AI-Lab CS251 S25

Reference: *Dickey, E., Bejarano, A. & Garg, C. AI-Lab: A Framework for Introducing Generative Artificial Intelligence Tools in Computer Programming Courses. SN COMPUT. SCI. 5, 720 (2024). https://doi.org/10.1007/s42979-024-03074-y*

The following three sections are taken directly from internal course records, the finalized AI-Lab Prelab, and Lecture Notes for the AI-Lab itself, respectively, for CS251 Data Structures and Algorithms at Purdue University during Spring 2025.
The lecture slides accompany this document or can be found on Ethan Dickey's homepage: https://www.cs.purdue.edu/homes/dickeye/index.html

Informal Notes

Tarjan's Algorithm in class activities

Guaranteed 100% for homework 6+ (the AI-required problem) if you complete its problem and do both sets of surveys.

Cheating policy: if you copy paste from ai, it is still cheating. We will evaluate correctness, so be careful if you use the tools for help. Additionally, you will not have these tools on exams, so you need to prepare yourself accordingly. You're responsible 100% for your responses. You may not complain that the AI gave you something wrong.

Survey Links

Perception BEFORE: []
Perception AFTER: []
Usage BEFORE: []
Usage AFTER: []
MSLQ BEFORE: []
MSLQ AFTER: []

- Consult Ethan Dickey and Andres Bejarano for templates of these surveys (we only used parts of the MSLQ)

In-class:

- Tarjan's algorithm
- Intro to LLMs/good things/prompt strategies
- Assigned pairs
- Experience failure:
    - You have 3 options:
    1. Have it create a Tarjan's example, run through the steps, spot its mistakes
    2. Create exam practice problems for Tarjan's algorithm with answers, critique either the question or the answer (or both)
    3. Create practice problems for writing pseudocode on a topic from this semester, using one of the algorithms you've been taught (I've been told you're bad at that)

Postlab ideas:

- Writing test cases, then validate the test cases
    - https://chat.openai.com/share/c7b2acfc-9bf1-4202-95e1-544eab7fcc0c
- Solving 5 problems this week for the price of 4
    - 2 pair problem solving problems
- Writing a reflection on what you didn't understand about its solution
- Required to use chatgpt until you hit a breaking point, then write about it/provide evidence for it.
- **Reflection (see below in policy)**

Postlab policy

- We would strongly encourage you to use the tools with phrases as follows:
    - Help me with X, don't give me all the answers but help lead me to it
    - Provide suggestions for how to start without solving my problem
    - Find my bug but don't tell me exactly where it is, help me discover it for myself
    - Only give me a partial solution to my problem, don't give it all away but guide me to the answer
    - Here's my problem, help me find the issue by asking me guiding questions only
- **Reflection:**
    - Which version of GenAI
    - If it failed, when and why, provide links
    - If it didn't, how did you prompt it and interact with it so that it gave you a good answer
    - Short answer, 1 paragraph
    - If you didn't use it, why not and explain your solution in terms of the week's concepts (or whatever week's concepts that you used, make sure it's in language we used in the course).

ChatGPT Pre-lab Worksheet

Reference: *Dickey, E., Bejarano, A. & Garg, C. AI-Lab: A Framework for Introducing Generative Artificial Intelligence Tools in Computer Programming Courses. SN COMPUT. SCI. 5, 720 (2024). https://doi.org/10.1007/s42979-024-03074-y*

Note: Please make sure that you do the pre-lab surveys in Brightspace before starting this lab.

Please ensure that you have completed the following steps before you attend the lab/lecture:

Step 1:

You must have an account to use chatGPT. If you don't have one already, please sign up for one at https://chat.openai.com/. You may also use your Purdue-provided gpt4 access. No particular version is required, but we would recommend going with something strictly higher than ChatGPT 3.5.

Step 2:

Once you have signed up for an account, familiarize yourself with the tool by getting comfortable with the user interface and by asking the tool a few questions.

1. You can start with a general question, for example: "Tell me about yourself," and once the tool responds, continue asking the tool some basic questions. You can talk to it as you would to a human. Some example questions are:
    a. What are some of the places to visit in Indiana? or Does West Lafayette/Lafayette, Indiana, actually have anything to do?
    b. What is a good recipe for chocolate chip cookies?
    c. What the heck is dairy-free chocolate and why does it need maple syrup?
    d. How do I make dairy-free chocolate?
    e. Help me plan a trip to <a place you want to visit>. (e.g. Mars via Elon)
2. Once you are comfortable with these general questions, move towards questions that are more in line with your studies in computer science. You can ask questions about topics that you have learned in this class and ask the tool to revisit the topic with you or reteach you a topic. Some example questions are:

a. What is Big-O?
b. What is a linked list? Help me visualize a linked list.
c. What is a left leaning red-black tree? Give me an example of a left leaning red black tree.
    i. Be wary of incorrect information
d. Take a look at this for inspiration/hints at one activity during class: https://chat.openai.com/share/c7b2acfc-9bf1-4202-95e1-544eab7fcc0c

Note: If you have a question about one of its responses or it uses terminology that you are not comfortable with, try asking it follow-up questions. A general tip to any question you ask is to be specific. Tell it exactly what you want. Give it context. Give as many details as you can. For example, for the question - Help me plan a trip to <a place you want to visit>, you can tell it details like:

- How much time do you have?
- What time of the year do you want to visit that place, or are you open to any time during the year?
- Is there a specific thing/activity you are interested in at that place?
- Other relevant details that you might have.

