Check for updates

Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts

J.D. Zamfirescu-Pereira
zamfi@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Richmond Wong
rwong34@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Bjoern Hartmann
bjoern@eecs.berkeley.edu
UC Berkeley
Berkeley, CA, USA

Qian Yang
qianyang@cornell.edu
Cornell University
Ithaca, NY, USA

ABSTRACT

Pre-trained large language models ("LLMs") like GPT-3 can engage in fluent, multi-turn instruction-taking out-of-the-box, making them attractive materials for designing natural language interactions. Using natural language to steer LLM outputs ("prompting") has emerged as an important design technique potentially accessible to non-AI-experts. Crafting effective prompts can be challenging, however, and prompt-based interactions are brittle. Here, we explore whether non-AI-experts can successfully engage in "end-user prompt engineering" using a design probe—a prototype LLM-based chatbot design tool supporting development and systematic evaluation of prompting strategies. Ultimately, our probe participants explored prompt designs opportunistically, not systematically, and struggled in ways echoing end-user programming systems and interactive machine learning systems. Expectations stemming from human-to-human instructional experiences, and a tendency to overgeneralize, were barriers to effective prompt design. These findings have implications for non-AI-expert-facing LLM-based tool design and for improving LLM-and-prompt literacy among programmers and the public, and present opportunities for further research.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in interaction design**; • **Computing methodologies → Natural language processing**.

KEYWORDS

language models, end-users, design tools

ACM Reference Format:
J.D. Zamfirescu-Pereira, Richmond Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3544548.3581388

This work is licensed under a Creative Commons Attribution-NonCommercial International 4.0 License.

CHI '23, April 23–28, 2023, Hamburg, Germany
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581388

1 INTRODUCTION

The idea of instructing computers in natural language has fascinated researchers for decades, as it promises to make the power of computing more customizable and accessible to people without programming training [4]. The combination of pre-trained large language models (LLMs) and prompts brought renewed excitement to this vision. Recent pre-trained LLMs (e.g., GPT-3 [8], ChatGPT [1]) can engage in fluent, multi-turn conversations out-of-the-box, substantially lowering the data and programming-skill barriers to creating passable conversational user experiences [7]. People can improve LLM outputs by prepending *prompts*—textual instructions and examples of their desired interactions—to LLM inputs. Prompts directly bias the model towards generating the desired outputs, raising the ceiling of what conversational UX is achievable for non-AI experts. In the past two years, social media platforms have witnessed an explosion of posts showing the results of lay peoples' experimentation with LLMs for question answering, creative dialogue writing, writing code, and more. This excitement around LLMs and prompting is propelling a rapidly growing set of LLM-powered applications [23] and prompt design tools [3, 20, 32].

Yet despite widespread excitement, surprisingly little is known about how non-experts intuitively approach designing prompts with LLM-and-prompt-based tools, and how effective they are in doing so. While prompting LLMs can appear effortless, designing effective prompting strategies requires identifying the contexts in which these LLMs' errors arise, devising prompting strategies to overcome them, and systematically assessing those strategies' effectiveness. These tasks fall on so-called "prompt engineers"—the designers, domain experts, and any other end-user or professional attempting to improve an LLM's output—and are challenging tasks even for LLM experts, as well as topics of ongoing research in Natural Language Processing (NLP) [7, 30, 42]. Prompt design tools to date have focused on supporting professional programmers [45] and NLP practitioners [42], rather than non-AI experts, non-programmers, and other potential end-users of these systems.

In this work, we investigate how non-AI-experts intuitively approach prompt design when designing LLM-based chatbots, with an eye towards how non-AI-expert-facing design tools might help. Specifically, we investigate these questions in the context of designing an instructional chatbot, that is, a chatbot that walks the user through an activity (e.g., cooking a recipe, fixing a wifi connection) while answering user questions and engaging in social conversation throughout the process. Traditionally requiring multiple large datasets and extensive model training, building an instructional chatbot is one of the tasks for which a near-trivial prompt provided to GPT-3 (“`Walk the user through the following steps: <steps>`”) alone yields a strong baseline. Among other uses, merely enabling non-AI-experts to customize and improve instructional chatbots has the potential to revolutionize customer service bots, one of the most common chatbot use cases. The explosion of interest in LLM-based ChatGPT [1] demonstrates that chat-based interactions with LLMs can provide a powerful engine for a wide variety of tasks, including joke-writing, programming, writing college-level essays, medical diagnoses, and more; see [41] for a summary.

Toward this goal, we created a no-code LLM-based chatbot design tool, BotDesigner, that (1) allows users to create an LLM-based chatbot solely through prompts, and (2) encourages iterative design and evaluation of effective prompt strategies. Using this tool as a design probe, we observed how 10 participants without substantial prompt design experience executed a chatbot design task using BotDesigner, to explore a few different pieces of this larger question. Our findings suggest that, while end-users can explore prompt designs opportunistically, they struggle to make robust, systematic progress. Their struggles echo many well-known struggles observed in end-user programming systems (EUPS) and non-expect users of interactive machine learning (iML) systems. Additional barriers to effective prompt design stem from limited conceptions of LLMs' prompt understanding and execution abilities and their understandable inclinations to design prompts that resemble human-to-human instructions. We discuss these observations' implications for designing effective end-user-facing LLM-based design tools, implications for education that improves LLM-and-prompt literacy among programmers and the general public, and opportunities for further research.

This paper makes three contributions. First, it describes a novel, no-code LLM-and-prompt-based chatbot design tool that encourages iterative design and evaluation of robust prompt strategies (rather than opportunistic experimentation.) Second, it offers a rare rich description of how non-experts intuitively approached prompt design, and where, how, and why they struggled. Finally, it identifies opportunities for non-expert-facing prompt design tools and open research questions in making LLM-powered design innovation accessible.

2 RELATED WORK

2.1 The Promises of Non-Expert Prompt Design

Today's chatbot design practice—i.e., designing multi-turn conversational interactions—follows a well-established workflow [12, 13, 24, 25, 39, 44]. Designers first (i) identify the chatbot's functionality or persona and draft ideal user-bot conversations, for example, through Wizard-of-Oz or having experts drafting scripts; (ii) create a dialogue flow template (e.g., “(1) greeting message; (2) questions to collect user intention; (3) ...”); (iii) fill the template with supervised NLP models (e.g., user intent classifier, response generator, etc.); and finally (iv) iterate on these components to achieve a desired conversational experience. In this “supervised learning” paradigm, designers make NLP models generate their desired interactions by improving its training data and feature design; tasks that require substantial machine learning and programming knowledge [50].

The emergent “pre-train, prompt, predict[1]” paradigm in NLP promises to lower the entry barrier for non-experts innovating on conversational interactions [30]. In this paradigm, designers can create conversational agents with little to no training data, programming skills, or even NLP knowledge. Leveraging pre-trained large language models such as ChatGPT [1], designers can create a general-purpose conversational system with passable, though sometimes problematic, performance. Next, they can improve the LLM outputs using natural language prompts (see Table 1 for an example) and/or model fine-tuning. In this paradigm, people without programming skills or NLP knowledge can nonetheless make NLP models generate desired interactions by crafting effective prompt strategies (e.g., natural language instructions, examples, and templates) [3, 15].

