Programming as a Dialogue with GPT-4

Programming as a Dialogue with GPT-4

Is this the future of programming?

Ken Kahn

Retired University of Oxford, toontalk@gmail.com

Since the birth of Logo [9] computer programming has played a fundamental role in facilitating the acquisition of powerful ideas [7]. Learning the concepts and details of programming is a big task but is considered more than worthwhile by the constructionist community and proponents of computational thinking. Debugging programs is hard but a great opportunity for learning some important ideas and skills. Now there is an alternative: creating programs and debugging them by chatting with GPT-4 [6]. In this paper we present several examples of the construction of sophisticated AI programs without writing a single line of a program (or assembling any blocks). The experience is a bit like being the "navigator" giving guidance to the "driver" in pair programming. We ask questions about what this means for both empowering learners as well as helping them to acquire powerful ideas. While it is too early to draw definitive conclusions, the questions raised by this new approach warrant further investigation.

CCS CONCEPTS • Artificial Intelligence • Machine Learning • Human-computer Interaction

Additional Keywords and Phrases: Automatic programming, GPT-4, Computational thinking, Pair programming

1 INTRODUCTION

Over the past 50 years, numerous efforts have been made to enable children to construct AI programs [1,7], with significant developments in recent years, such as Machine Learning for Kids [5], eCraft2Learn Snap! Blocks [2], Wolfram [10], and AI additions to Scratch [8]. GPT-4 [6] now presents a novel approach, allowing children and adults alike to create and debug programs through guidance provided in natural language. In this paper, we share our experience using GPT-4 to develop various AI applications, highlighting the benefits and challenges of this new method.

2 METHODOLOGY

We used GPT-4 to generate AI applications capable of performing the following tasks:

1. Speaking random numbers with random language, pitch, rate, and voice, with subsequent enhancements to repeat user speech in different ways.
2. Drawing on a video feed in response to user gestures and spoken commands.
3. Creating and training a neural network to predict confidence from text data.
4. Simulating a conversation between two personas, allowing user interjections.

In every case GPT-4 created buggy code. When told the program wasn't behaving as expected it always apologized and suggested a correction. Sometimes this was repeated a few times but in all but one of many dozen exchanges it managed to fix the bugs. Sometimes the program failed completely with no error message. Asking GPT-4 to enhance the program to report errors reliably resulted in a version that displayed error messages. Copying those error messages to GPT-4 was sufficient to receive fixes to the problems.

We never presented GPT-4 with a complete description of the desired final product. Instead we asked for a simplified version of what we wanted. Once that was working we asked for enhancements one at a time. Perhaps GPT-4 could manage a more complex task but more likely it may fail or misunderstand and produce something different. Also incremental construction is more likely to lead to the user understanding how the app works.

GPT-4 doesn't just answer requests with code but interleaves the code with explanatory text. If one doesn't understand a generated function one can request GPT-4 to add additional comments. Typically code is lightly commented but GPT-4 produces very informative comments when requested. One can also ask for explanations or a pseudo code version. GPT-4 can be prompted to generate a description of how the app works that would be understandable to a child of a specified age.

Links to the logs of the entire conversations and to the web apps described below are all freely available [3]. The source code for the web apps is available using *view page source* menu item in the browser. [3] also lists six additional web apps created using ChatGPT. These include a water balloon game, an illustrated story generator, fireworks animation, and an app intended to help users understand a proof that there is no largest prime.

3 AN APP THAT SPEAKS RANDOMLY

3.1 GPT-4's response to "Summarize this discussion"

In this discussion, you asked for help in creating a web page that speaks a random integer less than 1000 in a random language, pitch, rate, and voice when a button is clicked. I provided you with HTML, CSS, and JavaScript code to achieve this functionality. Later, you requested modifications to display the number, language, and other details, as well as error messages to the user.

Next, you asked to replace the button with a cartoon image of a parrot that repeats what the user says in a random language, pitch, rate, and voice. I provided code to implement this feature, and to display the text heard along with the language, pitch, and other details.

You then requested to replace the button with a custom image, enlarge the images, and lay them out horizontally. I provided the necessary HTML and CSS changes.

To add translation functionality to the parrot, I provided code examples using the Google Translate API and the Hugging Face API. I also showed how to ask the user for their API key and use it in the translation function.

