"RobotTalkZone"

Clara Osterburg Correa
Immatrikulationsnummer: 6050790

Saskia Juergens
Immatrikulationsnummer: 6143565

Subrina Jahan
Immatrikulationsnummer: 6100956

March 2024

Domain: conversation logs / a conversation chatbot
Task: Conversation Agent

Abstract:

Human interaction with computers and devices is becoming increasingly integrated into every aspect of life. From waking up in the early morning with an alarm to late-night office work, movies, or random ebook reading, everything is now connecting humans and technologies in all aspects. People rely on Google, ChatGPT, and so on. Hence, our project is also a technology like a chatbot where users can chit chat with AI like a friend, and the chatbot will continue the conversation with different scenarios. In this paper, we have discussed how this process is done in two parts with the OpenAI API, and then we implement one of the features to observe the features and question answering references.

Introduction:

Dialogue systems are typically classified into two main types: open-domain and task-oriented. Open-domain systems aim to engage users in casual conversations, so-called chit-chat conversations. On the other hand, task-oriented systems focus on specific tasks such as finding a movie or playing a song. These two directions have traditionally been studied separately due to their distinct purposes. However, transitioning smoothly from social chatting to task-oriented dialogues is crucial for capitalizing on business opportunities, yet there is a lack of publicly available data focusing on such transitions.[1]
We want to combine these two types and build a chatbot that can lead a small talk chat and move on from topic to topic until he always ends with the same topic. This paper delves into exploring conversations starting from open-domain social chatting and gradually transitioning to task-oriented purposes. This transformation should be gentle and inconspicuous. In addition, we want some influence on the model with parameters that we implement. The currently famous NLP from OpenAI also has the skill to fulfill the task, but as a coder, you have not much influence. We have opted for a model that consists of two parts. One in which we can exert influence, a feed-forward model and the OpenAI API. This combination allows us to influence our results differently. Our idea is to approach the topic of creating a chatbot that leads a conversation on a set topic. In the first step, we want to analyze what constitutes a conversation. In the second part, we want to implement one of these features, such as asking questions back or referring to related issues. As a resume, we make a transition from chit-chat to task-oriented Dialogue.

NLP and Deep Learning:

Natural Language Processing (NLP) harnesses machine learning algorithms and statistical methods to empower computers and other devices to comprehend, generate, and recognize text and speech in human language. Its applications span translation, summarization, graphical content generation from speech, voice recognition, and more. In contemporary human-computer interaction, NLP plays a pivotal role, evident in the usage of virtual assistants like Siri, Alexa, customer service chatbots, and ChatGPT.
In our project, Robot Talk Chatbot, we leverage NLP techniques such as sentiment analysis, coreference resolution, part-of-speech tagging, and natural language generation to enable rich conversations across a diverse range of topics.

The openAI API:

The openAI API, specifically the ChatGPT model, serves as a cornerstone in our project. ChatGPT offers a versatile platform for developing dialogue systems, which are commonly classified into open-domain and task-oriented systems. Open-domain systems engage users in casual conversations, whereas task-oriented systems focus on specific tasks like finding a movie or playing a song. To achieve a smooth transition from social chatting to task-oriented dialogue we use a model consisting of two parts: a feed-forward model with adjustable parameters for exerting influence and the ChatGPT API. This combination allows us to shape results effectively and facilitate a seamless transition from chit-chat to task-oriented dialogue.

Feed forward deep learning model:

The feed forward deep learning model is a fundamental architecture used in Natural Language Processing (NLP) tasks, such as text classification, sentiment analysis, named entity recognition, and machine translation . It involves passing input data through multiple layers of neurons, with each layer applying a transformation to the input data. Diese Modelle lernen, Merkmale aus Rohdaten zu extrahieren und komplexe Muster in Sprache zu erfassen, was es ihnen ermöglicht, Aufgaben mit hoher Genauigkeit auszuführen.[2]

In the project "Robot Talk Chatbot," NLP techniques including sentiment analysis, part-of-speech tagging, and natural language generation are utilized to create a conversational chatbot capable of engaging in rich conversations on diverse topics. The openAI API provides a powerful platform for accessing state-of-the-art NLP models and services, allowing developers to leverage advanced language understanding capabilities in their applications.