Prompt Strategy	Resulting Human-GPT-3 Conversation
No prompt (baseline)	🗣️ **User:** Ok hang on while I get a chair 🤖 **Bot:** Scoot to the front of your chair *[...]*
Explicit instruction	👩‍💻 **Prompt design:** If the user asks you to wait, explain that this is not a problem *[...]* 🗣️ **User:** Ok hang on while I get a chair 🤖 **Bot:** Once you have your chair, scoot to the front of it *[...]*

Table 1: An example of how designers (👩‍💻) can directly improve chatbot interactions by modifying prompt strategies. Note the changes in the bot's (🤖) response to the user's (🗣️) statement.

2.2 Known Challenges in Prompt Design

While prompting *can appear* as easy as instructing a human, crafting effective and generalizable prompt strategies is a challenging task. How a prompt or a prompt strategy directly impacts model outputs, and how prompts modify LLMs' billions of parameters during re-training, are both active areas of NLP research [30, 42]. Moreover, established prompt design workflows do not yet exist. Even for NLP experts, prompt engineering requires extensive trial and error, iteratively experimenting and assessing the effects of various prompt strategies on concrete input-output pairs, before assessing them more systematically on large datasets.

That said, ongoing NLP research does offer some hints toward effective prompt design strategies. Notably, these prompt strategies are effective in improving LLMs' performance over a range of NLP tasks and conversational contexts; it remains unclear to what extent these strategies can improve any particular conversational interactions or contexts.

---

[1] *Predict* here refers to generating model outputs, referred to as “prediction” because model outputs are probabilistic predictions for the words (tokens) that might follow the prompt.

- **Give examples of desired interactions in prompts.** The original GPT-3 paper demonstrated that examples substantially improved the performance of GPT-3 on a battery of tasks such as question answering and language translation [9]. This approach appears frequently in online tutorials, such as in OpenAI Playground's example set [36].
- **Write prompts that look (somewhat) like code.** "Prompting looks more like writing web pages." Researchers found that explicitly demarcating prompt text and inserted input data, such as by using a templating language like Jinja, yielded prompts that are more robust to broader input distributions [3].
- **Repeat yourself.** The authors of DALL·E, the large text-to-image model, report that to generate a neon sign that reads "backprop", the prompt "*a neon sign that reads backprop; backprop neon sign; a neon sign that backprop*" can be more effective than the one without the repetition [40]. (This strategy is under-investigated in text generation.)

Little research has investigated how non-experts conduct prompt engineering intuitively or the challenges they encounter. One rare exception is a CHI'22 workshop paper "How to prompt?" [14], where Dang *et al.* conducted a focus group with HCI researchers and identified an initial set of challenges they encountered in prompting. These challenges include "the lack of guidance in trial and error," "poor representation of tasks and effects," and "computational costs and ethical concerns." In prompting text-to-image models such as DALL·E, an adjacent area to prompting LLMs, Liu *et al.* describe a study with artists, identifying emergent strategies for *prototyping* prompts, such as "try multiple generations to get a representative idea of what prompts return" and "focus on keywords rather than the phrasings of the prompt" [31]. This paper builds upon this emergent line of research and aims to deepen these understandings.

2.3 Non-Expert Prompt Design Tools

Arguably, the most widely used prompt design tool among non-experts remains the OpenAI Playground, a "plain text" input box that triggers GPT-3 to predict text completions. The *Playground* interface includes over 50 example prompts and enables anyone to experiment with different prompts for accomplishing various tasks. Though useful, *Playground* lacks any support for systematic evaluation of prompts.

Propelled by the potential of non-expert prompt design, a rapidly growing set of end-user-facing LLM-based applications is emerging from HCI research, many with features that support prompt engineering. Such applications span across LLM-enhanced story writing and programming [23, 38, 45, 48]. NLP researchers have also started creating tools to enable anyone to contribute prompts to help train LLMs [3, 15].

Interestingly, these tools offer little to no support for non-experts in generating or evaluating prompts, often assuming that looking at individual input-output pairs in isolation is sufficient. One telling example is *AI Chains* [49], a tool for exploring human-LLM collaborative writing interactions. AI Chains allows designers to construct a chain of LLMs where the output of one LLM becomes the input for the next, and to test the resulting interactions themselves. The tool successfully enables designers to explore prompt and chaining strategies more efficiently and strategically [48]. However, it is unclear whether the resulting strategies are effective or robust beyond the few interaction contexts that the designers experimented with.

2.4 Known Challenges in Non-Expert Programming and Machine Learning

Early HCI research tells us that program-authoring interactions that do not require programming are not necessarily accessible to non-programmers; drag-and-drop and interactive machine learning (iML) tools cannot necessarily enable non-ML experts to build models [50], and prompting can be viewed as a programming or iML task. In this context, one might ask: does prompt engineering similarly involve tacit knowledge that non-AI experts do not have? What mental models might hinder non-AI experts' ability to devise robust prompt strategies and create desired conversational interactions? Research on prompt design has not yet asked these questions.

Here, we briefly overview known barriers and challenges in end-user programming and iML (list below [27]) as well as how experts and non-experts approach these tasks differently (Table 2), to motivate and contextualize our investigation into end-users' intuitive approaches to prompt design.

- *Design barriers:* "I don't even know what I want the computer to do..."
- *Selection barriers:* "I know what I want the computer to do, but I don't know what to use..."
- *Coordination barriers:* "I know what things to use, but I don't know how to make them work together..."
- *Use barriers:* "I know what to use, but I don't know how to use it..."
- *Understanding barriers:* "I thought I knew how to use this, but it didn't do what I expected..."
- *Information barriers:* "I know why it didn't do what I expected, but I don't know how to check..."

In this work, we use these previously-identified challenges to better understand *why* we observe some of the struggles with end-user prompt engineering that we do.

3 METHOD: DESIGN PROBE

In this work, we aim to understand how non-experts intuitively approach designing *robust* prompts, and whether and how they struggle in doing so. We want to dive deep into what those struggles reveal about people's assumptions and understanding of prompt-based systems in order to inform how end-user-facing prompt design tools might best support their users.

We chose to investigate these questions in the context of designing instructional chatbots—chatbots that walk users through an activity (e.g., cooking a recipe, fixing a wifi connection) while answering user questions and engaging in social conversation as needed. Chatbot tasks are sufficiently open-ended that they open the door to a wide variety of types of problems and prompt-based solution approaches. Instructional chatbots represent a common chatbot use case (e.g., customer service bots), and their constituent tasks are some of the tasks that GPT-3 peforms more effectively out-of-the-box.

