasierter Zugriff auf eine KI über eine Django REST API

von

Henning Beier

Betreuer :	Daniel Bitzer
Matrikelnummer :	517370
Abgabedatum :	28. August 2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
HTTP/S	Hypertext Transfer Protocol (Secure)
KI	Künstliche Intelligenz
REST	Representational State Transfer

1 Einleitung und Zielsetzung

In dieser Seminararbeit soll eine Künstliche Intelligenz (KI)-Anwendung über eine REST[1] API[2] aufrufar gemacht werden. Eine KI-Anwendung muss dazu in einer Art eingebunden werden, die es erlaubt auf diese selbst in geeigneter Weise über die Endpunkte einer API zuzugreifen. Das bedeutet, dass die KI-Anwendung als Webservice zur Verfügung gestellt wird und damit auch einen dialogbasierten Zugriff auf die KI-Anwendung und dessen Funktionen selbst ermöglicht.

Verwendet werden dazu soll Django, ein quelloffenes Web-Framework das weit verbreitet im produktiven Einsatz ist und ebenso wie viele KI-Anwendungen in der Programmiersprache Python geschrieben wurde. Dieses zusammenführen und bereitstellen als eine Lösung im Softwarebereich, bzw. hier als Dienst, stellt dabei einen typischen Arbeitsbereich der IT-Infrastruktur dar. Die KI-Anwendung muss also in eine geeignete IT-Infrastruktur eingebunden werden. Ein sicherer und belastbarer Webservice, ist das angestrebte Ziel dieser Seminararbeit.

1.1 Praktische Relevanz und Problemstellung

Aktuell werden in vielen Bereichen neue Projekte und Lösungen mit KI entworfen, entwickelt und getestet. Dazu werden KI-Frameworks wie TensorFlow, PyTorch oder Keras benutzt. Für die vorgenannten, aber ebenso wie auch für andere KI-Frameworks, hat sich Python als Programmiersprache durchgesetzt. Dies, da sich mit Pyhton große Datenmengen vergleichsweise einfach aufnehmen, verarbeiten und ausgeben lassen. Große Datenmengen sind für die KI-Entwicklung, häufig z.B. als Trainingsdatensätze, unverzichtbar.

Entwürfe und Testentwicklungen von KI-Anwendungen erfolgen insbesondere in frühen Phasen meist lokal in einer Entwicklungsumgebung. Ergebnisse zu präsentieren oder ein Austausch mit Dritten darüber ist damit nicht immer einfach möglich. Zumeist müssten Dritte erst einen Python-Interpreter sowie umfangreiche weitere Abhängigkeiten, wie z.B. das verwendete KI-Framework selbst, installieren um die KI-Anwendung überhaupt erst Ausführen zu können. Das fällt nicht jedem interessieren leicht. Auch gibt es, durch die dynamische Entwicklung, meist verschiedenste Versionen der Software und Komponenten die zu unbeabsichtigten und unerwarteten Problemen und Hürden führen können.

[1] Representational State Transfer, API-Architekturansatz, konzipiert und entwickelt für Webservices
[2] Application Programming Interface, Programmierschnittstelle

1.2 Methodik

Die Seminararbeit soll das gewünschte Ziel, durch exploratives, vertikalen Prototyping erreichen. Vertikal bedeutet dabei, dass von der einfachen Konsolen KI-Anwendung bis hin zum dialogbasierten Aufruf über den Webserver, alle Ebenen durchdrungen werden sollen. Dabei sollen alle Schritte, jede eingesetzte Software und Lösung gründlich betrachtet und alle Schritte und Entscheidungen nachvollziehbar dokumentiert werden.

Zuerst soll eine einfache KI-Anwendung, die als Konsolenanwendung lokal in einer Entwicklungsumgebung, lauffähig ist erstellt werden. Hier ist ein üblicher Prototyp einer KI-Anwendung, z.B. aus einem einfach Tutorial, aus. Zum erreichen des Ziels, reicht es dabei aus, wenn nur wenige einfache Funktionen des verwendeten KI-Frameworks zum Einsatz kommen.

Im Anschluss daran, soll ermittelt werden auf welche Arten die Einbindung einer solchen KI-Anwendung in das Django-Framework möglich ist. Mögliche Lösungsansätze sollen ermittelt, betrachtet und bewertet werden.

Die schrittweise Einrichtung einer Django-Installation wird analog durchgeführt und alle gemachten Einstellungen, eingerichteten Erweiterungen und Anpassungen nachvollziehbar dokumentiert. Dies umfasst dabei auch die notwendigen Module für Bereitstellung der API sowie den Einsatz von Django in einem produktiven Umfeld.

