

IN4MATX 285:

Interactive Technology Studio

**Practice: Large Language Models
and Generative AI**

Today's goals

By the end of today, you should be able to...

- Explain, at a high level, how Large Language Models respond to prompts
- Follow a few best practices for prompting, particularly when coding
- Articulate opportunities for LLMs to support development and design practices
- Recognize the limits of LLMs towards supporting those practices

Disclaimers

- This is an overview of how LLMs and GenAI work, and how that influences when you might and might not want to use them
 - I can't cover the entirety of LLMs and GenAI in one lecture
- The field is still emerging, much of what I say today will be irrelevant in 5 years

**What's a Large Language Model, and
how does it work?**

Taking a step back: Natural Language Processing

Natural Language Processing (NLP)

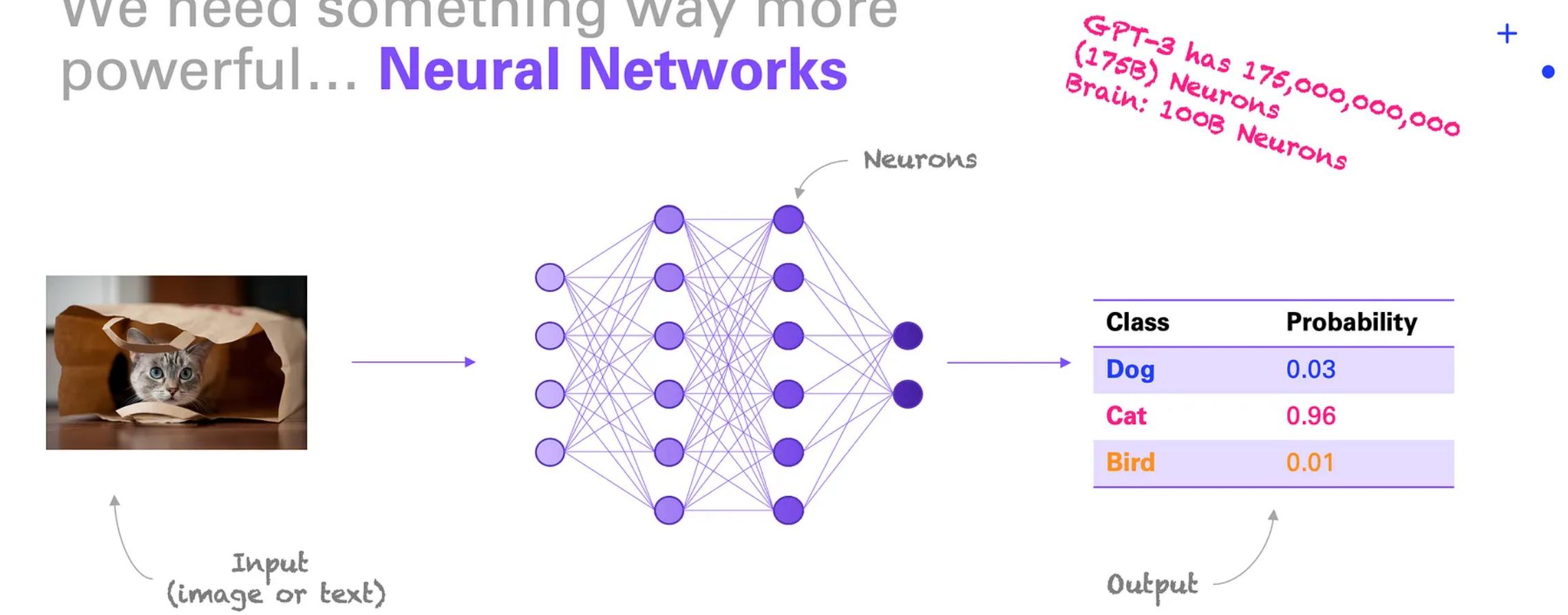
- Super common NLP task: predict the next word in a sentence
 - “The cat likes to sleep in the _____”
- How do we make this prediction?
 - We have a dictionary. We know what words are theoretically possible, but some will be more likely than others
 - So we get a bunch of example sentences, and try to model what might come next
 - Given enough examples, we can make good predictions

<https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f>

Natural Language Processing (NLP)