Treat your prompt as if you were giving instructions to a personal assistant. They would need to know every detail you can give them to make the best trip possible for you. The same is true of ChatGPT.

Step 3:

Finally, let's introduce the topic that will be covered in the lab/lecture. Use (copy) the prompt that we give you for this task. We have provided you with an exact prompt that you will use for this lab as well as a version with the basic structure of the prompt written out. You only need to introduce yourself to the topic. We will be going into the details in the lab/lecture.

Prompt to copy: Act as a professor of computer science at Purdue University with a masters in Engineering Education. I want you to give me a short introduction to Tarjan's Algorithm. I already know DFS/BFS and Kosaraju's and have an understanding of graphs, though I don't know why Kosaraju's is different, please highlight this. The learning objectives for this section are as follows: 1. Classify graph problems for Tarjan's algorithm application by identifying key SCC-related characteristics. 2. Create solutions for graph problems using Tarjan's algorithm, focusing on problem analysis and constraint optimization. 3. Debug Tarjan's algorithm implementations through tailored test cases addressing graph problem complexities. No need to repeat the LOs.

Note the very specific task we assigned: giving a short introduction to a particular topic.

Fill-in-the-blank Prompt: “Act as a [insert top-level expert in the field you are trying to learn about, with more qualifications or titles if necessary]. I want you to give me a short introduction to [whatever the topic is]. I already know [list all relevant previous topics here]. The learning objectives for this section are as follows: [do some work to figure these out (see below), or ask the professor].”

Feel free to ask questions or clarifications on the response ChatGPT generates. Chat with it like you would to a professor and ask for further explanations and examples. You can also ask it to help you visualize the topic you are trying to learn. Note that ChatGPT3.5 is not very good at giving visual examples for Tarjan’s and some other graphical topics, in particular.
Note: You may want to try the sample prompt with a topic of your choice. A pro-tip is to use Bloom’s Taxonomy for defining the learning objectives. You can define the learning objectives in your own language and ask ChatGPT to rephrase/rewrite them using Bloom’s Taxonomy. More about Bloom’s Taxonomy here: https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/

You can use the following prompt to rewrite learning objectives using Bloom’s Taxonomy:
Act as a professor of engineering education to rephrase these learning objectives using Bloom's taxonomy. The learning objectives are: <List the learning objectives you want>

You can use this prompt to get learning objectives for an introduction to a topic:
Act as a professor of engineering education and computer science. Give me 3 learning objectives using Bloom's taxonomy for the topic of Tarjan’s Algorithm for an in-class lecture.

Lab/Lecture

This part of the lab will focus towards familiarizing students with uses of ChatGPT, when should it be used, when should it not be used, what it is good at, what might it not be good at etc. This will also have an interactive part where the students will teach themselves a new topic. Students should have their laptops on hand to use ChatGPT in class as the instructor does it on the projector.

Start off with what is GenAI? How is it useful in daily life? What are some of the simpler tasks it can do and is good at?

- Try using GPT.
- Make sure to include examples - Some of these have been covered in the pre-lab worksheet. A quick recap or live demonstration should be good to get everyone on the same page.

Then proceed towards using GenAI in education. When should you use it and when should you not use it?

- Personal tutor for conceptual questions:
    - It is good to teach oneself one topic
    - Good to ask questions on a concept you don't understand
    - It is good to generate practice questions on a topic you want to strengthen
- It is good to reword sentences to make them more precise and clear. It is also quite creative in adding narrative to a story or a motivating example for a question or in a different scenario which requires adding narrative to a story or a motivating example.
    - Re/writing/drafting/narratives/motivating examples
- Generating practice questions and test cases:
    - https://chat.openai.com/share/c7b2acfc-9bf1-4202-95e1-544eab7fcc0c
- Asking homework questions or project questions without understanding the topics is something that should not be done.
- A very important point to get across to the students is to make sure they know that not only will it (ChatGPT) make mistakes, but that it sounds very confident in them. Give a specific example here and say we'll play with this idea of mistakes in output to help teach you topics.
    - https://chat.openai.com/share/20de44cb-7c29-47a1-88ce-db77e923c312
    - This is incorrect about the runtimes. See wikipedia.
- It shouldn't be blindly trusted as it makes mistakes when solving some of the easiest of questions. (something we will see later in the lab)

Do the students know how to use GenAI well? Go over how to use GenAI. Using the framework saw a drastic improvement to the answers provided by ChatGPT.

GenAI Framework: How can we use GenAI well?

