Praktikumsleistung (PrL)

Prof. Dr. Andreas Wölfl   (andreas.woelfl@th-deg.de)

Internet Technologies
"'Chatbot Project"'

Complete **Tasks 1-2** in order to pass the PrL (Praktikumsleistung).

Note the following submission rules:
- Groups of up to 4 students can make a joint submission.
- The submission must include:
    - The names and student IDs of all group members.
    - All source code and configuration files.
    - A setup manual that documents offline installation and usage.
- The submission must not include:
    - Dependencies such as the `node_modules` directory. Instead, provide a `package.json` and `package-lock.json` in the source code.
    - Microsoft Azure account credentials.
    - Any secrets, such as API keys or access tokens.
- Submit the solution in the form of a ZIP archive via iLearn.

Task 1 - Chatbot Implementation

Implement a chatbot, i.e., a computer program that simulates human conversation through text chats.

The following **functional** requirements shall apply:
- The bot should be able to understand and interact within conversations, conducted in Q&A[1] fashion, and limited to the topic assigned to you.
    - The bot should steer the conversion by asking aggressively.
    - The bot should understand the intents from a user's answer (utterance).
    - The bot should have a soft fallback if it cannot understand an user's utterance (i.e., ask the user to rephrase).
    - The bot should have a hard fallback if the bot failed to advance the conversation repeatedly (i.e., start all over again).

---

[1]Question and answer

- The bot should accomplish certain tasks in context of the chosen topic and therefore remember the conversation history. Examples:
    - Place an order in a restaurant.
    - Compile a workout plan in a gym.
- The bot should be able to conduct a conversation that exceeds 20 Q&A turns.
- The bot should not repeat itself (except for fallback questions).
- The user interface should have a modern look & feel supporting responsive web design for mobile devices, tablets, and desktop computers..
- The user interface should display questions and answers alternating in a list-like shape.

The following **non-functional** requirements shall apply:
- The frontend should be implemented in the form of a HTML5 website.
- The frontend should have a layout realized with CSS.
- The frontend should be based on the React framework having at least 4 React components.
- The frontend should communicate with the backend using Socket.IO and the Websocket protocol.
- The backend should be based on Node.js[2] and use Express.js to serve the application's user interface.
- The backend should identify the intents of a user based on keyword-spotting[3].
- The backend should be extensible in a sense that adding new topics / intents / answers does not require changing the source code.

The following rules for **code reuse** shall apply:
- It is allowed to reuse existing layouts, e.g., from *bootsnip*[4] or from *onaircode*[5], except the chat layout by Pavel Komiagin[6].
- It is allowed to reuse the code examples from the lecture, uploaded to iLearn, except the chat layout by Pavel Komiagin.
- It is **not** allowed to reuse code from any other sources (incl. ChatGPT).

Hints:
- In order to make the user interface modern looking and responsive, use a framework such as Bootstrap 5.
- Note the quality of your code will be evaluated as well → write clean code.

---

[2] Version 18.14.2 LTS
[3] The identification of keywords in text and/or utterances
[4] https://bootsnipp.com/tags/chat
[5] https://onaircode.com/html-css-chat-box-examples
[6] https://bootsnipp.com/snippets/ZlkBn

Task 2 - Chatbot Deployment

Deploy the chatbot implementation to the Microsoft Azure[7] cloud.

The following **functional** requirements shall apply:
- The chatbot should be available via a publicly available URL.
- All communication should be encrypted via SSL/TLS.

The following **non-functional** requirements shall apply:
- Enroll for a "Azure for Students"[8] subscription to get 100$ of credits.
- To host the chatbot, use one of the following Azure services:
  - Azure App Services (preferred)
  - Azure Virtual Machines

Task 3 - Chatbot Engine (optional)

Improve your chatbot by replacing the keyword-spotting approach by an implementation based on the Microsoft Azure's *Language Understanding* (LUIS[9]) service or the APIs provided by OpenAI[10] to detect the intent from a user's utterance. The same functional and non-functional requirements apply as for Task 1, except for keyword spotting.

Submit a fully operational solution in addition to (not 'instead of') the solution of Task 1 to iLearn. However, for Task 2, deploy the solution of Task 3 instead of (not 'in addition to') Task 1.

Note: Successful completion of this task carries a 10% bonus for the exam.

---

[7]https://azure.com
[8]https://azure.microsoft.com/en-us/free/students/
[9]https://luis.ai
[10]https://platform.openai.com/