- This is where math comes in, which I'll mostly gloss over
- Computers try to act like brains, with tons of “neurons” that activate when they see a pattern they've seen before
- E.g., we have lots of example sentences which include both “cat” and “box”

We need something way more powerful... **Neural Networks**



Natural Language Processing (NLP)

- This is still prediction, with a probability associated
 - E.g., 8.5% of the time we will say that the cat likes to sleep in the “box”, 7.1% in the “bag”

Language modeling

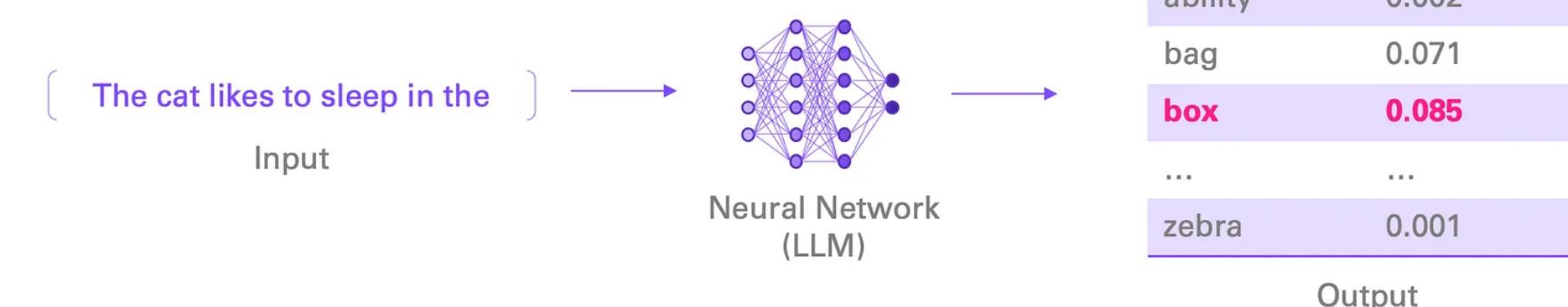
Imagine the following task: Predict the next word in a sequence

[The cat likes to sleep in the ___] → What word comes next?

Can we frame this as a ML problem? Yes, it's a classification task.

+ .

Now we have (say)
~50,000 classes
(i.e., words)

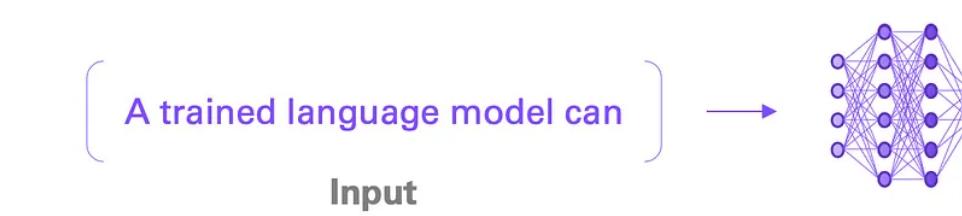


Natural Language Processing (NLP)

- If we chain this together, one word at a time, we can start to generate sentences

Natural language generation

After training: We can generate text by predicting one word at a time



LLMs are an example of what's called "Generative AI"

Word	Probability
speak	0.065
generate	0.072
politics	0.001
...	...
walk	0.003

Output at step 1

Word	Probability
ability	0.002
text	0.084
coherent	0.085
...	...
ideas	0.041

Output at step 2

Large Language Models

- What I've described is NLP. What makes *Large Language Models* special?
- They're **Large**, in two ways
 - They have a ton of input examples. Like, the entire internet, and everything that's ever been digitized (books, media, etc.). So they know a lot about a lot of things
 - They have a ton of neurons. Like, more than the human brain, which lets them do really good pattern matching
- This poses good questions of copyright, ethics, etc.