Throughout the discussion, I provided explanations and troubleshooting tips to address various issues and questions that you encountered. The final result is a web app that speaks a random integer and repeats user speech in different languages, pitches, rates, and voices, while also translating the text using the Hugging Face API.

3.2 GPT-4's response to "Can you explain how this app works to a young child"

Sure! This app has two pictures: one of colorful numbers and another of a parrot. When you click on the picture of the colorful numbers, the app will make your computer or phone say a random number out loud. It's like picking a number from a hat, but the computer does it for you! What's even more fun is that the number will be spoken in a different language, voice, and speed each time you click.

Now, when you click on the parrot picture, the app listens to what you say, just like a parrot listening to you talk. Then, the parrot (actually, the computer pretending to be the parrot) will repeat what you said, but with a random language, voice, and speed. It's like having a funny talking parrot on your screen!

So, this app is a fun way to play with numbers and voices while learning about different languages.

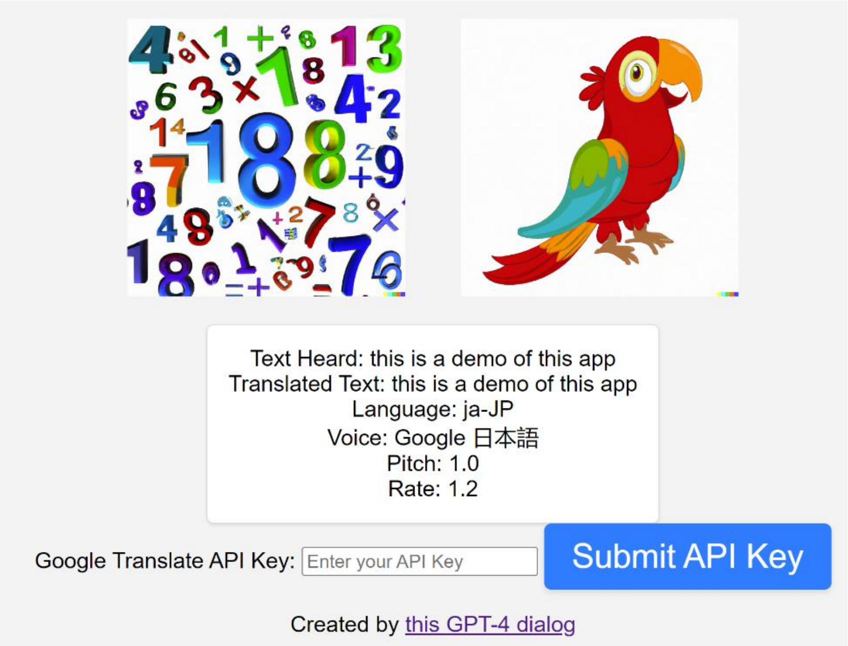
3.3 List of prompts

Here is the entire list of prompts used to create this app. Comments in parentheses and italics.

1. How can I create a web page with a button that when clicked speaks a random integer less than 1000 in a random language, pitch, rate, and voice.
2. The first time the button is clicked the console has this error "speak random number.js:39 No voices available for the selected language". After that it works fine. *(GPT-4 explained why this happens and fixed it.)*
3. Display the number, language, etc. Also report error messages to the user without using the console
4. Add CSS *(looked much nicer after this)*
5. Add all the languages that are available in Chrome *(Previously it had listed just English, Spanish, and French. Its solution will work for any browser.)*
6. Add a cartoon image of a parrot *(created separately using DALL-E 2)* that when clicked repeats what the user says with random language, pitch, rate, and voice.
7. Display the text heard along with the language, pitch, etc.
8. Replace the button with "DALL·E 2023-04-05 12.03.42 - colorful collection of random integers.png" *(I generated an image with DALL-E 2 to use here.)*
9. Make the images larger and lay them out horizontally
10. Can the parrot repeat what was said translated?
11. Ask the user for the API key *(GPT-4 warned against putting the API key in the app and suggested using a server.)*
12. Can I use Hugging Face instead *(At first it generated code that used the Google Translate API. I have an account with Hugging Face so suggested that.)*
13. I'm getting errors like "script.js:145 POST https://api-inference.huggingface.co/models/Helsinki-NLP/opus-mt-en-es 503" *(This error is due to the fact that Hugging Face often replies with an error message that the model hasn't finished loading yet. I replaced the Hugging Face code with the Google Translation version and it worked fine.)*

3.4 The app

The app consists of 3KB HTML and 7KB of JavaScript. The dialogue consists of 4000 words and elapsed over 2 to 3 hours.