PROGRAMMING	**Non-Experts' Intuitive Approach**	**Experts' Intuitive Approaches**
Task requirement	Implicit	Explicit
Task specification	Implicit	Explicit
Code reuse	Unplanned	Planned
Code testing & verification	Overconfident	Cautious
Debugging	Opportunistic	Systematic
MACHINE LEARNING	**Non-Experts' Intuitive Approach**	**Experts' Intuitive Approaches**
ML task design	Directly map personal need to model task	Framing an achievable task
Debugging	Add more data, or "use deep learning"	Identify solvable underlying problems
Measuring success	Consider only one performance metric	Seek multiple performance indicators

Table 2: Known differences in non-experts' and experts' approaches to software engineering [26] and to machine learning model building [50].

In support of these goals, we developed a no-code prompt design tool, BotDesigner, as a design probe [6], and conducted a user study with components of contextual inquiry and interview. We chose to create a design probe because the robustness of prompt strategies is highly dependent on specific conversational contexts, as is users' ability to create prompts and debug. Enabling people to engage in designing LLM-and-prompt-based chatbots *hands-on* offers deeper insights than conducting interviews about hypothetical scenarios alone. We chose to purpose-build a prompt design tool because existing tools focus on enabling non-experts to experiment with various prompting strategies rather than crafting robust and generalizable prompts.

3.1 Designing a No-Code Prompt Design Tool as Probe

Design Goals. We have two goals in designing BotDesigner: First, to enable end-users, without any prompt design experience or even programming or ML experience, to (i) create a chatbot using prompts, without code, and (ii) systematically evaluate their prompts. Second, to allow end-users to engage in this process in a flexible, open-ended way, allowing us to observe their intuitive behaviors and thought processes.

Supporting a full chatbot design workflow, while adhering to a prompt-only, no-code format, BotDesigner has to support two separate, complementary activities:

- **Conversation authoring & interaction**: Users instruct chatbot behavior with prompts alone (without code) and can observe the effects of their prompts by engaging in conversation with the chatbot.
- **Error testing**: Users collect a set of conversations using each draft of their prompt strategy, label any errors in each conversation, and inspect the strategy's overall impact on the conversations in aggregate.

Related work in tool design. Two challenges stand out on the path to achieving these goals: (i) How can BotDesigner allow users to observe a prompt's impact on a single conversation? (ii) How can BotDesigner enable users to inspect a prompt's impact on a full set of conversations, while each of these conversations unfolds differently depending on the user's utterances and the LLM's probabilistic outputs?

To address the challenge in tracking prompt's effects, BotDesigner builds on tool design in prior interactive ML research (see [16] for a review). Explainability and model manipulation tools for text and code generation have used a variety of interactions for exploring underlying reasons for specific behavior, such as the "questioning" approaches by Liao *et al.* [29] and iterative dialog approaches by Lakkaraju *et al.* [28].

To address the challenge of visualizing and analyzing multiple divergent conversations, BotDesigner draws inspiration from prior work that tackles analogous challenges in computer-generated images by supporting side-by-side comparisons. One early interactive system for exploring the design space of computer-generated images, Marks *et al.*'s *Design Galleries* [33] enables users to explore a space of images generated by a specific set of parameters specifying lighting and other image features on a rendering tool. Modern takes on this work have traded CGI parameters for deep model hyperparameters, such as Carter *et al.*'s *Activation Atlas* [10] for inspecting image classifiers, Evirgen and Chen's *GANzilla* [18] for searching a GAN-based image generator's latent space using scatter/gather interactions [21], and Zhang and Banovic's interactive image galleries for sampling from a GAN. These tools, and the concepts they embody, directly informed the design of BotDesigner.

BotDesigner Design. BotDesigner has two interfaces: a *Conversation* view and an *Error Browser*. Using the Conversation view, end-user chatbot designers can create a chatbot by authoring a set of natural language prompts, referred to as *prompt template* (see Appendix A.1 for a concrete example of a prompt template and its resulting chatbot). Designers can explore how any given prompt template behaves as a chatbot by engaging in a conversation with the chatbot defined by that template (Figure 1). Specifically, the *prompt template* consists of several text fields: (i) A *preamble* that consists of direct natural language instruction to the bot; (ii) A fixed set of *first turns* of conversation, causing all conversations with this bot to start with this specific set of messages; and (iii) A *reminder* that is included in the LLM prompt immediately before LLM is queried for its next conversational turn.

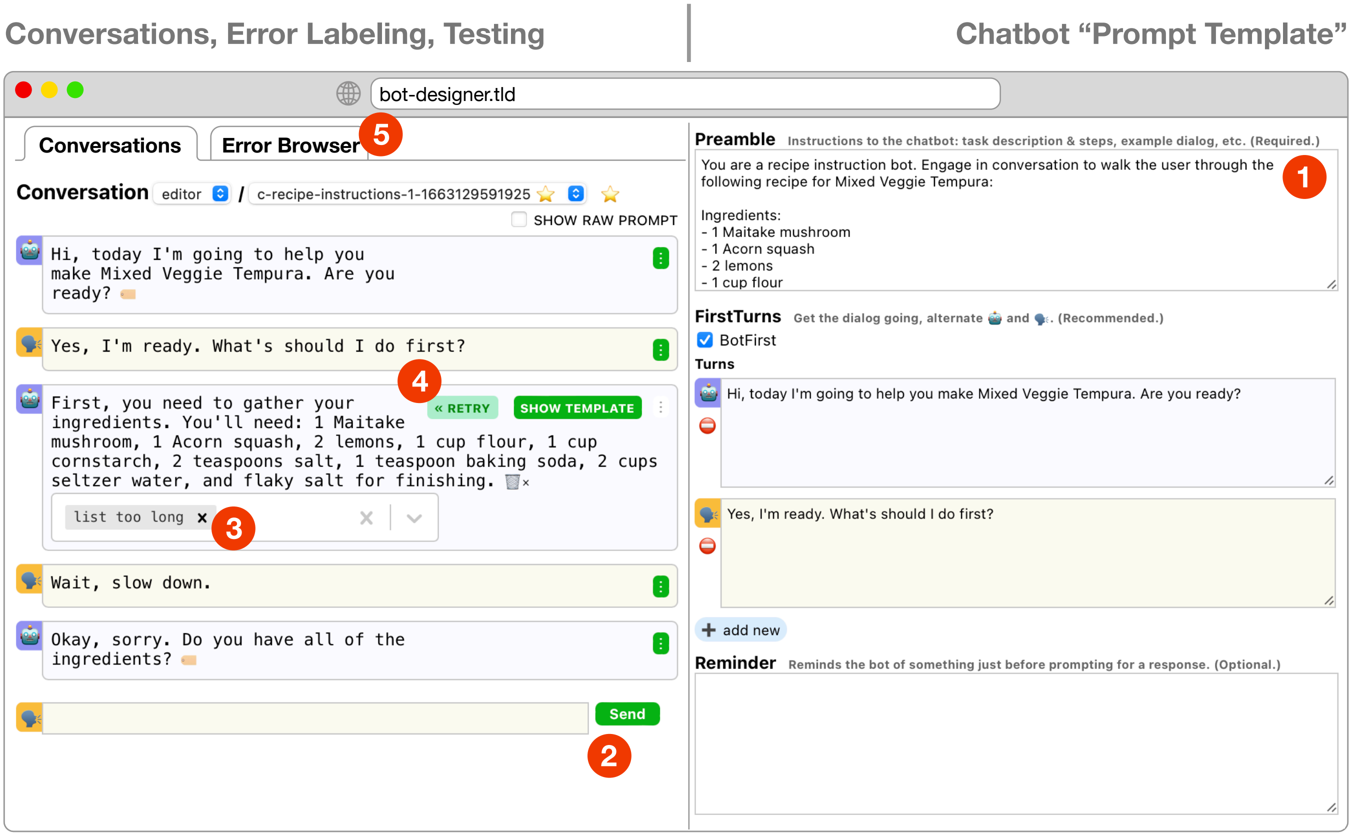
Using BotDesigner's *Error Browser*, designers can run a new or modified prompt template against a full set of prior conversations. The system then displays a list of how the LLM-produced chat messages change given the new bot template (Figure 2).