Large Language Models versus Generative AI

LLMs vs. GenAI

- Large Language Models are specifically trained to understand and generate text
 - Examples: ChatGPT, Gemini, Claude
- Generative AI is a broader category of using AI to generate other forms of content
 - Images/Videos: Mindjourney, Stable Diffusion
 - Code: Copilot, Replit, Cursor

LLMs vs. GenAI

- Generative AI may have non-text input and/or non-text output, but the fundamentals are the same
 - E.g., if we want to generate images, we still process example images in largely the same way

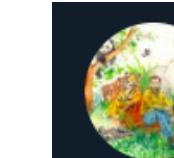
**So how do LLMs/GenAI help with
Development? Design?**

GenAI and Development

- Tons of examples of code exist, so Generative AI can similarly be used to generate code which meets certain criteria
- What can get generated well?
 - Common functionality: If it's likely that someone has done it before, an LLM can help you reproduce it
 - Syntax: LLMs pick up patterns in the examples, code is strict in syntax, therefore lots of examples

GenAI and Development

- “Vibe coding”: the latest craze
 - Ask an LLM to make a thing, ask it to fix the errors
 - “Not too bad for throwaway weekend projects”



Andrej Karpathy ✅
@karpathy

Ø ...

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

3:17 PM · Feb 2, 2025 · 4.9M Views

GenAI and Development

- Copilot, Replit, Cursor all can help with generating code that fulfill specific goals
 - Drawing on examples that exist
- And, can attempt to explain what a piece of code does
 - Drawing on documentation that exists
- “Better autocomplete”, “Improved documentation”

The screenshot shows a code explanation interface. At the top, there's a code snippet:

```
18 const originalResJson = res.json;
```

Below it is an 'Explain' button and a 'TypeScript' label. The main area contains an explanation:

Certainly! Let's break down the provided code snippet from your TypeScript express application:

```
res.json = function (bodyJson, ...args) {  
  capturedJsonResponse = bodyJson;  
  return originalResJson.apply(res, [bodyJson, ...args]);  
};
```

There is also a 'Copy' button. Below the code, there's an 'Explanation:' section with numbered points:

- 1. Function Assignment:**
 - `res.json` is a method in Express that sends a JSON response. Here, it's being overridden with a custom function to add extra functionality to the original `res.json` method.
- 2. Parameters:**
 - The function takes a parameter `bodyJson`, which represents the JSON data to be sent as the response.
 - The `...args` is the rest parameter syntax in JavaScript/TypeScript, allowing the collection of all remaining arguments supplied to the function into an array.
- 3. Captured JSON Response:**