(Lecture Notes)

Understanding how to interact with GenAI

- The single most important thing you can get from this section is that **<u>you *must* be *clear*, *concise*, and *specific* with what you want</u>.**
    - Imagine you are giving instructions to a 4-6 year old child. They are very good at executing *exactly* what you tell them to do, assuming they know how to do it, but are not very good at understanding what you *meant* for them to do, if it does not align with what you told them to do exactly. ChatGPT is like a 4-6 year old child that has been trained on the entire internet instead of just 4-6 years of human experiences.
    - Furthermore, you can converse with it ("Chat") just like you would a human. You can say things like "the part about X wasn't very clear, could you explain it differently, please?" or any other way you would communicate with a human being. It is a *language* model, so understanding what you say to it is what it is best at.
- Another thing we want to emphasize here is *do not expect it to be perfect all the time, and do not give up because it didn't give you what you wanted the first time*. Much like a google search, you may have to tailor your search to get exactly what you want.
    - For example, if you ask it to give you 10 things that match your requirements, you may get 2-4 decent or good responses. From there, you can tell it which ones you liked and work with each one individually to make it better.

Crafting your prompt

There are 2.5 parts to crafting your prompt:

1. **Set your context.** For example, "act as a collegiate professor with a PhD in computer Science for a sophomore level data structures and algorithms course. Students are familiar with java."
    - Alway specify at least one expert to act as. You could specify multiple (e.g. "act as a computer science professor and experienced statistics researcher") or add qualifications like we did in the prelab, if those qualifications aren't obvious (e.g. "Act as a professor of computer science at Purdue University who has been trained how to teach well.").
    - Tell it what setting that expert is acting in. These give it a starting point for what level of help you are asking for. For example, "Act as a political science professor teaching a summer school for gifted high school students" vs "Act as a political science professor teaching a summer school for academically advanced high school students" tells two slightly different messages.
        - The first states that the students are smart but likely don't have an introduction to the material yet and have likely only experienced high school level education before.

        - The second states that students are smart and likely have taken collegiate-level dual-enrollment-style courses, and thus are more accustomed with the speed and density of collegiate-level material.
        - In our prompt, we specified "sophomore level" class because data structures and algorithms courses can be offered at many different levels, so you could get anything from beginner to advanced without being more specific.
    - Lastly, tell it what your audience is specifically familiar with, when asking it to teach you. This tells it what technical lingo is ok to use and what things it can assume you know/skip over in explanations.
2. **Ask it your specific request**. For example, "give me a set of 10 problems to practice implementing BSTA."
    - It won't be good on most of them. You may get 2-4 good problems in a set of 10. Don't expect it to be perfect (see previous section for more on this).
    - In our request, we did several things of note. Let's break down the prompt:
    - "give me": a command, a request, specific and directed at it.
    - "a set of 10 problems": a specific large number so that you have variation to choose from on a specific, actionable thing (a set of problems).
    - "to practice": part of your goals, it gives problems that are more tailored to learning and practicing the different components of the subject
    - "implementing BSTA": words that clearly identify the specific topic you want to practice. You may get more or less specific, with varying success (likely higher success the more specific you get). For example, you could ask for problems about "binary search applications" in general, which will give you more theoretical questions, or about "optimizing search parameters using BSTA," which will give you a set of situations to practice that specific task.
3. 2.5: **Tell it your goals**. This is an optional part of the prompt. If your task is simple and short enough, there is likely no need to tell it your goals. If you want to craft a long set of questions and examples or a detailed itinerary or give you ideas or any number of complex tasks, it is best to tell it where you generally want to go.
    - For example, in the prelab, we told it what our learning objectives were for the introduction to this topic. From there, it knew generally what we wanted you to learn and was able to give a more tailored response.
    - You can ask ChatGPT which tasks would be best to specify goals in the prompt for. We found the results interesting.

Interacting and dealing with the response

It's a *chat*bot! Use the chat feature! You can tell it to be more specific, work with a particular example, give you answers to a particular problem it created, or a large variety of other things that tailor responses to exactly what you want. Some important things to note:

1. When you are asking it to refine a previous response (instead of asking for new content related to previous responses), make sure you **tell it what you liked or didn't like from its response that you want it to do more of or change**, respectively, before asking it

to reword. Simply asking it to reword could give you a large range of even worse responses. Don't stop being as specific as you can during your conversation.

2. When refining particular subsets of lists or specific parts of a response, try to **separate your queries into one per list element or part of a response**. It doesn't do as well when you ask it to refine all of the questions at once. If you think about it, your goals for refining are likely a little different for each question.
3. If it's been a lot of prompts since you asked it to act a particular way or since a response that you keep referencing, **try reminding it what you are talking about**. The LLMs can get confused when the conversation has been going too long without direct reference to certain things. It can, for example, forget that you made a modification to code a long time ago, so it may be helpful to remind it what code you liked by telling it that and by copying and pasting the code.
    a. **It can also be helpful to just start a new chat**. Especially when it has gotten a lot of things wrong, it can be helpful to give it a fresh start, just like you do when you come back to debugging your code after a break (except it has to be retaught the context, etc).