Text Heard: this is a demo of this app
Translated Text: this is a demo of this app
Language: ja-JP
Voice: Google 日本語
Pitch: 1.0
Rate: 1.2

Created by this GPT-4 dialog

Figure 1 - A screenshot of the Speak Randomly app

4 AN APP THAT DRAWS ON A VIDEO BY TRACKING A FINGER

4.1 Summary of the dialogue as generated by GPT-4

In this conversation, you asked for help in creating a web page that can detect the direction your finger is pointing and use it to draw on a canvas. The app should also change the color of the drawing based on the last color you spoke. I provided you with code and instructions for creating the app using handpose and speech recognition. We went through several iterations of troubleshooting issues with the code, such as accessing the **<video>** element, speech recognition not working, and displaying the last spoken color *[Author's note: it was the last spoken utterance not color]* and current color on the same line. Eventually, the app started working as intended.

4.2 GPT-4's response to "Could you generate a description of how it works that a 10-year old might understand"

This app is like a magical drawing board. It uses your computer's webcam to see your hand and can tell which way you're pointing your finger. When you move your finger around in front of the webcam, it's like using an invisible paintbrush to draw on the screen!

There's another cool trick this app can do: it can listen to what you say and change the color of the paintbrush based on the color you speak. So, if you say "red," the app will start drawing with red, and if you say "blue," it will switch to blue.

The app uses two special abilities of your computer. One is called "handpose," which helps the computer recognize your hand and know where your finger is pointing. The other is called "speech recognition," which helps the computer understand the words you say.

When you use the app, it watches your hand and listens to your voice. It then combines both of these inputs to create a fun and interactive drawing experience!

4.3 The app

The final product consists of 3KB HTML and 4KB JavaScript. The dialogue consists of about 4000 words and elapsed over 2 to 3 hours.

5 AN APP THAT PREDICTS WHETHER SOME TEXT EXPRESSES CONFIDENCE

5.1 Summary of the dialogue as generated by GPT-4

In this discussion, you asked for help in creating a web page that predicts the level of confidence for a given text based on a CSV file containing labeled data. I provided guidance for building a solution using TensorFlow.js and the Universal Sentence Encoder (USE) in a browser environment. We went through several iterations to improve the model, user interface, and functionality:

1. We created a simple neural network using TensorFlow.js and trained it using the provided data. We used the USE to convert text to embeddings.
2. We built an interactive web page that allows users to train the model and make predictions. The page includes a form for text input, a button to train the model, and a graph to display training loss.
3. We added CSS to style the web page.
4. We modified the model to include likelihood scores in the predictions.
5. We improved the model's architecture by adding more layers and dropout regularization.
6. We added an input field to let the user specify the number of training epochs.
7. We updated the graph in real-time during training to show the progress.
8. We added comments to the train function for better understanding.
9. We addressed an issue with case sensitivity by preprocessing the input text.

By following the provided guidance and code snippets, you should have a functional web page that trains a model to predict confidence levels in text and displays the results interactively.

5.2 GPT-4’s response to “Can you explain how this app works to a young child”

Sure! Imagine you have a magic box that can tell if someone is feeling confident, neutral, or not confident just by listening to what they say. This magic box needs to learn what confidence sounds like before it can start making guesses. So, we give it examples of things people have said before, and whether those sentences showed confidence, were neutral, or lacked confidence.