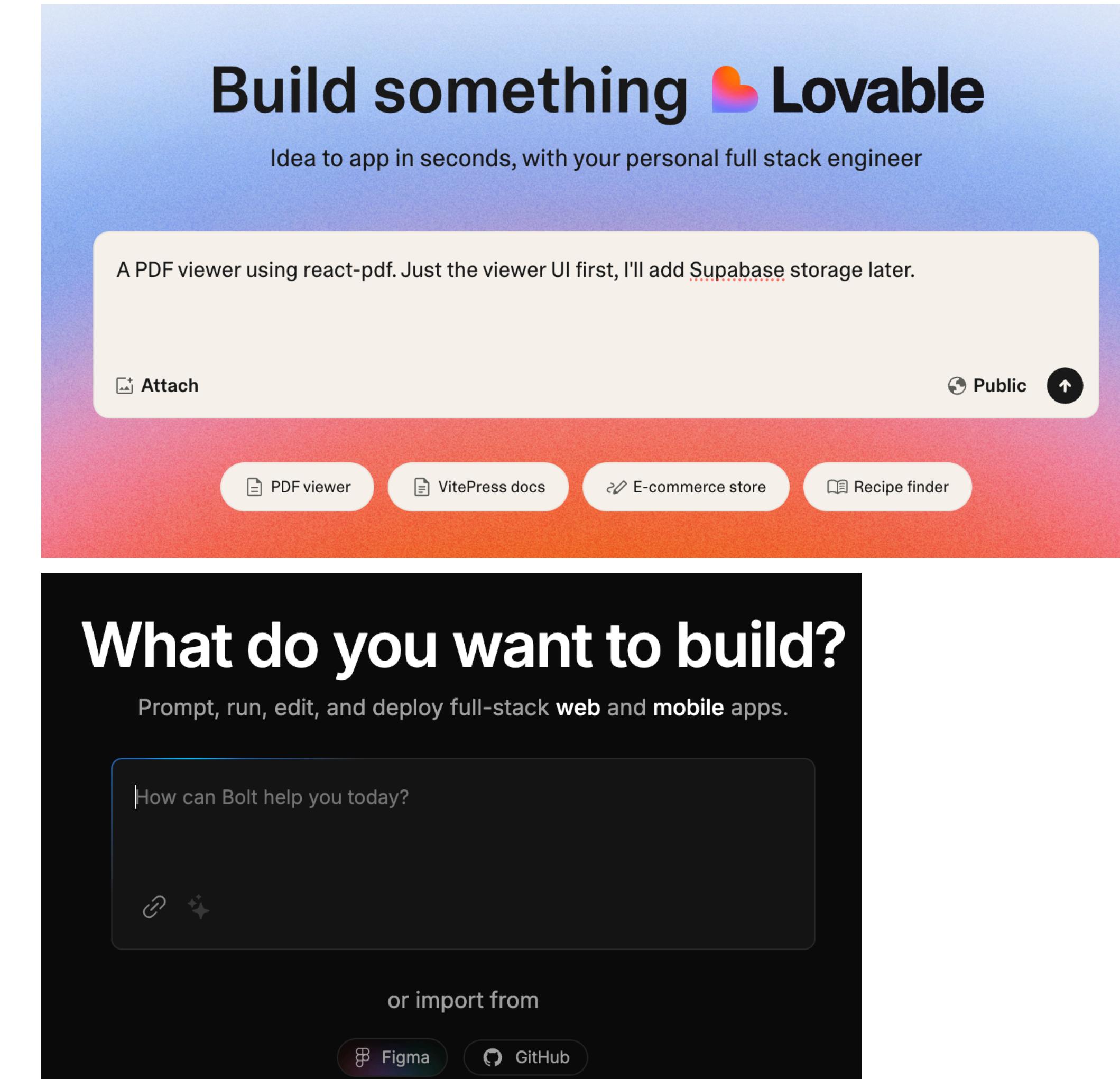
GenAI and Design

- Tools exist, or will exist, to help implement apps via text descriptions and/or convert Figma mockups to functional prototypes
- Loveable, Bolt, Figma Make
- Worth looking into and seeing what you think of them

<https://lovable.dev/>

<https://www.figma.com/make/>

<https://bolt.new/>



Effective prompting

Effective prompting

- Providing examples

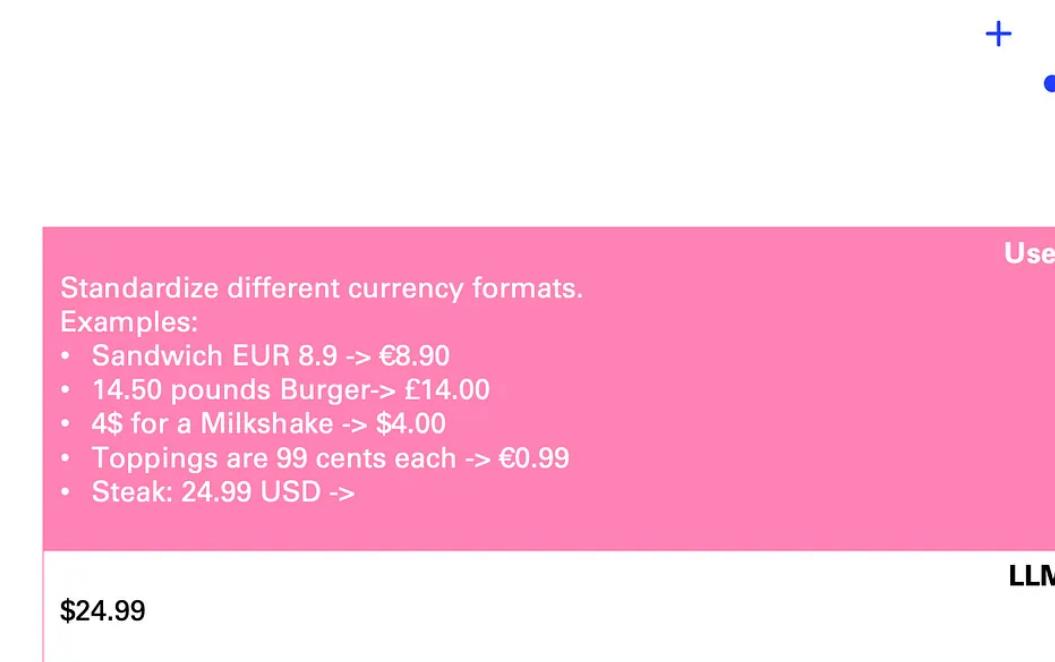
- “Write me some code that turns X input into Y output”
- “Format phone number 9492049299 as (949) 204-9229”

- “Few shot” learning

Few-Shot Learning

Providing **examples** helps the LLM understand and follow your task.

