Python Screening Task 3: Evaluating Open Source Models for Student Competence Analysis

Research Plan:

To do this, I would first start by searching open-source AI models that have the ability to comprehend and work with code. I want to find models that can do much more than identify syntax errors, as I am interested in models that can "think" in a conceptual manner. I would conduct my search through websites like Hugging Face when searching for special AI models that are meant to comprehend and work with code. I would especially examine the models in the 'Code Llama' group because they are made for this specific purpose. I would also seek out projects hosted on GitHub that may have smaller projects and more specialized tools that can analyse code in an educational setting.

To assess whether a model is sufficient to support a student's learning, I would look at three criteria: how well it understands the code, whether the questions it generates are useful, and the time it takes to do these things. To test this, I would select some typical scenarios that a student learning Python might encounter, for example, a student attempting to write a function to conduct a simple task, but missing the logic of the function. To truly assess whether a model is adequate, I would create some simple mistakes in Python code, mistakes that someone like me, a student, might make. Then I would framed a thoughtful, good question for each. I would give the model the bad code, and assess whether the questions it asked, were as good as mine.

Reasoning:

1) What makes a model suitable for high-level competence analysis?

In order for an AI model to be effective at analysing a student's capabilities, it must understand the rationale behind a student's code, and not simply if there is an error in the code. This is what is known as high-level competence analysis. An outstanding model is able to identify a student's more serious misconceptions: for example, a student might confuse a list with a dictionary, or they might not grasp that they have different variable availability in various parts of the program. A good AI would not just say "you made a mistake here", it would act like an astute tutor, one that leads the student to self-discover the error by asking questions in a way that helps the student reflect on what they did. For instance, instead of giving them the answer, it might ask, "what do you think will happen to this variable inside the function? What is the difference between a list and a dictionary?" I want the AI model to promote genuine learning of concepts, and not have students simply copy-paste lines of code.

2)How would you test whether a model generates meaningful prompts?

To determine if the AI questions are effective, I will enlist the help of a couple of friends who are experienced in Python. I would have the AI generate a lot of questions and then let my friends use them and score them from 1 to 5. They would score the questions based on the following criteria: Are they clear? Do they identify a real problem in the code? And most importantly, do they engage the person in an analysis of the mistake vs just looking up the answer? I want the AI to generate a question that is personal to the student problem, not a generic question.

3)What trade-offs might exist between accuracy, interpretability, and cost?

This is an enormous challenge. Larger and more capable models like GPT-4 would clearly be capable of this task, but they are closed-source, expensive, and computationally heavy. In an educational setting, we will never know how the model generated any of the real sample prompts. The problems with the uncertainty in what it generates are problems in and of itself, and if it generates some ridiculous prompt, I would not even know how to make sense of it. And yes … it will be less accurate than a bigger model, but a smaller open-source model will represent less compute when running and is easier to adapt and to explain. So it will be a trade off between a bigger, less easily understood model, and a smaller, less accurate, but cheaper and more understandable model.

4)Why did you choose the model you evaluated and what are its strengths or limitations?

I would utilize a model like Code Llama for my initial evaluation. I would select it because it is a free, open-source LLM for coding purposes, and is small enough that I could run it directly on my computer rather than pay a large cloud service fee. Its green flag is its broad training data set on code, so it's got a decent baseline understanding of Python, for example. Its red flag is its small size; it probably will not be great at complex problem solving . What you have to say, though, is that it will not produce perfect or even acceptable responses to those tricky mistakes like a gargantuan AI such as GPT-4. It's a good starting point for a school project because of how easy it is to access and use.