Im Schlussschritt werden die Django-Installation und die, für die Einbindung vorbereitete, KI-Anwendung zusammengeführt. Der dann mögliche, dialogbasierte Zugriff auf die KI-Anwendung als Webservice, die Django REST API, soll durch den Einsatz eines dynamischen Frontends, belegt werden.

2 Theorie: Begrifflichkeiten und Grundlagen

In diesem Abschnitt werden einige für das weitere Verständnis hilfreiche Begrifflichkeiten und Grundlagen kurz erklärt und eingeordnet. Dabei wird als Zielgruppe der Seminararbeit, eine zumindest in IT-Grundlagen versierte, Leserschaft angenommen.

2.1 REST API

Die Abkürzung API bedeutet Application Programming Interface, ins Deutsche übersetzt also „Schnittstelle zur Programmierung von Anwendungen“. Eine API ist damit eine Schnittstelle zu einem Programm oder zu einer Programmierung und erlaubt es, meist anhand detaillierten Dokumentation, den Zugriff auf die bzw. die Verarbeitung von Daten aus der Schnittstelle. Im heutigen Sprachgebrauch ist mit API meist der Zugriff auf einen Webservice gemeint. Grundsätzlich aber, umfasst der Begriff API nicht nur die vorgenannten Webservices, die eine Protokollorientierte Schnittstelle sind, sondern auch API anderer Typenklassen wie Datei-, Funktions- oder Objektorientierte APIs.

Mit Representational State Transfer (REST) beschreibt Fielding erstmals im Jahr 2000 in seiner Arbeit *Architectural styles and the design of network-based software architectures* einen hybriden Architekturansatz aus der Vermischung verschiedener netzwerkbasierter Architekturansätze für Schnittstellen. Dieser Ansatz hat sich zwischenzeitlich durchgesetzt und wird vielfach in modernen Webservices verwendet. Fielding beschreibt in seiner Arbeit umfassend wie eine zustandslose Kommunikation zwischen Client und Server über eine einheitliche Schnittstelle (API) mittels bekannter und bewährter Hypertext Transfer Protocol (Secure) (HTTP/S)-Methoden, wie GET, POST, PUT und DELETE, umgesetzt wird.

Heute werden REST APIs vielfältig eingesetzt: Ob zum Datenaustausch zwischen Unternehmen oder auch der Realisierung verschiedener Frontends bzw. Endgeräte für eine Anwendung – Aus der Praxis sind Schnittstellen dieser Art heute nicht mehr wegzudenken.

2.2 KI-Anwendungen und -Frameworks

Die Konrad-Adenauer-Stiftung e. V. (Wangermann, 2020) wie auch das Bundesministerium für Bildung und Forschung, Referat Künstliche Intelligenz (2022) selbst, beschreiben

in ihren jüngsten Veröffentlichungen in ausführlicher Breite wie wichtig die Entwicklung von KI in allen Bereichen und auf allen Ebenen ist.

Dabei ist der parktische Einstieg in das Thema KI-Entwicklung im Vergleich zu anderen Forschungsfeldern ungelich leichter. Es werden, für einen Start in den Themenbereich, nicht viel mehr als ein Rechner und eine Internetverbindung bennötigt. Die Entwickler der gängigen bzw. etwa weiter verbreiteten KI-Frameworks, stellen umfangreiche Dokumentation mit die anhand praxsisnahmer Beispiele und Anleitungen, den Einstieg leicht machen sollen.

Die KI-Frameworks TensorFlow, Keras und PyTorch haben sich über die letzten Jahre nicht nur auf dem Markt gehalten, sondern wurden von der Entwicklergemeinde deutlich mehr genutzt. Besonders die positive Entwicklung von PyTorch, hier mit einem Zuwachs von 468% gemessen an Aktivitätsindikatoren auf Github.com, hebt sich hier noch weiter von den beiden anderen, vorgenannten mit jeweils etwa 50% ab.