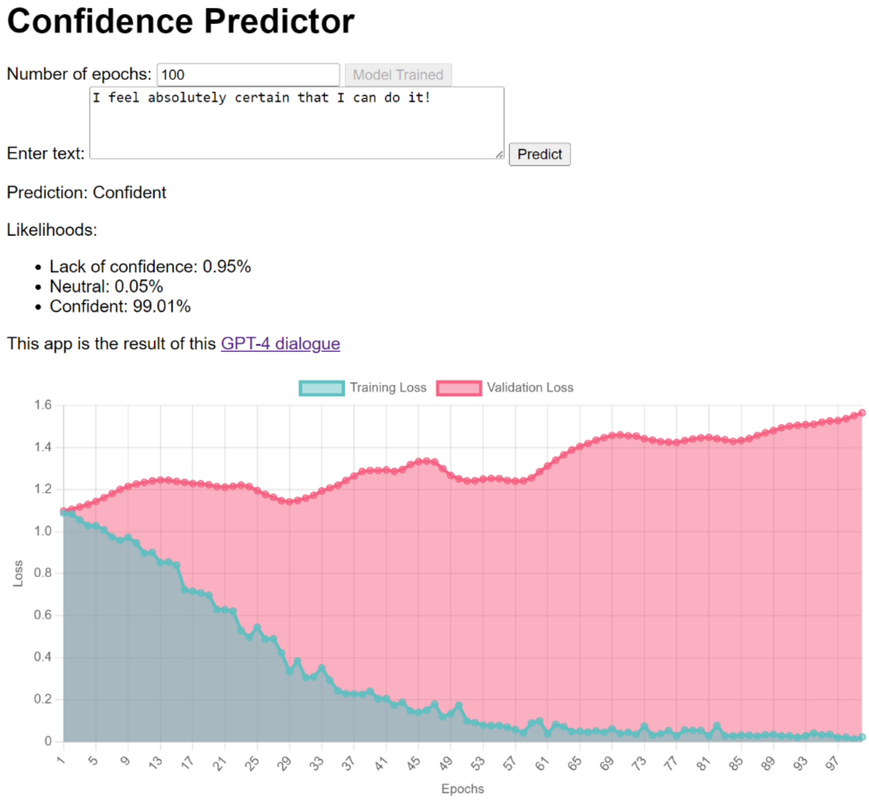
When you press the "Train Model" button on the web page, the magic box starts learning from these examples. It practices many times (called "epochs") to get better at understanding the differences between the three types of sentences. While the box is learning, you can see a chart that shows how well it's doing.

Once the magic box has finished learning, you can type a new sentence into the text box, and the magic box will try to guess if it's a confident, neutral, or not confident sentence. It will also tell you how sure it is about its guess, like "80% confident" or "60% not confident."

And that's how this app works! The magic box learns from examples and then uses what it learned to make predictions about new sentences.

5.3 The app

The final product consists of 2KB HTML and 6KB JavaScript. The conversation consists of about 3800 words and elapsed over 2 to 3 hours.



Confidence Predictor

Number of epochs: 100

Enter text: I feel absolutely certain that I can do it!

Prediction: Confident

Likelihoods:

- Lack of confidence: 0.95%
- Neutral: 0.05%
- Confident: 99.01%

This app is the result of this GPT-4 dialogue

Figure 2 - A screenshot of a confidence predicting app

6 AN APP THAT SIMULATES THE CONVERSATION BETWEEN ANY TWO PERSONAS

6.1 Summary of the dialogue as generated by GPT-4

In this discussion, you sought assistance in creating a web page where users can engage in a conversation with two virtual personas, Aristotle and Galileo powered by GPT using OpenAI's Chat API. We provided guidance on creating the chat interface, setting up the API calls, and handling different aspects of the conversation, such as alternating between personas.

Throughout the conversation, we addressed several issues related to the implementation, including repetitive messages, persona confusion, error handling, message formatting, and input field modifications. We also discussed how to make the speaker's name bold in the chat interface and suggested adjusting the temperature parameter to reduce the chances of repetitive responses from the AI.