This is especially helpful to ensure a specific **output format**.



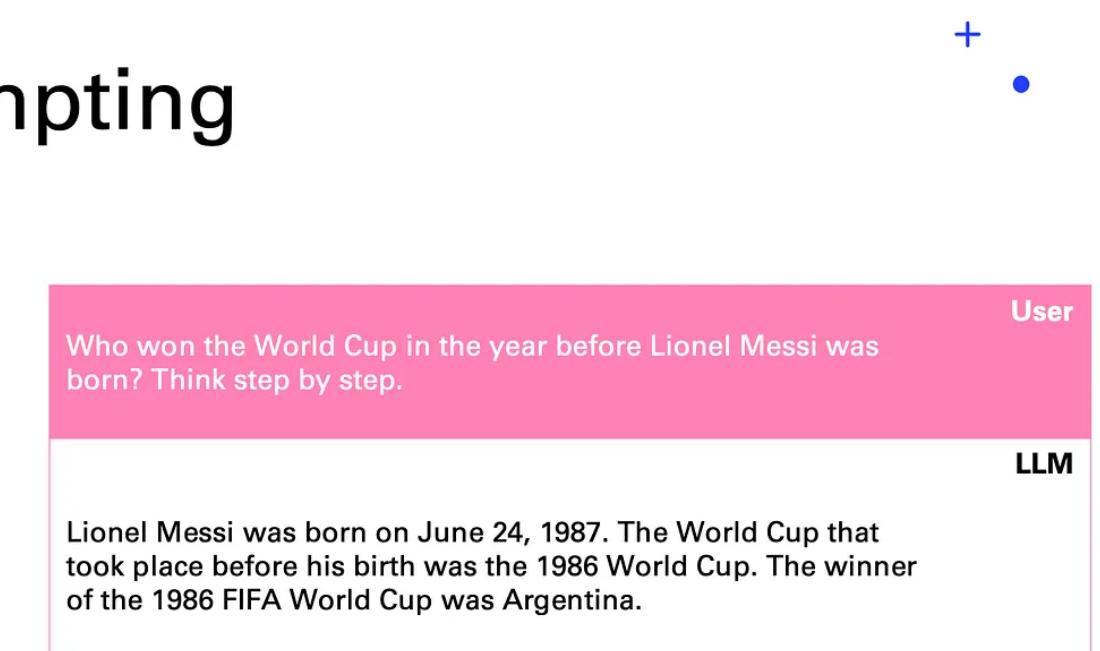
Effective prompting

- Asking for step-by-step thinking
- “Make an app an interface with XYZ characteristics. Then, connect it to a database that has ABC columns”
- Could break it down into separate prompts
- Chain-of-thought prompting

Chain-of-Thought Prompting

Ask the model to solve complex tasks
step by step.

Why does this work?
It gives the model a **working memory**, similar to humans.



Effective prompting

- Overall, recognize that prompting is a conversation
 - Ask for edits, refinement, etc.
- Offer context when you can
 - Everything from low-level stuff like programming language to higher-level design goals
 - Specificity helps, but LLMs can also get overwhelmed
- Ask for explanations to help you understand what was produced

Effective prompting

- I think/hope that we will introduce other interaction paradigms eventually, but for now prompting is what we have
 - Research is actively working on this
 - We have a lot of research suggesting that text is not great for these sorts of inputs