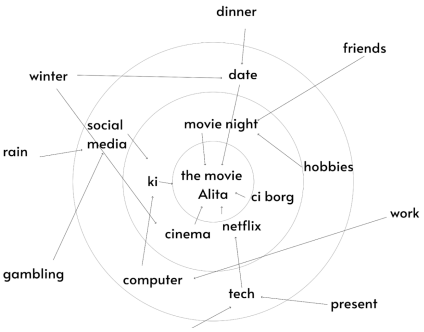
From chitchat to task orientated:

By combining openAI API with customizable feed-forward models, developers can create chatbots that adapt to user preferences and gradually guide conversations towards specific tasks. This concept not only enhances user engagement but also opens up new opportunities for personalized assistance and automation.

Our Project Concepts

Concept 01:

1. We employ a chatbot with two distinct NLP models: We utilize the ChatGPT API model, where requests are sent and responses are received.
2. We employ a feed-forward deep learning model for analyzing the current topic and guiding the conversation towards topics related to the movie "Alita". Tags are analyzed, and if they match, new tags are generated to steer the conversation closer to the topic of the movie "Alita". The training data for this task is generated using ChatGPT. The new tag is then sent to the API with a request to generate a new question related to that tag. This iterative process continues until the conversation revolves around the movie "Alita".




```
{
 "topics": [
   {
     "tag": "weather",
     "pattern": [
       "It is raining.",
       "I like cold weather",
       "It is so hot today!"
     ],
     "followingTags": ["cinema", "time"]
   },
   {
     "tag": "cinema",
     "pattern": "I like the cinema.",
     "followingTags": ["movieweather", "insideHobbies"]
   },
   {
     "tag": "movieweather",
     "pattern": [
       "i love watching movies when it is dark and cold.",
       "In wintertime i watch more movies"
     ],
     "followingTags": ["favouriteMovie", "Alita"]
   },
   {
     "tag": "Alita",
     "pattern": "I don't no the movie Alita. Tell me more!",
     "followingTags": ["done"]
   }
 ]
}
```

Advantages:

- More influence and insight into training.
- Easy additional implementation where only some questions are directed to OpenAI, allowing for energy-efficient models.

Disadvantages:

- Limited influence on how NLP works.
- Limited influence on small talk topics.

Concept 02:

To compare, we implemented a simpler version of the chatbot solely using OpenAI. The OpenAI model handles the transition from chitchat to task-oriented dialogue autonomously. We will evaluate and compare the effectiveness of both concepts.

ChatGPT API > ApiSolo.py > ...

```python
## 04.03.24
## author: Clara Osterburg Correa
## RobotTalkZone Concept 02 | simple API directly linked to openAI with specified request
## the code sends the hole conversation to openAI and a description of how to reply

import openai
import gradio

openai.api_key = "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"

## type of request
messages = [{"role": "system", "content": "you are a film critic with a select range of hobbies and intere

## chatgbt request
def CustomChatGPT(user_input):
    messages.append({"role": "user", "content": user_input})
    response = openai.ChatCompletion.create(
        model = "gpt-3.5-turbo",
        messages = messages ##all messages
    )
    ChatGPT_reply = response["choices"][0]["message"]["content"]
    messages.append({"role": "assistant", "content": ChatGPT_reply}) ##adding new message for sending all
    return ChatGPT_reply

demo = gradio.Interface(fn=CustomChatGPT, inputs = "text", outputs = "text", title = "RobotTalkZone")

demo.launch(share=True) ##launch on gradio
```

Advantages:

- Simple implementation.
- Smooth conversation flow.

Disadvantages:

- No influence on how NLP works.
- No influence on small talk topics.