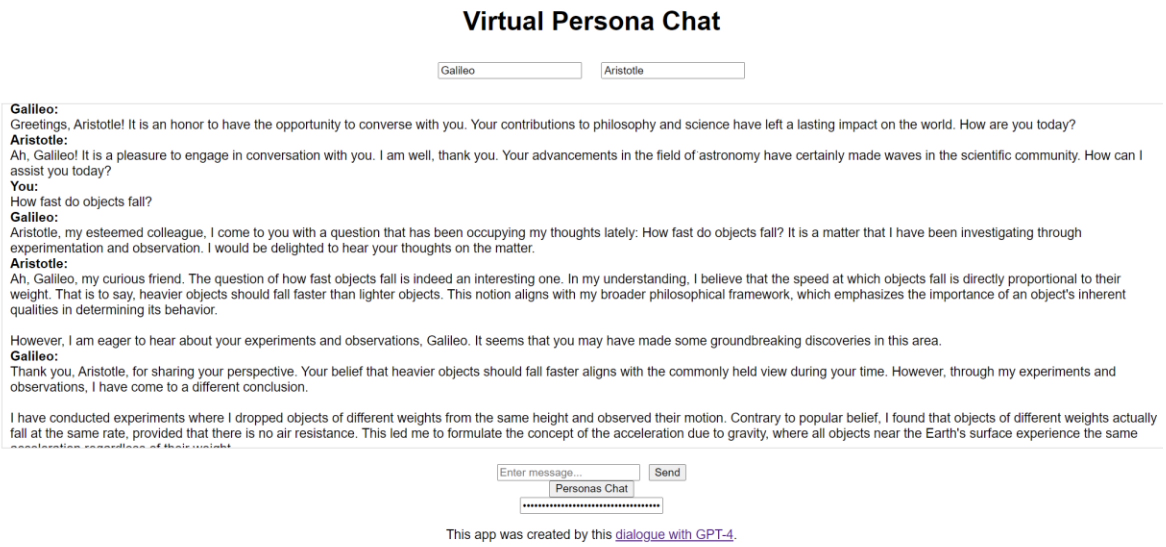
By the end of the discussion, you had a working chat interface where users can converse with the virtual personas, and the API key input field was modified to be hidden like a password field.

6.2 GPT-4's response to "Can you explain how this app works to a young child"

Sure! This app is like a special chatroom where you can talk to two pretend friends named Aristotle and Galileo. They are very smart and can answer lots of questions. When you type a message and press the button, the app sends your message to a big computer called GPT, which thinks of a reply for Aristotle or Galileo to say. Then the app shows their response on the screen so you can read it. You can keep chatting with them and learn fun things about science, philosophy, and much more!

6.3 The app

The final product consists of 2KB HTML, 1KB CSS, and 5KB JavaScript. The dialogue consists of about 6800 words and elapsed over 2 to 3 hours.



Galileo:
Greetings, Aristotle! It is an honor to have the opportunity to converse with you. Your contributions to philosophy and science have left a lasting impact on the world. How are you today?
Aristotle:
Ah, Galileo! It is a pleasure to engage in conversation with you. I am well, thank you. Your advancements in the field of astronomy have certainly made waves in the scientific community. How can I assist you today?
You:
How fast do objects fall?
Galileo:
Aristotle, my esteemed colleague, I come to you with a question that has been occupying my thoughts lately: How fast do objects fall? It is a matter that I have been investigating through experimentation and observation. I would be delighted to hear your thoughts on the matter.
Aristotle:
Ah, Galileo, my curious friend. The question of how fast objects fall is indeed an interesting one. In my understanding, I believe that the speed at which objects fall is directly proportional to their weight. That is to say, heavier objects should fall faster than lighter objects. This notion aligns with my broader philosophical framework, which emphasizes the importance of an object's inherent qualities in determining its behavior.

However, I am eager to hear about your experiments and observations, Galileo. It seems that you may have made some groundbreaking discoveries in this area.
Galileo:
Thank you, Aristotle, for sharing your perspective. Your belief that heavier objects should fall faster aligns with the commonly held view during your time. However, through my experiments and observations, I have come to a different conclusion.

I have conducted experiments where I dropped objects of different weights from the same height and observed their motion. Contrary to popular belief, I found that objects of different weights actually fall at the same rate, provided that there is no air resistance. This led me to formulate the concept of the acceleration due to gravity, where all objects near the Earth's surface experience the same

This app was created by this dialogue with GPT-4.

Figure 3 - Screenshot of a virtual conversation app that uses gpt-3.5-turbo

6.4 Not just Aristotle and Galileo

Later when testing the app I discovered that it didn't work properly when I entered new personas. So I continued the discussion but accidentally in a manner where ChatGPT didn't have enough context and the extended conversation isn't shareable. After several errors the app was revised to enable chats such as one where Aristotle and Galileo were replaced with the words "Lion" and "Elephant" [4].

6.5 But ChatGPT can do this without a specialized app

I then wondered if ChatGPT-4 could simulate a conversation between Aristotle and Galileo and myself. I tried the prompt "Can you simulate a conversation between Aristotle and Galileo in such a way that after every exchange I can add something to the conversation?" and not only did the conversation go well but I was able to bring Newton into the conversation at an appropriate time [4]. I tested this with four other chatbots with mixed results. And ChatGPT-4, as well as GPT-3, were able to simulate a conversation between a lion, an elephant, and myself.