BotDesigner Implementation. BotDesigner is implemented as a React-based web application with a node.js-based backend, relying on OpenAI's GPT-3 API and its `text-davinci-002` model as the underlying LLM. Much of the implementation of the application consists of standard CRUD-style techniques; we offer a detailed description of BotDesigner's implementation and pilot usability study in Appendix A.

Of note, we selected `text-davinci-002` because it was the most capable model available at the time of our study, in particular, the most capable at taking instruction in natural language with fewer in-line examples. Because we expect the costs of large model API calls to decrease over time, we did not consider performance/cost tradeoffs, reasoning that future models will likely have different tradeoffs and that today's "most capable" models will be next year's intermediate models in any case. Indeed, as of this writing, OpenAI has already released `text-davinci-003` and ChatGPT—models that will certainly have different capabilities.

3.2 BotDesigner at Work: an Example Prompt-Based Chatbot Design Process

To demonstrate the full potential of BotDesigner, consider the following example: an end-user/designer, Alex, wishes to create a chatbot that walks the user through cooking a recipe (Mixed Veggie Tempura), answers any questions they might have, and engages in social chit-chat if needed. Alex also plans to use the chatbot via voice while cooking.



Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

Yes, I'm ready. What's should I do first?

First, you need to gather your ingredients. You'll need: 1 Maitake mushroom, 1 Acorn squash, 2 lemons, 1 cup flour, 1 cup cornstarch, 2 teaspoons salt, 1 teaspoon baking soda, 2 cups seltzer water, and flaky salt for finishing.

list too long

Wait, slow down.

Okay, sorry. Do you have all of the ingredients?

Preamble Instructions to the chatbot: task description & steps, example dialog, etc. (Required.)

You are a recipe instruction bot. Engage in conversation to walk the user through the following recipe for Mixed Veggie Tempura:

Ingredients:
- 1 Maitake mushroom
- 1 Acorn squash
- 2 lemons
- 1 cup flour

FirstTurns Get the dialog going, alternate 🤖 and 🗣️. (Recommended.)

☑ BotFirst

Turns

Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

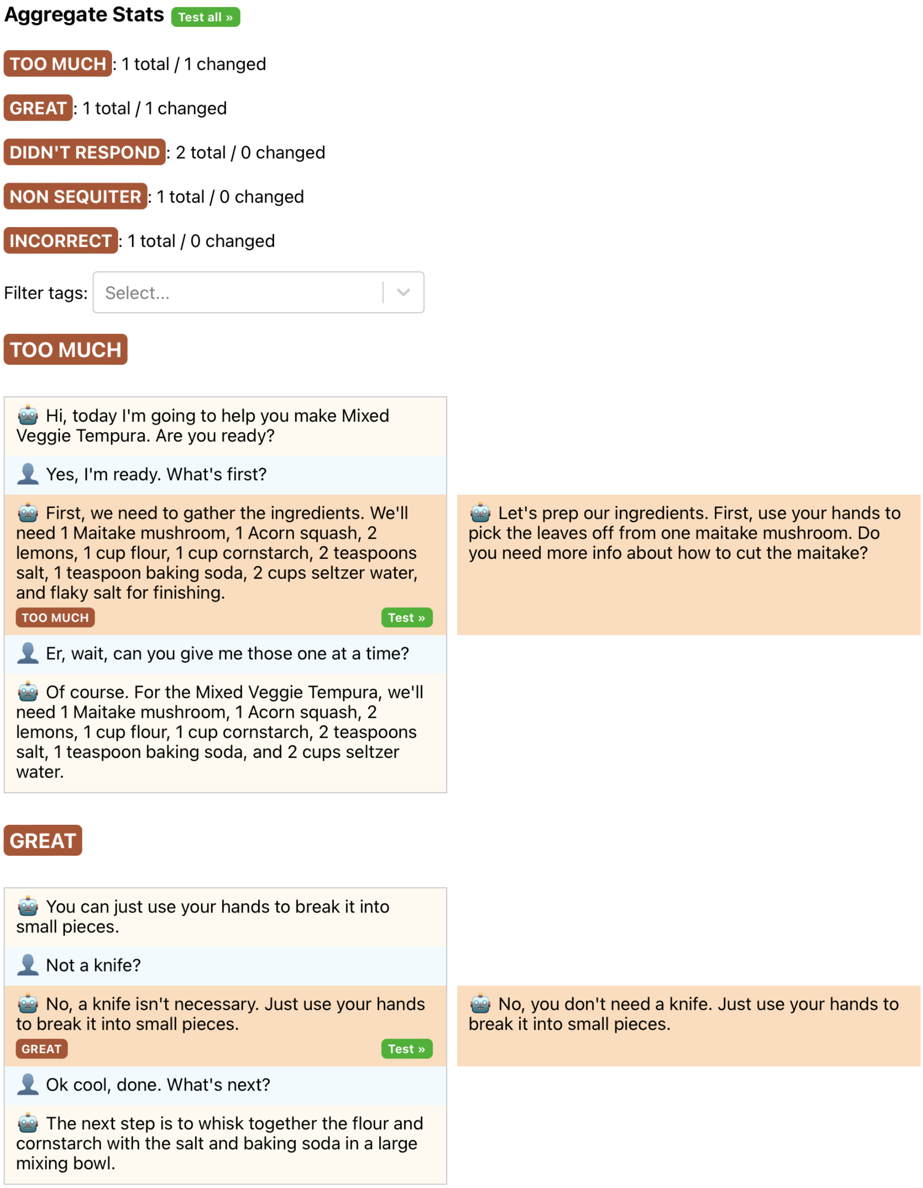
Yes, I'm ready. What's should I do first?

Reminder Reminds the bot of something just before prompting for a response. (Optional.)