Concept 03 (Not Yet Implemented, Mentioned in Text):

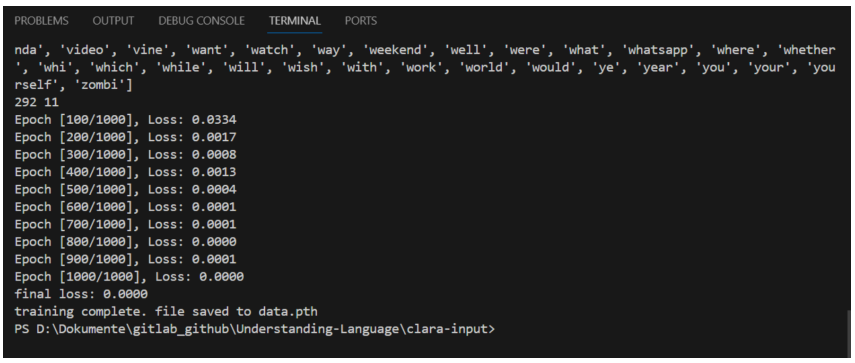
Additionally, we aim to develop a framework featuring slots that the algorithm must populate to complete the task.

```
{
  "request_slots": [
    {
      "movieKnowledge": ["no", "yes"],
      "reserve_tickets": [
        { "cinema": ["cinemax Bremen", "home"] },
        { "city": "Bremen" },
        { "time": [{ "year": "", "day": "", "clock_time": "" }] },
        { "number_of_people": "" }
      ]
    }
  ]
}
```

Chatbot Implementation:

Getting started

- Install Feed forward model PyTorch and dependencies (see official PyTorch website)
- nltk installation via terminal: pip install nltk
- install python
- get access to an Chat GBT API. Copy the API into the dokument in the folder ChatGBT API as the Varibale openai.api-key.
- Install openai via terminal: pip install openai
- Install gradio via terminal: pip install gradio
- run train.py. This will dump data.pth file. It will look like that:

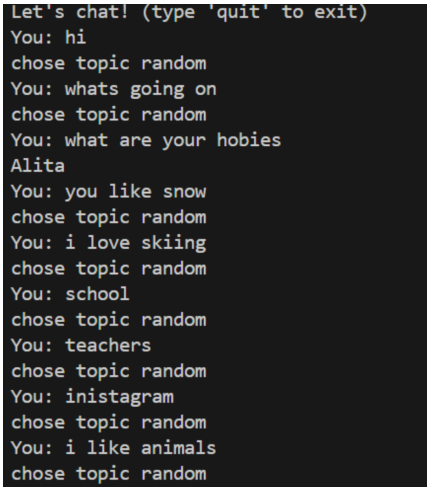


- run chat.py

We pass the category recognized by the previous model to the Chat GBT model. Subsequently, we send the category to the OpenAI API with the command to engage in small talk and transition into the next topic. We present the chat as a web interface using Gradio, where users can also provide input. When the topic of 'Alita' is introduced, the chatbot initiates the filling of JSON files with slots such as movie knowledge, cinema, city, time, and number of people. The chatbot populates these slots by asking questions.