Limitations

Missing context

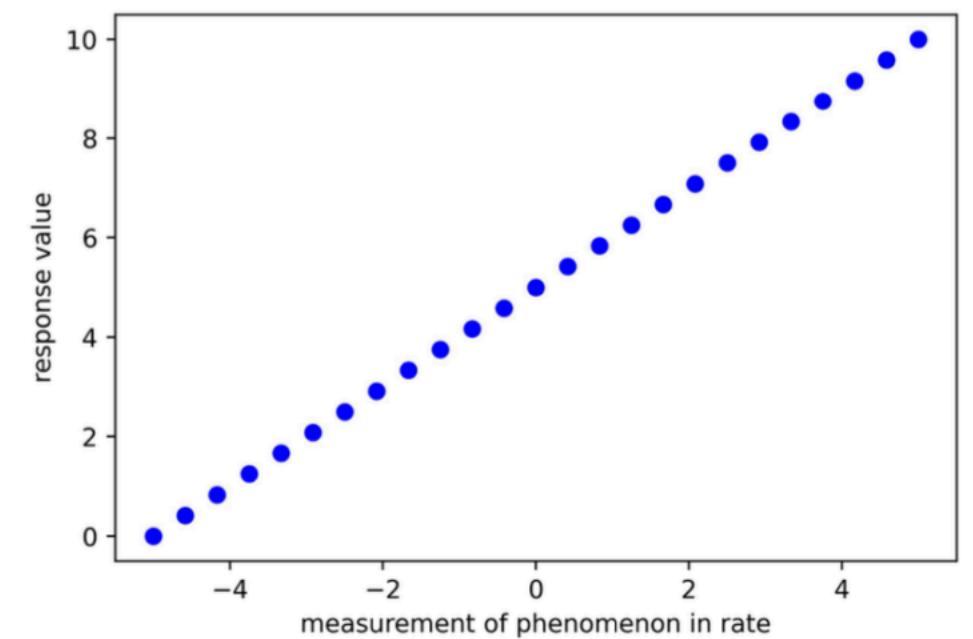
- GenAI only knows what you tell it, and problems are often under-specified
 - GenAI doesn't know the rest of your codebase
 - GenAI doesn't know your design context
 - As you feed in more context via your prompts, the predictive power decreases
- What's really tough about coding is piecing together smaller functions, algorithms, etc. to work as you hope them to
 - Also, connection between different parts of your application like the interface, database, authentication flow, etc.

Hallucinations/truthiness

- LLMs are, inherently, probabilistic. They're predicting the next word, then choosing among the options
- There is nothing to ensure that LLMs are producing text which is factual
 - And, if some text on the internet says something with confidence, LLMs will repeat it
- These are *hallucinations*, LLMs making up facts which aren't true
 - Potentially solvable by adding trusted sources and comparing against them
 - But still an active research area