Abbildung 1: Entwicklung der KI-Framework Nutzung auf Github.com

KI-Framework	Forks (10/2018)	Forks (08/2022)	Veränderung in %	Pull Requests (10/2018)	Pull Requests (08/2022)	Veränderung in %	Issues (10/2018)	Issues (08/2022)	Veränderung in %	Ges. (10/2018)	Ges.(08/2022)	Veränderung in %
TensorFlow	68.709	87.100	27%	8.941	21.654	142%	14.037	35.427	152%	91.687	144.181	57%
Keras	13.094	19.200	47%	3.292	5.491	67%	8.133	11.405	40%	24.519	36.096	47%
Microsoft Cognitive Toolkit	4.089	4.400	8%	458	555	21%	2.985	3.268	9%	7.532	8.223	9%
Torch	2.312	2.400	4%	509	511	0%	664	726	9%	3.485	3.637	4%
PyTorch	4.772	16.200	239%	7.282	56.052	670%	5.578	27.925	401%	17.632	100.177	468%
Caffe	15.777	19.000	20%	2.164	2.233	3%	4.401	4.781	9%	22.342	26.014	16%

Hinweise:
- Eigene Zählung am 27.08.2022 auf den in der Spalte URL genannten Seiten.
- Heise, 10/2018, Artikel "Vergleich von Machine-Learning-Frameworks", URL: https://www.heise.de/tests/Vergleich-von-Machine-Learning-Fr
- Forks, Pull Requests und Issues jeweils von Github.com, wo zutreffend Gesamtwerte (offene und geschlossene)

KI-Framework	URL
TensorFlow	https://github.com/tensorflow/tensorflow/
Keras	https://github.com/keras-team/keras
Microsoft Cognitive Toolkit	https://github.com/microsoft/CNTK
Torch	https://github.com/torch/torch7
PyTorch	https://github.com/pytorch/pytorch
Caffe	https://github.com/BVLC/caffe

Quelle: Eigene Darstellung

2.3 Django Web-Framework

Django ist ein quelloffenes und frei verfügbares Web-Framework, dass nach eigenen Angaben schnell, vollständig, sicher, skalierbar und vielseitig ist (Django Software Foundation, 2022). Untermauert wird diese eigene Behauptung von Django durch bekannte Anwendern und Nutzer wie z.B. Instagram, Pinterest und Delivery Hero (StackShare, Inc., 2022). Eine Studie bestätigt Django weiter eine große Gemeinsachft an Entwicklern, eine bedeutsame Anzahl an Fragen und Antworten bei Stackoverflow und wird auch als gute Lösung für größere Projekte betrachtet (Ghimire, 2020).

Abbildung 2: Logo des Web-Frameworks Django

Quelle: Django Software Foundation,
https://www.djangoproject.com/community/logos/

Django lässt sich dabei einfach um Funktionen und Pakete erweitern. Der Modulare Aufbau mit Datenmodellen, Sichten und Einstiegspunkten lässt zusammen mit dem Template-System und der einfachen Abstraktion für verschiedene Datenbanken sowie einer praktischen Administrationsoberfläche lässt dabei dennoch eine schnelle Entwicklung von Webseiten zu.

3 Praktischer Teil: Vom Skript zum Webservice

Die Entwicklung erfolgt in verschiedenen Phasen, die aufeinander aufbauen. Erkenntnisse aus einer Phase, bedingen darauf folgende Schritte. Ein Wechsel zwischen den Bereichen und ein wiederholtes Aufgreifen einer Teils der Lösung, ist daher nicht als Zeichen von Unstrukturierheit zu verstehen, sondern als für den Prozess des hier angebwendeten, vertikalen Protoypings schlicht notwendig.

3.1 Vorbereitung einer KI-Anwendung

Als Beispiel für bereitzustellende die KI-Anwendung wurde ein Chatbot gewählt. Eine Chatbotanwendung bietet sich an, da die Verarbeitung von menschlicher Sprache ein bedeutets Forschungsfeld innerhlab der KI ist. Bereits seit Jahrzehnten wird in diesem Berich intensiv Forschung betrieben: Von Mitte der 1960er Jahre an mit z.B. ELIZA, einer Sprachverarbeitung die im wesentlichen auf Mustererkennung abzielt und seinerzeit vorallem durch den fehlenden Zugriff auf Informationen beschränkt war (Weizenbaum, 1983). Bis hin zu jüngsten Entwicklungen die auch durch Schlagzeilen in der allegmeinen Presse Aufmerksamkeit erlangten: Microsofts Chatbot Tay funktionierte technisch, wurde aber bereits nach wenigen Tagen wieder abgeschaltet, da sich ihr Charkter in unerwünschte Richtungen entwicklte: Sie verbreitete Hassbotschaften und leugnete den Holocaust (heise online, Andreas Wilkens, 2016). Noch mehr Aufmerksamkeit erlangte jüngst der Chatbot LaMDA von Google. Dies, da einer der Entickler, zu der Überzeugung gelangte, dass der Chatbot ein Bewusstsein und sogar eine Seele entwickelt habe (heise online, Martin Holland, 2022).