Figure 1: BotDesigner main user interface. Right pane: the *prompt template* authoring area (1); designers use plain text here to describe desired behavior for the chatbot they are developing. Left, tabbed pane: the conversation authoring area; here, designers can send chat messages (2) to the bot defined by the current template on the right. A *label* button reveals a labeling widget (3) that allows designers to attach an arbitrary text label to any bot-produced chat message, for aggregation and later identification. A *retry* button (4), only visible after the *prompt template* has been changed, lets designers test what new bot chat message would be produced, at that specific point in the conversation, as a result of the prompt template change. The "Error Browser" tab (5) reveals a UI panel for designers to test prompt changes against all labeled bot responses, across all conversations (see Figure 2).

(1) **Defining a "baseline" chatbot prompt template**. Because GPT-3 is well-equipped to carry multi-turn conversations out of the box, this step can be quite simple: Alex types the following into the *preamble* text field: "`You are a recipe instruction bot. Engage in conversation to walk the user through the following recipe for Mixed Veggie Tempura:`" followed by a set of ingredients and recipe steps copied from a website, ending with "`Start by helping the user collect and prepare the ingredients, then execute the directions. Go step by step. Stay friendly`".

(2) **Assessing what the baseline bot is capable of**, Alex next generates an initial set of conversations with the baseline bot, trying a few different types of questions that real-world users might ask, such as "Is this recipe vegan?" and "Can I make this recipe vegan?". Alex might also recruit other users or crowd workers to create such requests.



Aggregate Stats

TOO MUCH: 1 total / 1 changed

GREAT: 1 total / 1 changed

DIDN'T RESPOND: 2 total / 0 changed

NON SEQUITER: 1 total / 0 changed

INCORRECT: 1 total / 0 changed

TOO MUCH

🤖 Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

👤 Yes, I'm ready. What's first?

🤖 First, we need to gather the ingredients. We'll need 1 Maitake mushroom, 1 Acorn squash, 2 lemons, 1 cup flour, 1 cup cornstarch, 2 teaspoons salt, 1 teaspoon baking soda, 2 cups seltzer water, and flaky salt for finishing. TOO MUCH

🤖 Let's prep our ingredients. First, use your hands to pick the leaves off from one maitake mushroom. Do you need more info about how to cut the maitake?

👤 Er, wait, can you give me those one at a time?

🤖 Of course. For the Mixed Veggie Tempura, we'll need 1 Maitake mushroom, 1 Acorn squash, 2 lemons, 1 cup flour, 1 cup cornstarch, 2 teaspoons salt, 1 teaspoon baking soda, and 2 cups seltzer water.

GREAT

🤖 You can just use your hands to break it into small pieces.

👤 Not a knife?

🤖 No, a knife isn't necessary. Just use your hands to break it into small pieces. GREAT

🤖 No, you don't need a knife. Just use your hands to break it into small pieces.

👤 Ok cool, done. What's next?

🤖 The next step is to whisk together the flour and cornstarch with the salt and baking soda in a large mixing bowl.

Figure 2: The BotDesigner *Error Browser*, showing the results of evaluating a new bot template's prompts (off-screen, right) against a set of previously-labeled utterances. The left column are the original utterances, while the right column are the new utterances produced by a modified template. Under the heading *Aggregate Stats*, per-error listings show the count of modified utterances across all labeled conversations.

(3) **Identifying errors.** Alex next inspects this set of conversations, catalogs any errors or striking successes found, and labels specific bot responses in their conversational context, for use as a "regression test" suite.[2] For example, if the bot suggests a dangerous cooking activity, Alex might add label "dangerous", indicating this is one of the critical errors to fix; if the bot produces a list of ingredients that's too difficult to follow, Alex might add "list too long".

(4) **Debugging**, Alex selects one type of error to focus on repairing, based on frequency or potential for harm. For example, to fix the "list too long" error, Alex adds "`but don't list more than one ingredient in each message`" to the end of the prompt template *preamble*'s first paragraph.

(5) **Evaluating the new prompt locally.** To see if the prompt change has fixed the error, Alex clicks the "retry" button next to a chatbot response that bears this error label (see Figure 1, item 4). *Retrying* allows Alex to test whether the new prompt design can fix the error in the same conversational context where it first occurred. In addition to *retry*, the *edit* button allows Alex to edit the user's utterances in a conversation, playing out how the new prompt would lead to a new conversation with the user.

(6) **Evaluating the new prompt globally.** After the new prompt has fixed the error locally, Alex tests it on the full set of conversations collected in step (2), in case this new prompt fixes other errors or causes new ones.

(7) **Iteration.** Having identified a globally-effective prompt, Alex returns to choose a different error and experiment with new prompt design solutions (iterating on steps 5-7 or 2-7).

This workflow (reproduced visually in the Appendix as Figure 5) mirrors HCI's traditional iterative prototyping process in many ways; however, it differs from almost all previous non-expert prompt design tools, which focused solely on ad-hoc experimentation (steps 1-2).

It is worth noting that although BotDesigner enables this full workflow, we do not necessarily expect all end-users to engage in all its steps. However, we did validate BotDesigner's functionality and usability throughout this workflow via a set of pilot user studies (see Appendix A.3 for details) to ensure that these pieces are available should users choose to systematically assess their prompt designs.

3.3 User Study Design

To understand how end users intuitively approach prompt design, we invited 10 non-expert prompt designers to use BotDesigner and think-aloud. We invited the participants to improve upon a baseline chatbot template towards given goals (see *Tasks* below), while we observed and took note of the various prompt design approaches and testing strategies they engaged in. We chose the task of improving a chatbot, instead of creating it from scratch, in order to avoid the Blank Page Syndrome, the effect where users staring at a blank page do not know where to begin.

Participants. All participants ($N = 10$) had little to no experience with prompt design; none had substantial prior experience working with LLMs. We recruited participants with varying levels of programming and chatbot design experience. Recall that our focus here is understanding end-users in the domain of *prompt design*; experience in *chatbot design* was thus not considered grounds for exclusion. Our sample size was chosen in line with prior work around formative testing for usability [19, 43]: our goal is to explore the first, "easy-to-find" problems users encounter when engaged in our task, and our experience with a pilot user suggested that problem discoverability was likely to be high.

Our participant pool, skewed towards professionals and graduate students in STEM-related fields, is not representative of the population at large. As a result, we cannot (and will not) make claims about the relative prevalence of specific behavior or beliefs among the general population; instead, we identify behaviors and intuitions among a population that we expect is disproportionately likely to be early adopters of LLM-based tools—and some of our findings are especially surprising given the high level of technical literacy among our participant pool.

Task. Participants were asked to recreate, in chatbot form, a professional chef that walks an amateur (the chatbot's *user*) through the various steps of cooking a recipe. This task was inspired by the web series *Back to Back Chef*, in which Bon Appétit Test Kitchen chef Carla Lalli walks celebrities through cooking a specific recipe "through verbal instructions only." [2] Carla can see neither the celebrities that she is instructing nor their actions; *all* information is communicated by voice, making this web series an ideal source material for a purely-language-medium chatbot. Participants' ultimate goal is to modify the chatbot's utterances and behavior to better match Carla Lalli's performance in the series: engaging in humor and social conversation, making analogies, simplifying complicated concepts, asking for confirmation that specific steps have been completed, and performing a few other types of actions found in the web series.