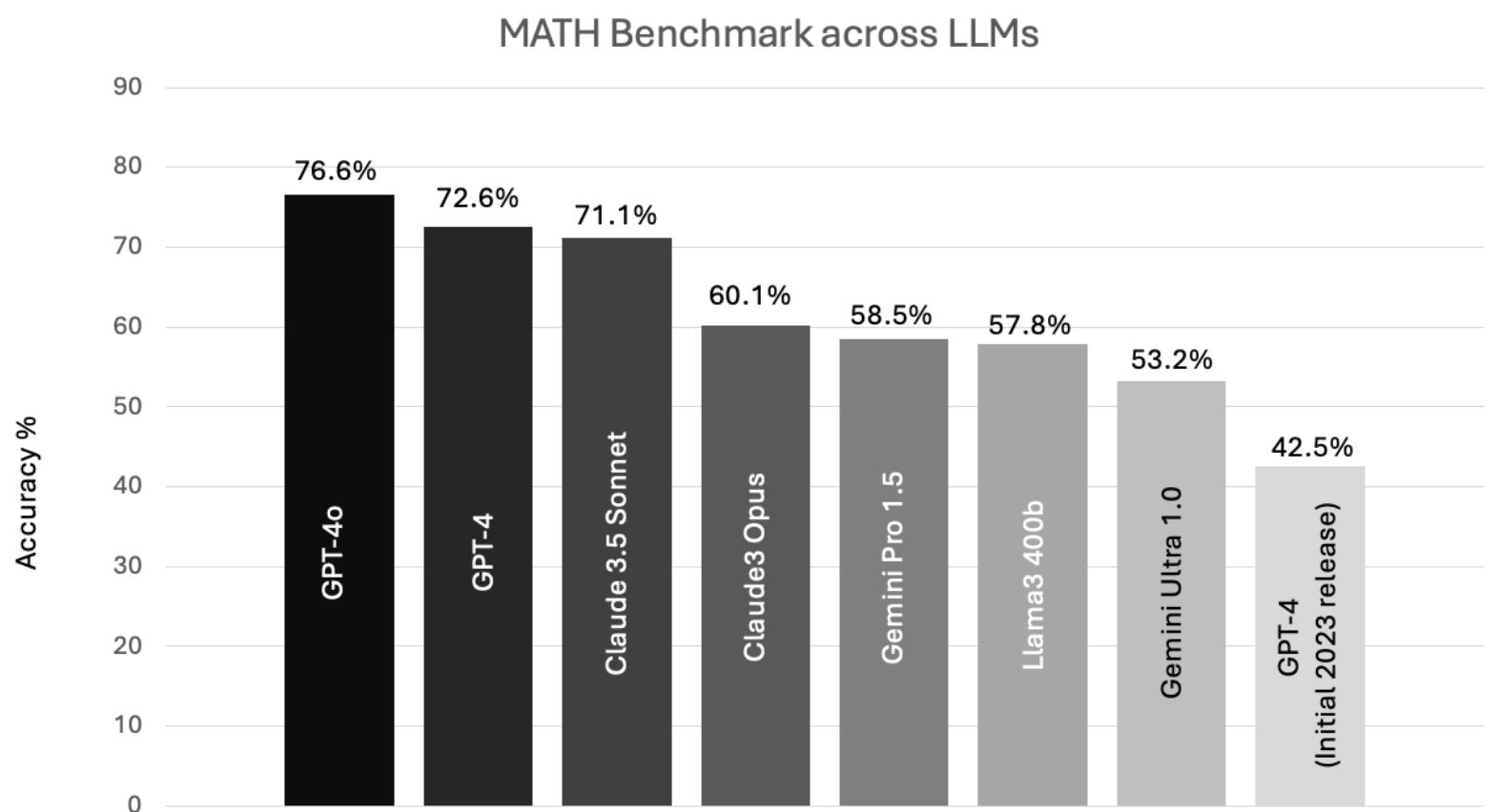
Hallucinations/truthiness

- For example: LLMs are surprisingly bad at math, even basic math
- LLMs are building off of examples, rather than a deeper understanding of the context
- ChatGPT etc. will probably get $1+1=2$ correct, because it's seen examples, but screw up other basic math problems
- This is also an active area of research, and has improved a lot



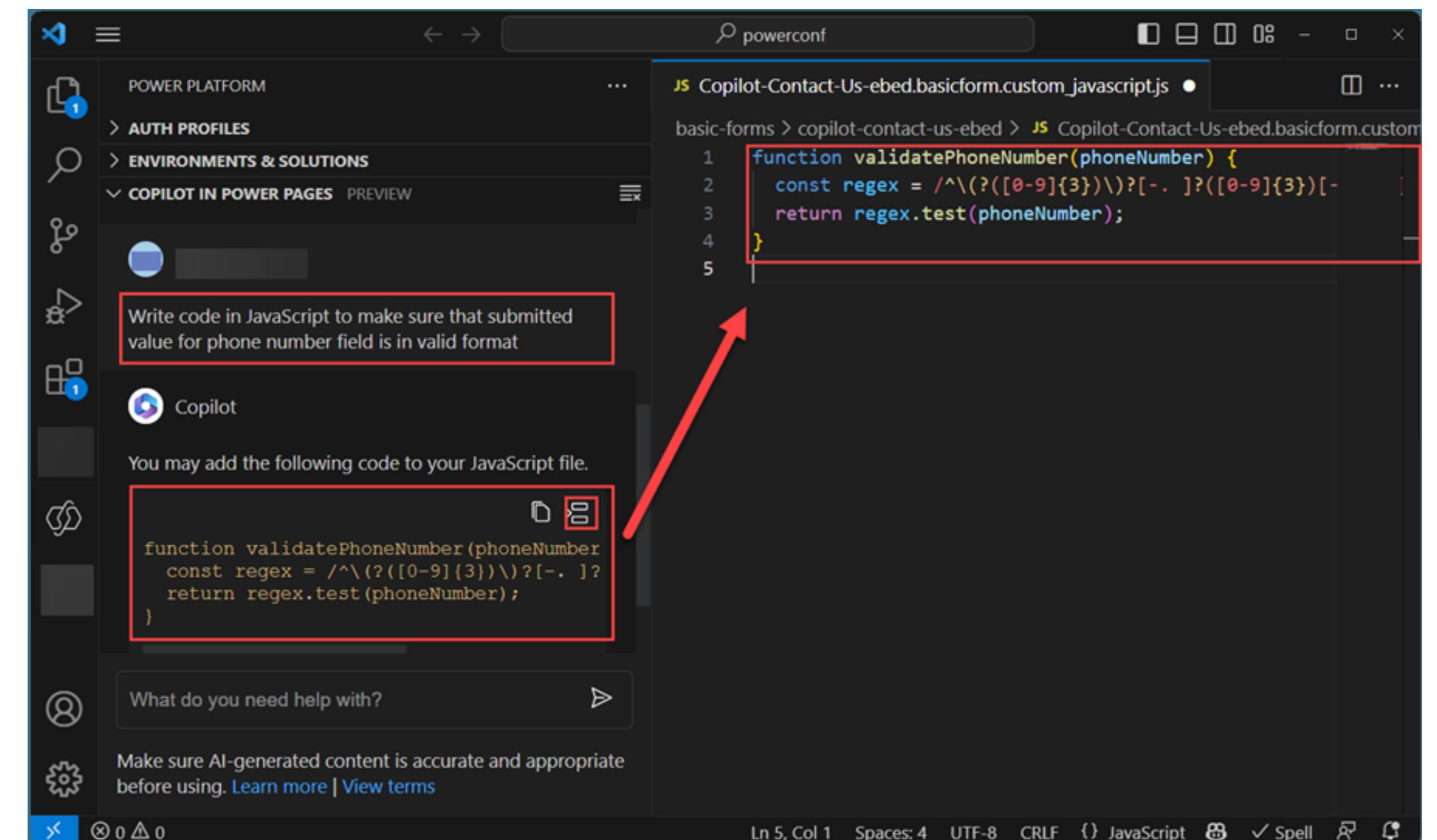
What is the minimum x value among the set of points in the figure? 4 ✗