7 TECHNICAL DETAILS

Everything was done in a Chrome browser connected to the GPT-4 March 23 version of ChatGPT. Access to ChatGPT Plus is currently required to access GPT-4 in this manner and costs $20/month. Alternatively, using the OpenAI API of GPT-4 costs $0.06 for every thousand tokens (750 words) generated. So each dialogue would have cost between 25 and 50 cents. Free alternatives exist but the performance of Bard, Bing Chat, Claude, ChatGPT 3.5, and some open source models were markedly inferior to GPT-4. But there will no doubt be improvements in free alternatives.

GPT-4 can generate programs in other languages (at first it suggested Python for the confidence prediction app). An advantage of JavaScript is that running the app can be done in any modern browser without installing any software.

8 DISCUSSION

There are many reasons for a child to learn to program. The question this paper raises (without answering) is how many of them still hold in an era where there is easy access to GPT-4 and the like. One answer is that the child is still programming – only they are doing so in English (or their language of choice, GPT-4 claims “Some of the languages I can converse in include English, Spanish, French, German, Italian, Portuguese, Dutch, Russian, Chinese, Japanese, Korean, Arabic, and many more.”). But they are programming at a much higher level of abstraction than what programming languages offer. They are expressing intentions not procedural instructions. It is a dialogue where the child not only tries to express what they want but also needs to provide the chatbot with useful feedback about failures and errors. The programs (nicely commented and formatted, displayed in five colors) are there for the child to copy and paste. Perhaps also to read or to ask questions of.

The focus of this paper is building AI apps but GPT-4 has been used to create a wide variety of apps including video games. Probably the majority of Scratch and Snap! projects could have been reproduced in JavaScript by chatting with GPT-4. Computer programming has empowered children to creatively express themselves in powerful ways. Chatbots like GPT-4 arguably empowers a larger number of children in a greater variety of tasks. They need fewer technical skills. (I typed the prompts but this is easy to replace with speech. They need to copy and paste the output of GPT-4 into HTML and JS files, but this was automated by ChatGPT’s WebDev plugin after this study was performed. They can construct apps that rely upon powerful AI capabilities such as pose detection, speech synthesis and recognition, vector encoding of texts, neural network training, and much more. While students may no longer be creating programs line-by-line or block-by-block, they instead will be designing apps and creatively combining AI capabilities.

REFERENCES

[1] Kahn, Kenneth. Three interactions between AI and education. In E. Elcock and D. Michie, editors, Machine Intelligence 8: Machine Representations of Knowledge. Ellis Horwood Ltd. and John Wylie & Sons, (1977).

[2] Kahn, Ken and Winters, Niall. “AI Programming by Children”, Constructionism Conference, Vilnius, Lithuania, August (2018).

[3] Kahn, Ken. https://docs.google.com/document/d/1cfRh__IrStR2cszcbG_otphx1MQiu4-b5afhpKd4A48/edit?usp=sharing (2023). Shorter alias: https://tinyurl.com/apps-created-by-GPT4

[4] Kahn, Ken. https://docs.google.com/document/d/1TwBgy3m3L1B-Ub9NAipujebTOK7TWhdzdHwl1aTMTCw/edit?usp=sharing (2023). Shorter alias: https://tinyurl.com/aristotle-galileo-plus

[5] Lane, Dale. https://machinelearningforkids.co.uk/ (2023).

[6] OpenAI. https://platform.openai.com/docs/guides/chat (2023).

[7] Papert, Seymour. “Mindstorms: Children, Computers, and Powerful Ideas.” (1981).

[8] Scratch. https://scratch.mit.edu/studios/2924464 (2023)

[9] Solomon, Cynthia Ann, Brian Harvey, Ken Kahn, Henry Lieberman, Mark L. Miller, Margaret Minsky, Artemis Papert and Brian Silverman. “History of Logo.” Proceedings of the ACM on Programming Languages 4 (2020): 1 - 66.

[10] Wolfram, Stephen https://writings.stephenwolfram.com/2017/05/machine-learning-for-middle-schoolers/ (2017).