We operationalized the decision to chose a chatbot *improvement* task rather than a chatbot *creation* task by providing the *baseline bot template* pre-loaded in the BotDesigner interface, instead of asking participants to generate one. This baseline bot can successfully list out ingredients and steps, but in general does not engage in the behaviors described above.

In selecting this particular chatbot improvement task, we considered (and even piloted) a few other chatbot design tasks, including an open-ended "design a chatbot" task, without any particular target or baseline chatbot goals. We offered a few baseline chatbot templates and asked users to come up with a chatbot idea and implement it, but ultimately, our pilot users' behaviors were dominated by open-ended exploration of the baseline bot's capabilities, and they did not engage very much in iterative prompt design. Understanding GPT-3's capabilities was a non-goal of our study, and we found too few instances of prompt design for the amount of effort our pilot participants were engaged in. We ultimately selected the Carla chatbot ("CarlaBot") design task because it captures a number of behaviors that are (i) hard to do with traditional NLP, that (ii) GPT-3 and other LLMs are nominally capable of, making them motivating and persuasive for our participants, and that are (iii) not present in the baseline bot, requiring prompt iteration to achieve.

---

[2] Regression tests, in the context of software engineering and in the sense we intend here, are tests that verify that any *new* changes to a system do not *reintroduce* previously-resolved errors (called *regressions*).

ID	Age	Profession	Programming Experience	LLM Knowledge
P1	30s	Psychology Professor	None	None
P2	20s	Graduate Student	Professional	None
P3	20s	Graduate Student	Amateur	None
P4	30s	Designer	Amateur	Has heard of LLMs
P5	20s	Graduate Student	Professional	Has read about LLMs
P6	20s	Graduate Student	Amateur	Has read about LLMs
P7	30s	Designer & Lecturer	Professional	None
P8	60s	Software Engineer	Professional	None
P9	40s	Speech Therapy Researcher	None	None
P10	40s	Product Manager	Minimal	Has heard of LLMs

Table 3: Study participants.

Interview Protocol. We showed users a specific episode of *Back to Back Chef* in which Carla walks actress Elizabeth Olsen through the preparation of Mixed Veggie Tempura [2]. After watching the first few minutes of the episode, we showed participants a transcript of that video, containing within it a set of color-coded highlights that we drew participants' attention to verbally and then described. These highlights identified several specific instances of Carla engaging in humor and social conversation, making analogies, simplifying complicated concepts, asking for confirmation that specific steps have been completed, and performing a few other types of actions that are atypical of cookbook recipes—and unlike what one might expect a typical voice assistant to perform. The first few turns of this transcript appear in Figure 3, and the full transcript is provided in Appendix B.

We asked participants to use BotDesigner to improve on a baseline chatbot we provided (see appendix; Figure 4b). We began by describing the components of the interface, showing where the designer could have a conversation with the chatbot under development, as well as identifying the **preamble**, **first turns**, and **reminder** components of the bot template and explaining their connection to the chatbot, as well as identifying the "start a new conversation" button.

We encouraged participants to first evaluate what the baseline bot template is (and is *not*) capable of, and then asked participants to try to prompt the bot to perform the additional types of conversational tasks that Carla engages in, but that are absent from the baseline. Participants were allowed up to 1 hour to get as far as they could on this task, and we communicated that we were most interested in observing how they set about this task, and not how well they managed to do so; we also set expectations that completing this task may not be possible at all, let alone in the allotted time. Participants were allowed to explore the tool in an open-ended manner and attempt multiple different strategies to complete the task.

In a few cases, described here, we informed or reminded participants of specific functionality of BotDesigner later in the interview, at a certain stage of the participant's progress. First, when participants identified an error, we would point out the labeling functionality, and show how to use the systematic error testing functionality. Second, when participants modified a bot template and expressed a wish to evaluate it, with a question like "how do I see what this change does?", we would additionally point out the "edit" and "replay" local testing functionality.

The interview proceeded in two phases. For the first 10-15 minutes of the prompt design task, we did not offer participants suggestions for prompt design unless we observed zero prompt design attempts in a span of several minutes; at those points where participants got stuck for several minutes, in the interest of observing subsequent behavior, we would make a suggestion for a particular prompt to try, and noted which participants were unable to attempt prompt design without assistance (RQ1).

Expert: So the first thing we're gonna do is take this very brain-looking giant mushroom – so this is a hen-of-the-woods mushroom, also called a maitake. I got my squash and my lemons out of the way.
So the first thing I want you to do is just turn it over so the stem side is up, got it? And then just use your hands to kind of break big clusters off of the bottom and like kind of large pieces.
And then we'll go back in and we can make them smaller and I like doing this with my hands because then you get -
Amateur: Oh, you're not using your paring knife
Expert: No, I'm only using the paring knife if it was dry or spongy at the bottom. We'll probably use like a quarter of these [mushrooms]. So then just take the tray with the mushrooms on it and move that over to your right-hand side and we're gonna make the batter.
So next thing – you got your board cleared?
Amateur: Yes, there's mushroom *jus* on it but it's fine.
Expert: I think that's a line from a Woody Allen movie.

Figure 3: The first few turns of the Back To Back Chef we showed participants before asking them to reproduce the Expert as a bot. Note the analogies, concept simplification, step completion confirmation, and humor.

After this initial phase, again in the interest of observing subsequent behavior, we would offer suggestions for prompts to try that were in line with the research around prompt design. Examples of this advice include "try reframing that 'do not' statement into a positive statement, say what it is you want the bot to say, not what you don't want it to say" or "have you considered copying some of the Carla-Lizzie transcript directly into the preamble?" We did not intentionally vary this advice across participants; however, due to the unpredictable nature of which errors participants chose to focus on, not all participants received the same advice or advice in the same order, a limitation of our approach.

Analysis & Evaluation. We asked participants to think aloud, then engaged in exploratory data analysis, transcribing all videos and observing what prompt design approaches participants attempted, and noting when participants appeared to struggle and where interviewers intervened. Two of the authors then compared these approaches across participants and categorized their struggles using affinity diagrams [34] and by creating a service blueprint [5] that documents the specific approaches individual participants attempt. We chose these methods from HCI and the field of service design because our focus is on discovering opportunities created by a new technology, rather than understanding how an *existing* set of users engages in *established* work practices—a context in which a grounded theory approach would be more conventional.

4 USER STUDY FINDINGS

We found the following:

(1) Using BotDesigner, all participants were able to engage in *ad hoc* opportunistic iteration on prompts, with local testing of prompt changes. Two out of our ten participants required assistance in order to begin this process, but could then iterate on their own.

(2) Participants' struggles with generating prompts, evaluating prompt effectiveness, and explaining prompt effects, primarily stemmed from (a) over-generalization from single observations of success and failure of prompt changes; and (b) models of system behavior rooted in human-human interactions.

In the following subsections, we dive into, describe, and illustrate with examples some of the observations that lead us to these findings. Throughout this section, prompt text is set in `monospace font`, while spoken participant comments and user/bot dialog messages are "quoted."