What is the minimum value for the range on the x axis? 0 ✗



Incomprehensible

- Code generation is one thing, but developers need to understand what the code is doing
 - Is it introducing bugs? Errors? Security vulnerabilities?
 - Is it inefficient? Requiring libraries which aren't needed?
- Does using GenAI make developers less diligent? Maybe



A screenshot of a code editor window titled "powerconf". The file being edited is "Copilot-Contact-Us-ebed.basicform.custom_javascript.js". The code is as follows:

```
function validatePhoneNumber(phoneNumber) {
    const regex = /^(\d{3})\)?[- ]?(\d{3})[- ]?(\d{4})$/
    return regex.test(phoneNumber);
}
```

The code is highlighted with a red box. A red arrow points from the bottom of the slide towards this highlighted code. To the left of the code editor, there is a sidebar with the following sections:

- POWER PLATFORM
- > AUTH PROFILES
- > ENVIRONMENTS & SOLUTIONS
- > COPILOT IN POWER PAGES PREVIEW

Below the sidebar, there is a message: "Write code in JavaScript to make sure that submitted value for phone number field is in valid format". Underneath this message, there is a section labeled "Copilot" with the text: "You may add the following code to your JavaScript file." The same generated code is shown again in a code block with a red border.

Privacy and IP

- Anything that you connect to GenAI becomes part of the examples it uses for training
 - Your intellectual property can become part of the example set
 - Private user data, etc.
- There are ways of avoiding this risk, like enterprise agreements. But it's worth discussing and keeping in mind

Overall reflections

Overall reflections

- Generative AI can be decently effective at entry-level coding and design
 - I wouldn't be surprised if it did a good job at our class assignments, for example
- But, you have to know what to ask it to do
 - Having domain expertise and design/technical expertise helps

Overall reflections

- But, use with caution
 - Using it changes your work from generating to stitching/fixing
 - Is this less work? Easier work? Depends.
- I don't see UX designers or developers getting replaced by AI any time soon
- What have your experiences been?

Today's goals

By the end of today, you should be able to...

- Explain, at a high level, how Large Language Models respond to prompts
- Follow a few best practices for prompting, particularly when coding
- Articulate opportunities for LLMs to support development and design practices
- Recognize the limits of LLMs towards supporting those practices

IN4MATX 285:

Interactive Technology Studio

**Practice: Large Language Models
and Generative AI**