Für die Seminararbeit hier, wird jedoch nur eine sehr einfache Ausführung eines Chatbots herangzogen. Grundlage dafür ist ein Tutorial in dem der Autor Loeber Grundlagen über den Aufbaus einer KI-Anwendung näher bringt (Loeber, 2020).

Speziell dieses Tutorial wurde gewählt, da es als KI-Framework PyTorch verwendet. PyTorch ist neben TensorFlow eines der aktuelle führenden KI-Frameworks (siehe auch Abbildung 1), hat jedoch für diesen Anwendungsfall den Vorteil, dass es deutlich weniger Abhängigkeiten hat und kleiner in der Grundinstallation ist.

Zuerst wurde das gesamte Tutorial von mir durchgeführt und die KI-Anwendung lokal nachgebaut und getestet. Hier bei gab es keine nennenswerten Besonderheiten. Die vom Autor verwendete KI ist in diesem Fall ein künstliches, neuronales Netz mit drei Ebenen:

Abbildung 3: Programmcode: model.py aus Chatbot Tutorial

```
1  import torch
2  import torch.nn as nn
3
4
5  class NeuralNet(nn.Module):
6      def __init__(self, input_size, hidden_size, num_classes):
7          super(NeuralNet, self).__init__()
8          self.l1 = nn.Linear(input_size, hidden_size)
9          self.l2 = nn.Linear(hidden_size, hidden_size)
10         self.l3 = nn.Linear(hidden_size, num_classes)
11         self.relu = nn.ReLU()
12
13     def forward(self, x):
14         out = self.l1(x)
15         out = self.relu(out)
16         out = self.l2(out)
17         out = self.relu(out)
18         out = self.l3(out)
19         # no activation and no softmax at the end
20         return out
```

Man erkennt hier in Zeile 14 die sichtbare Eingabeebene, folgend eine verborgene Ebene Zeiel 16 sowie Ausgabeebene in Zeile 18. Da hier eine verborgene Ebene verwendet wird, spricht man von einem „Deep Learning“-Modell. Es handelt sich bei dieser Künstliche Intelligenz um ein künstliches neuronales Netz (KNN), das mit maschinellen Lernen (ML) ein mehrschichtiges bzw. tiefes Lernen (Deep Learning) durchführt. Mehr und tiefer soll in dieser Seminararbeit jedoch nicht darauf eingegangen werden, da es den Rahmen der Seminararbeit sprengen würde und nicht zum Thema selbst gehört. Deutlich wird aber, dass schon mit einer einfachen KI-Anwendung, also hier dem Einsatz des KI-Frameworks PyTorch, direkt wesentliche und grundlegnde Anwendungsfälle und mögliche Anforderungen abgedeckt werden können. Deutlich komplexere Modelle sind hier nun also nur noch eine Frage der tatsächlichen Umsetzung und eine Umsetzung scheitert hier nicht mehr an einer fehlenden IT-Infrastruktur.

Mit Abschluss des Tutorials umfasst die Anwendung nicht nur die Datei „model.py“, sondern auch eine „train.py“ die er erlaubt, dass KI-Modell anhand der Eingabedaten in der Datei „intents.json“ zu trainieren. Die Eingabedaten sind dabei wie folgt aufgebaut:

Abbildung 4: Auszug der Eingabedatei für das KI-Modell „intents.json“

```
1  {
2    "intents": [
3      {
4      "tag"      : "Begrüßung",
5      "patterns" : ["Hi", "Hallo", "Guten Tag"],
6      "responses": ["Hallo auch!", "Schön dich zu sehen!", "Willkommen!"]
7      },
8      {
9      "tag"      : "Verabschiedung",
10     "patterns" : ["Tschüss", "Bis bald", "Bye"],
11     "responses": [
12        "Danke für das Gespräch — Tschüss!",
13        "Pass auf das die Tür zu geht. Nodda."
14     ]
15     }
16     (...)
17   ]
18 }
```

An der Struktur dieser Daten kann die Funktionsweise nachvollzogen werden: Die KI-Anwendung überprüft Eingaben und versucht diese Anhand der erlernten Muster (Zeilen 5 und 10) einem Thema (Zeilen 4 und 9) zuzuordnen. Gelingt dies mit einer bestimmten Sicherheit, werden Antworten entsprechend dem Thema (Zeilen 6 und 11-13) zurückgegeben.