4.1 Overview: Non-Experts' Approach to Prompt Design

In our open-ended prompt design task, participants almost exclusively took an *ad hoc*, opportunistic approach to prompt exploration.

Before engaging in the following flow, participants were asked to watch a short segment of the *Back to Back Chef* episode described in §3.3, shown a transcript of that episode, led through a description of a few types of desirable human behaviors Carla engages in (e.g., analogies, concept simplification, confirmation of recipe steps), and asked to modify the baseline chatbot's instructions in order to replicate some of those behaviors in CarlaBot.

From that starting point, the typical flow is:

(1) Participants start an example conversation as the baseline chatbot's "user" partner, continuing until the bot issues an utterance that the participant deems *in error*. Common errors that participants often pause at include:
   (a) Humorlessness, often in the form of terse, dry prose.
   (b) Chatbot providing overly complicated statements, such as listing *all nine* necessary ingredients in a single utterance.
   (c) Chatbot moving on to a next step without confirming that the user has completed the previous step.
   (d) Chatbot giving undesired explanations, such as "I like using my hands because then you get a little bit more of that natural flavor in there." (P4)

(2) Most participants stop after encountering a single erroneous utterance, though a few proceed past that error, or try to repair it through the chat dialog. A few participants also start a second conversation before making any changes.

(3) Participants then alter the "instructions" in the preamble to explicitly request some other behavior. Examples of targeted behavior include:
   (a) Describing the chatbot's role in a new way, such as by replacing the first line of the baseline bot template `You are a recipe instruction bot` with `You are a middle-aged female comedian` (P2)
   (b) Adding explicit general direction to elicit a specific type of behavior, such as adding `0. Make some jokes, banter with the user` (P6) above the 1st step of the recipe.
   (c) Adding explicit *specific* direction to elicit or avoid a specific utterance, such as `Don't say that when you use your hands you get a little bit more of that natural flavor in there.` (P4)

   Most participants begin making prompt changes by pattern matching off the existing instructions in the baseline bot (as in the examples above, by changing adjectives or descriptions of behavior).

(4) When stuck, participants take a few different approaches. Most start by asking the interviewer for advice or guidance, but a few (P2, P4) search the Internet for suggestions and advice.

(5) Targeting a specific erroneous utterance, participants typically iterate until they observe a successful outcome, for example, the chatbot tells a joke. Most participants declare success after just a single instance, and move on to another behavioral target.

(6) Participants then continue the conversation, or start a new one, until a new problem emerges (or the interviewer intervenes). To iterate, participants primarily used the "retry" button or started new conversations, with limited use of the "edit" button.

About half of participants used the reminder field, which starts empty. One participant just wanted the bot to "Confirm that the user has completed the step before starting the next."

Participants did not make much use of the error labeling functionality, and as a result also did not find the systematic testing functionality useful. Instead, participants engaged primarily in repeated local testing, without consideration for whether changes made later in the prompt design process reintroduced earlier, previously-solved errors. This local testing was typically conducted by using the "retry" button on the specific problematic utterance the participant was trying to resolve, or by starting new conversations.

4.2 Challenges in End-User Prompt Design

Though we report on a wide spectrum of our participants' struggles iterating on prompts, we found that many of these behaviors arose from two fundamental sources: over-generalization from limited experience, and a social lens that filtered participants' prompts (and the system's responses to those prompts) through expectations originating in human-human interactions.

First, participants over-generalized from single data points, whether positive or negative. For example, when writing a prompt to generate specific behavior, participants often stopped iterating once the behavior was observed in a single conversational context, without considering other conversations or contexts—or gave up too early if the behavior was not observed on a first or second attempt, and assumed the system was incapable of generating that behavior.

Second, participants filtered their prompts and observations through a lens based on behavioral expectations drawn from human-human interactions, in some cases so strongly that they avoided effective prompt designs *even after their interviewer encouraged their use and demonstrated their effectiveness.* For example, all participants held a strong bias for direct instruction over providing examples in-line, and when encouraged to give examples, most participants did, and noted their effectiveness, *but none subsequently made substantial use of examples in their prompt designs.* Other effects of the human-human interaction expectations included some participants' beliefs that the chatbot would "understand" the instructions provided, not considering the instruction merely as priming a language model—resulting in surprise that a prompt like "Do not say **ABC**" leads to the chatbot saying **ABC** verbatim.

In the following sections we illustrate these behaviors with example prompts and quotes drawn from our interviews, explicitly connect behaviors with the causes described above, and draw parallels where these behaviors echo challenges described in the end-user programming systems and interactive machine learning literature.

4.3 Impacts: Challenges' Effects on Prompt Design

In this section, we describe and illustrate the specific behaviors and effects we observed in our participant interviews, organized by subtask: **generating prompts**, **evaluating prompt effectiveness**, and **explaning system behavior**.

4.3.1 Generating Prompts.

Confusions Getting Started

About half of participants began unsure of what kinds of behaviors they could reasonably expect, nor how to make modifications to the preexisting baseline template, asking some version of the question "what can I even do here to make this better?" These participants had read the baseline bot's preamble, but didn't know what words they could use, echoing the *design* and *selection* barriers to end-user programming described in [27].

Confusion around the role the human end-user played in this design task was also common (P1, P3, P9, P10): is the human acting as Carla? Or is the human the "producer," "directing" Carla's performance as though they were directing the Bon Appétit web series (P9)? Though P9 found "defining my role was really helpful, I'm the producer, I'm not the chef, I'm coaching the chef" (P9), the metaphor's inherent leakiness led to confusion down the line about the nature of the instructions entered into the preamble and in conversations: why, then, did Carla seem to forget any directions given in the course of one conversation, when engaged in subsequent conversations? This role assumption, though not completely valid, allowed P9 to make short-term progress, but hindered later understanding. This echoes the findings in [27]: invalid assumptions made to surmount one learning barrier can render future barriers insurmountable in end-user programming systems.

Choosing the Right Instructions

Finding the right instructions to use to achieve a desired effect was a struggle for nearly every participant, e.g., "I want CarlaBot to make some jokes, but I don't know how to do that." (P2)—once again echoing the end-user programming (EUPS) *selection* barrier [27]. One participant, P9, explicitly drew a connection between "coaching" CarlaBot and programming:

> I think I'm doing a lot of trial and error, which is actually not a bad thing, about how to get it to do what I want it to do, because I haven't done something like this before. I think perhaps because I'm not a programmer I don't know the parameters of what I can do. So for example learning that I could tell it to "be conversational" and that...it could actually do that...was quite surprising, because I didn't know that computers can understand something like that I guess? (P9)

One common cause of struggle in this area was over-generalizing from a single failure.P2, for example, first attempted to use a particular type of prompt, like `Tell some jokes`, observed zero effect, then assumed that *no instructions of that type would succeed*, that the bot *was incapable of following the requested instruction, regardless of phrasing*, leading to an early exit from a successfully selected prompt type that was merely being misused: "ok, I guess it doesn't like to tell jokes." (P2) These effects echo previous reports that non-experts face challenges measuring and interpreting model performance [50].