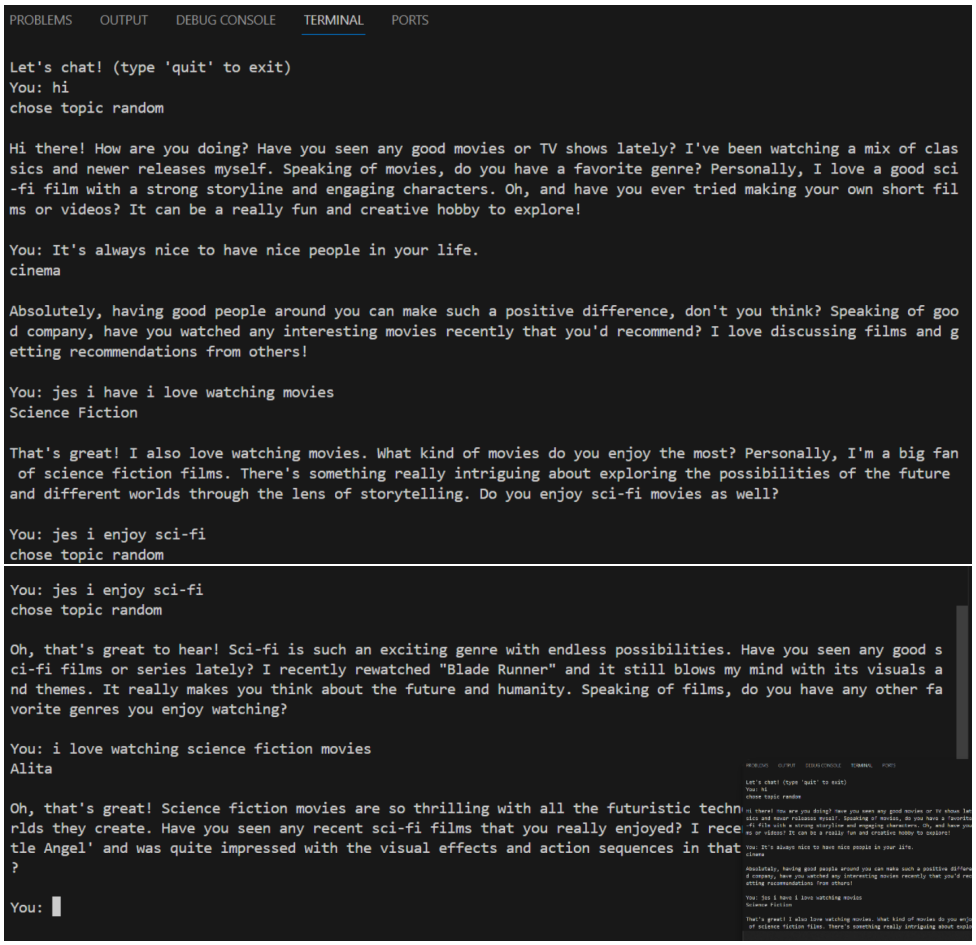
First experiment Feed forward chatbot:

Still not very trained:
In the initial attempt of Version 01, the training data for the feed-forward model is not suitable. The data needs to be manually refined and improved.

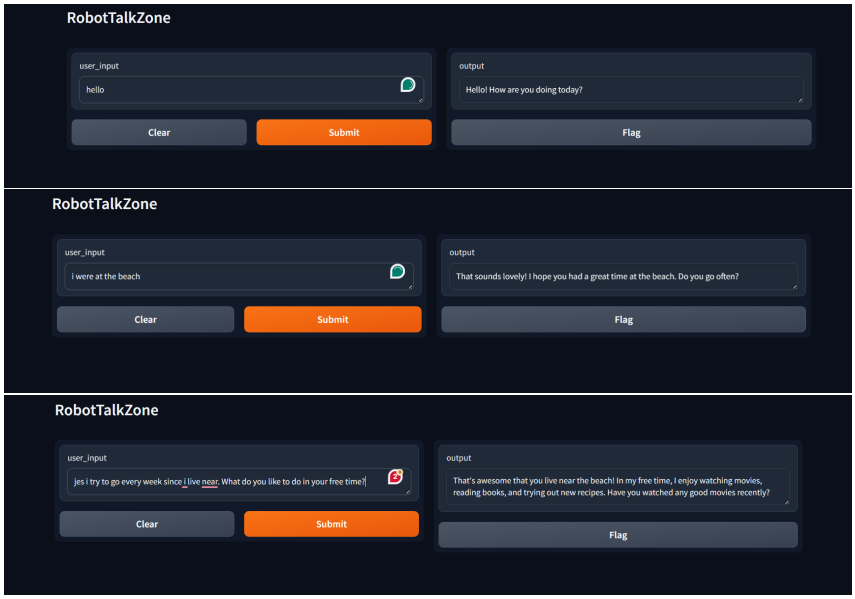


Implementation of chatgbt + feedforward:

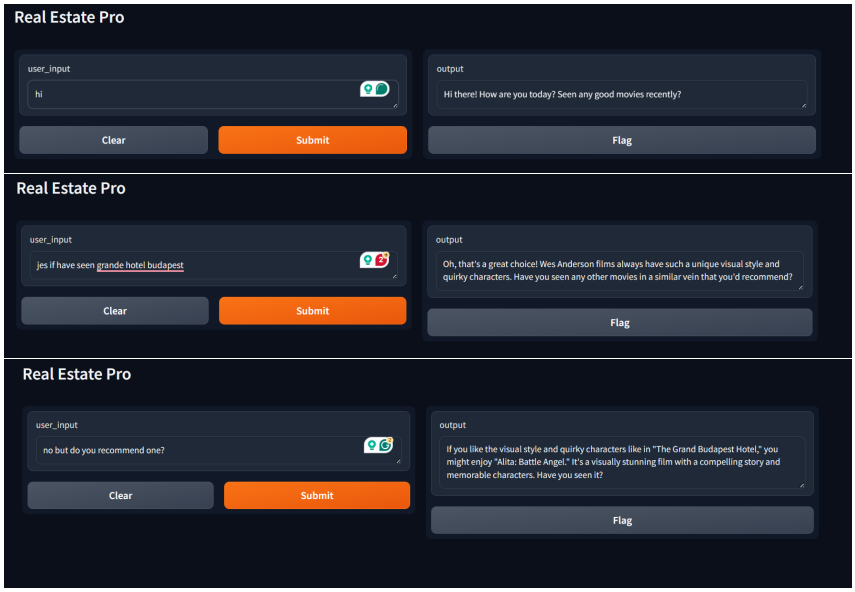
The issue lies in the fact that the training data does not adequately match the requirements of the feed-forward model, leading to difficulties in recognizing simple sentences, such as greetings. On a positive note, the implementation is functional, and the ultimate topic of conversation is "Alita."



Examples for concept 01



Examples for concept 02



Conclusion

In conclusion, our project explores the integration of open-domain social chatting and task-oriented dialogue within conversational AI systems. By leveraging advanced NLP techniques and deep learning models, we aim to create chatbots capable of guiding conversations towards specific tasks while maintaining a seamless and engaging user experience.

Through the implementation and evaluation of two main concepts, we have gained valuable insights into the strengths and limitations of each approach. Concept 01, which employs both the ChatGPT API model and a feed-forward deep learning model, offers more control and insight into training. However, it comes with limitations regarding its influence on NLP functioning and small talk topics. On the other hand, Concept 02, utilizing solely the OpenAI model, provides a simpler implementation with a smooth conversation flow but lacks control over NLP functioning and small talk topics.

Our experiments have highlighted the importance of balancing control and simplicity in designing conversational AI systems. While Concept 01 offers more flexibility in shaping dialogue outcomes, Concept 02 demonstrates the efficiency of relying solely on pre-trained models for task-oriented dialogue. The comparison between these concepts underscores the trade-offs between control and simplicity, emphasizing the need for tailored approaches based on specific use cases and user preferences.

Moving forward, we envision further refinement of our models and exploration of additional concepts, such as the development of frameworks featuring task-oriented slots for dialogue completion. By continually innovating and refining our approaches, we aim to contribute to the advancement of conversational AI and its integration into various domains, ultimately enhancing user experiences and facilitating seamless human-computer interaction.

References

[1] Ssu Chiu et al. "SalesBot: Transitioning from Chit-Chat to Task-Oriented Dialogues". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*. May 2022, p. 425. URL: https://www.aclweb.org/anthology/2022.acl-long.425.pdf.

[2] gk. "Contextual Chatbots with Tensorflow". In: *Chatbots Magazine* (May 2017). URL: https://chatbotsmagazine.com/contextual-chat-bots-with-tensor-flow-4391749d0077.