Der Vollständigkeit halber verweise ich noch kurz auf den Teil der KI-Anwendung die die in dieser Eingabedatei vorhandene, natürliche Sprache verarbeitet. Bevor diese Daten für das Trainung und die Verwendung im Gespräch mit dem Chatbot verwendet werden können, müssen die mittels Algorithmen der Verarbeitung natürlicher Sprache (Natural Language Processing, NLP) für die KI aufbereitet werden. Dazu weden die Sprachdaten tokenisiert, also die Eingabe in Sätze und die Sätze in Wörter aufgeteilt. In diesem Tutorial mithilfe des Paketes „nltk“ (Natural Language Toolkit).

In einem nächsten Schritt werden diese Wörter dann auf Ihren Wortstamm zurückgeführt. Dies war eine der ersten Stellen an denen ich das Ergebnis des Tutorials angepasst habe. Ich habe ein entsprechendes Paket für die Verarbeitung deutscher Sprache gesucht und mit „HanTa“ gefunden (Wartena, 2019). Dieser Programmablauf erfolgt nicht nur beim Trainung des KI-Modells, sondern auch beim Gespräch mit Chatbot über die „chat.py“.

Weitere Anpassungen durch mich in dieser allerersten Entwicklungsstufe der KI-Anwendung direkt nach Abschluss des Tutorials waren das Hinzufügen einer Log-Funktion die alle Eingaben und Antworten in Log-Dateien (hier als CSV) abspeichert. Ebenso erzeuge ich für jedes geführte Gespräch einen eindeutigen Identifikator (hier als UUID, Universally Unique Identifier), der es erlaubt geführte Gespräche in den Log-Dateien anhand der UUID vollständig nachzuvollziehen.

Alle Einstellungen und Parameter der KI-Anwendung habe ich in der Datei „config.py“ zusammengefasst. Abschließend habe ich die für das Starten der KI-Anwendung notwendigen Abhängigkeiten und Pakete für die Verwendung mit dem Python Tool „Pipenv“ in der Datei „Pipfile“ zusammengefasst.

Den gesammten ersten Entwicklungsstand, wie er bis hier beschrieben wurde, habe ich in den Anlagen zur Seminararbeit sowie auf Github[3] im Unterorder „10-example-ai-app“ dokumentiert.

3.2 Einrichtung Django-Basissystem

3.3 Erweiterungen für API

3.4 Pyhton Module und Pakete

3.5 Zugriff mit Vue.js und Axios auf Web-Frontend

4 Fazit und Reflexion

[3] https://github.com/haenno/ai-api/tree/main/10-example-ai-app

Anhang

Anhang 1: Beispielanhang

Dieser Abschnitt dient nur dazu zu demonstrieren, wie ein Anhang aufgebaut seien kann.

Anhang 1.1: Weitere Gliederungsebene

Auch eine zweite Gliederungsebene ist möglich.

Literaturverzeichnis

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. Irvine: University of California.

Ghimire, D. (2020). Comparative study on Python web frameworks: Flask and Django.

Wangermann, T. (2020). KI in KMU: Rahmenbedingungen für den Transfer von KI-Anwendungen in kleine und mittlere Unternehmen.

Wartena, C. (2019). A probabilistic morphology model for German lemmatization. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)* (S. 40–49).

Weizenbaum, J. (1983). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *26*(1), 23–28.

Internetquellen

Bundesministerium für Bildung und Forschung, Referat Künstliche Intelligenz (2022). *Zwischenbericht zur KI-Strategie*. Verfügbar unter https://www.ki-strategie-deutschland.de/home.html?file=files/downloads/Zwischenbericht_KI-Strategie_Final.pdf&cid=806 [27. 08. 2022].

Django Software Foundation (2022). *Django overview*. Verfügbar unter https://www.djangoproject.com/start/overview/ [27. 08. 2022].

heise online, Andreas Wilkens (2016). *Microsofts Chatbot Tay nach rassistischen Entgleisungen abgeschaltet*. Verfügbar unter https://heise.de/-3151646 [16. 06. 2022].

heise online, Martin Holland (2022). *Hat Chatbot LaMDA ein Bewusstein entwickelt? Google beurlaubt Angestellten*. Verfügbar unter https://heise.de/-7138314 [16. 06. 2022].

Loeber, P. (2020). *Implementation of a Contextual Chatbot in PyTorch*. Verfügbar unter https://github.com/python-engineer/pytorch-chatbot [27. 08. 2022].

StackShare, Inc. (2022). *Django - Reviews, Pros and Cons, Companies using Django*. Verfügbar unter https://stackshare.io/django [27. 08. 2022].

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit nicht einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Wilnsdorf, 28. August 2022
(Ort, Datum)

(Eigenhändige Unterschrift)