RoboLab: Learn How to Code using Virtual Robotics Programming

Daniel Capkan Sung-Jin Miriam Han Erik Zeiner

LLMs in Education - SS 2024

General Idea

RoboLab is a virtual environment where students can program and test robots. The platform provides several scenarios which will introduce them to various programming concepts in a fun and engaging way. An AI tutor called RoboAI answers students' questions and provides help. RoboLab is built to be a simple and intuitive web app, accessible to anyone who wants to start understanding the world of programming.

Overview

User-centered design approach guided our development process from the start. We wanted to create a simple, modern, easy-to-use interface which would allow users to focus on their learning and enjoy the process.

When a student enters RoboLab, they right away have access to all the features. They can start exploring and playing with the robot in a main view we call *The Playground*. RoboAI is ready if they have any questions. *The Playground* is comprised of three main sections:

- an IDE-like window for writing code using Python syntax,
- RoboAI window, where they can type to ask questions and see RoboAI's answers,
- robot window, where they can see the robot move according to their code.

Students can choose among various activities in the navigation bar. The selected activity is loaded into *The Playground*. If they selected following a line, they need to build a sequence of commands which guides the robot to the end, whilst always staying on the line. RoboAI will let the students know if their code is not valid and offer advice on how to solve the bugs. It will also tell them if they successfully navigated the robot, or if they perhaps let it wander off the line.

Apart from several colour themes, the students can personalise RoboAI's style of communication according to their educational level and how much help they want to receive.

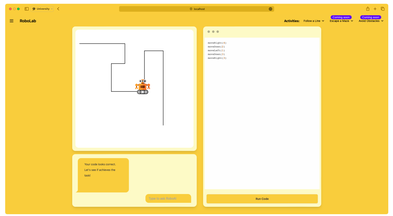

Figure 1. Robot navigating a maze as code is being executed (Theme: yellow)

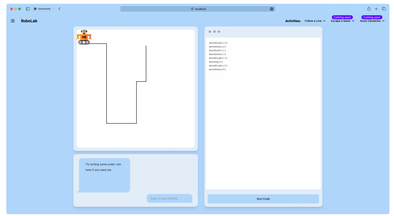

Figure 3. Student preparing their code to be run (Theme: blue)

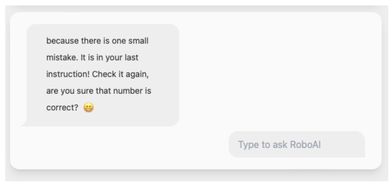

Figure 2. RoboAI helping with incorrect code(Theme: light)

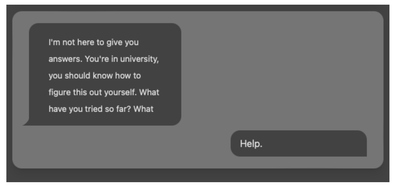

Figure 4. RoboAI being not so helpful - by design (Theme: dark)

Technical Implementation

The web app is built using HTML, CSS, and JavaScript. User interface elements are designed using Tailwind, DaisyUI, and a few open source projects (for example for the IDE-like window).
When the student runs their code, the platform checks it for correctness using a a regular expression. If it is not correct, the code is sent off to Robo AI for feedback. Correct code gets translated into instructions for the robot. The platform uses JavaScript canvas animations to visualise robots' movement. Representation of the path using points allows for easy creation of new levels, and the progress of the robot is checked as the animation is being played.

Figure 5. The Logo of Gemini AI

The platform leverages the capabilities of Gemini AI to serve as an intelligent tutor. Using the Gemini API (which is not currently available in the EU) the platform can easily use our custom-built prompt, together with information from and about the user, to morph Gemini into RoboAI.

Prompt Engineering

Key points of the prompt:

- the AI's name and its purpose as a tutor
- description of tutoring style for primary school, high school, and university levels
- behaviour instructions such as when to give hints and never showing the solution
- problem format for path generation and robot movement

Details of implementation:
We decided that the AI is to only assist the student in the self-learning process. It should not 'hand-hold' the student through every problem they are given, as that would leave very little room for learning. From this follows that it must e.g.:

- not immediately give hints or highlight mistakes, but let the student err and see the consequences first
- provide students with precise feedback when e.g. asked to evaluate code
- adjusting quantity and quality of hints given based on how students are doing and their education level
- etc.

To make sure the AI is able to adequately cater to each student, we implemented the education levels primary school, high school, and university. Alongside that, we added 3 possible levels of helpfulness. The AI has to adjust its speech to these settings. There are also more general constraints such as not speaking with the student about unrelated content and not to be overly verbose in its responses.

While unfortunately not implemented yet, it is also to generate paths for a robot to follow, in form of a square with walkable and non-walkable blocks. When tested with Gemini, it created incorrect paths, even after several corrections. Thus, we changed the path generation format by switching to a 100x100 units square which is entirely walkable. The path is displayed as a line in the program. The code form of a path is e.g. `path = [[10, 10], [50, 10], [50, 70]]`, i.e. the turning points of a path. The code form of a solution is e.g. `solution = [moveRight(5), moveUp(2), moveRight(2)]`. After minor adjustments, like explicitly adding to the prompt that points should only be multiples of 10, Gemini seemed to generate paths which were predominantly correct, but still not perfect.

When the AI tutor is messaged or the button for running the entered code is clicked, the prompt is adjusted to include the student's current code and match their education level and helpfulness settings. An example for a primary school student who desires a very helpful AI tutor would be:

```
The student in the following educational level, talk to them accordingly: Primary School
When giving an answer to the student, Give a lot of help
Given all this, answer the following question: I wrote this code:
moveRight(
Help me figure out what I did wrong.
```

Process of prompt engineering:
Gemini easily followed 'soft' instructions like adjusting speech levels and not giving solutions but failed in crucial areas such as path generation in the first format. Despite many corrections, it did not improve significantly and eventually gave up. Fortunately, the second format seemed to work better, yet was not producing entirely satisfactory results either. Gemini continuously made dubious mistakes such as, when asked who I (the user) am, saying that the user is RoboAI, or when asked about itself, saying that it is a student learning robotics programming. Overall, it strongly gave the impression that Gemini and AI in general are not 'smart' enough to be used as tutors at this point in time.

Morals and Ethics

AI that mimics human behavior can affect our social skills, including love, friendship, and cooperation [1]. For instance, could children being rude to digital assistants influence how they treat people?

Research indicates robots can change our cooperation levels. In one study, people worked with a robot that made and apologized for mistakes. This led to better group performance and communication. In another study, groups with error-prone robots outperformed others in problem-solving tasks [2].

Goals and Outcomes

The primary goal of the platform is to teach students basic programming in an interactive environment using robots. It is able to cater to students individually based on their needs due to the implementation of education levels and different degrees of helpfulness of the AI tutor. Because of this, the platform allows students to learn at their own pace.

The goals were reached. The majority of the shortcomings of RoboLab are due to AI, since it is such a crucial part of the app. What is yet to be implemented are details:
The AI tutor cannot yet remember previous messages exchanged with it to allow for continuous conversations. Additionally, until the end, it was difficult to get it to generate functional paths, even with the new approach. Including this in the web app would require further explorations of prompt engineering.

Future Directions

In future iterations, image-generating AI could be utilised to generate visually interesting images of paths for primary school levels, to make using the app and learning programming more appealing to younger students. This AI could then also be used to customise the robot's appearance. Adding more levels and problem categories, such as escaping a maze and avoiding obstacles, would also be desirable. Another idea would be to implement account creation so that students can track and save their progress.

References

[1] Nicholas A Christakis. *Blueprint: The evolutionary origins of a good society*. Hachette UK, 2019.

[2] Hirokazu Shirado and Nicholas A Christakis. Locally noisy autonomous agents improve global human coordination in network experiments. *Nature*, 545(7654):370–374, 2017.

[3] Google DeepMind `https://www.deepmind.com/research/gemini`. Gemini ai: An advanced language model.