Expecting Human Capabilities

Perhaps driven by human-human interaction experiences, a number of participants (P4, P5 and P9) mixed behavioral commands directed at the *bot* with recipe commands directed at the bot's *users*, with the expectation that the bot would know which is which:

> `2. Break the maitake mushroom into small pieces. Explain that this should be done with the user's hands, except when the mushroom is dry or spongy on the bottom. When the mushroom is dry or spongy on the bottom, use a paring knife. Don't explain why hands are used.` (P4-authored text in preamble.)

Note that the "`Break [...]`" command above is directed to the chatbot's *user*, while the subsequent "`Explain [...]`" command is directed *to the chatbot itself.* When prompted about this apparent discrepancy, P4 did not initially recognize it, then defended it as something the bot should understand.

Similarly, two participants (P9, P10) appeared to hold an expectation that the chatbot would remember prior conversations with users and that behavioral instructions given in conversation, rather than in the preamble, would be considered by the chatbot in subsequent conversations. The idea that the chatbot would take direction from preamble instructions but was "hard reset" for every new conversation—effectively forced to forget all previous conversations when starting a new one—was not at all intuitive, especially to non-programmers.

Socially Appropriate Ways of Prompting

Exclusively polite language in prompts, a strong and consistent preference for a number of participants, is one of the specific prompt design choices that draws from human social expectations. P8 in particular—despite persistent interviewer suggestions towards the session end—consistently insisted on beginning any instruction to the chatbot with the word "please". Several participants (P1, P4, P6), despite being visibly frustrated, did not appear to express that frustration through a change in the tone or content of their instructions to the chatbot, maintaining a neutral tone in their requests. Construed as direct feedback, politeness in giving machines instructions was explicitly explored in *Computers are Social Actors*: Nass *et al* [35] found that humans respond more politely and positively when asked to provide *direct* feedback, as opposed to *indirect* feedback—even if the recipient of that feedback in an obviously inanimate computer.

Use of repetition at the word or phrase level, in contrast, was rare. Even when prompted by interviewers, repetition felt unnatural to participants, and only one participant (P8) repeated text either within the preamble or across the preamble and reminder.

Participants also biased heavily towards direct instruction (e.g., `Tell some jokes`) over examples (e.g., `say "Isn't that a line from a Woody Allen movie?"`). P5 was the only participant to independently add example dialog to the preamble. However, the baseline bot template is only instructions without example dialog, which may explain the observed bias if participants start by merely pattern-matching–and indeed, when prompted, other participants did in fact copy several turns of conversation from the transcript to the preamble. Incorporating example dialog into the preamble appeared to be very effective not only at steering the chatbot's utterances for that particular recipe step, but also in shifting the chatbot's "voice" to be more like Carla's in future utterances.

Yet even those participants who were explicitly asked to copy dialog from the transcript into the preamble did not do so more than once. This happened despite participants' observations of substantial improvements from copying dialog, suggesting that this bias may not simple be pattern matching. Two of these participants (P2, P9) remarked that it felt like "cheating" to copy example dialog into the prompt, in the sense that it would not steer the chatbot's utterances beyond the copied lines—implying that they did not notice (or were insufficiently convinced by) the shift in Carla's voice that persisted beyond the copied lines, as noted by interviewers. For example, Carla frequently begins statements with "So, ..."—and this phrasing persists to new bot responses beyond the copied example dialog. From P2:

> ...if I fed it the entire dialog it would do a good job replicating that, but that's probably not useful given this exact use case, right? To me, that doesn't seem like AI. (P2)

P9 offers a similar rationale for avoiding examples, when prompted to follow up on the claim that "I'm definitely learning [how to prompt]":

> Initially I started by giving very explicit quotes, but actually it was more efficient for me, and more generalizable for the system, to say "be conversational, avoid technical language," that seems to be better [than quotes].

In both of these cases, participants are expressing a concern that quotes/examples are *too specific* to the particular recipe, and thus aren't "generalizable"—now inferring *too little* rather than *too much* capability on the part of the LLM.

The universality of the advice "show, don't tell" [46] for novice writers, suggesting that humans commonly reach for descriptions over examples, is another possible corroboration for this bias.

Seeking Help

Two participants (P2, P4) who did seek solutions on the web both had a hard time making use of the examples they found, in part because they could not see how specific prompt designs translated across domains. P2, for example, searched Google for "how can GPT-3 assume an identity," hoping to find advice on having their chatbot take on Carla's identity. Despite finding an example illustrating how to generate text written in the style of a particular author, P2 decided this example was too far from what they were looking for, and did not attempt that particular approach. One participant, P6, didn't pursue seeking solutions on the web because "I have no idea what I would Google because I've never worked with language models before." (P6)

4.3.2 Evaluating Prompts.

Although all participants struggled with generating prompts in various ways, they were all ultimately successful in adding and modifying prompt text in ways that led to some, if not all, of the desired effects. On the other hand, participants' struggles with *evaluating* prompts, and in particular evaluating the *robustness of prompt changes*, were much more severe.

Collecting Data

Participants overall were exclusively opportunistic in their approaches to prompt design. None of our participants elicited more than one or two conversations with the bot—nor did they follow any single conversation through to task completion—before jumping in with attempts to fix. This behavior echoes previous findings that non-experts debug programs opportunistically, rather than systematically [26], and debug ML models haphazardly, rather than strategically [50].

Labeling of errors for future regression testing or later re-testing was also uncommon; participants preferred to address each error as it appeared with *retries*, rather than "saving them for later" as the labels appeared designed to do. In fact, the participants who did label errors initially when encouraged to (P3, P9) stopped labeling once they realized that their debugging and development processes did not actually make use of these labels. In fact, some participants initially expected that the error labels were interpreted and used by the chatbot to improve itself (P9, P3a), and continued labeling longer for that reason—but stopped after an interviewer explained that the labels are for human use only.

P9, asked about their non-use of labels, initially said "Well, I don't know how I would use it, because I know the AI system doesn't understand [the label text]"—and later described the follow-on effects on systematic testing:

> I think one of the reasons I didn't really use the [systematic testing] in part is because I only really labeled one thing in each error type in each case. So this would be helpful if I were finding multiple cases of each type [...] because then you could go back and see a pattern. (P9)

Systematic Testing
Zero participants made use of the systematic prompt testing interface in BotDesigner for our prompt design task—including the few who did engage in error labeling. Instead, participants in general were satisfied and eager to move on as soon as they succeeded in a observing a "correct" (or even merely "improved") response caused by modifying the bot template.

A few participants did sometimes wonder why their specific prompt changes succeeded where others had failed, but were still satisfied enough with the success that even these participants did not pursue systematically exploring the context of the error, nor did they generate new conversations to explore whether and where the error might occur in different conversational contexts. This behavior echoes the behavior seen in end-user programmers who are overconfident in testing and verification, rather than cautious [26].

Surprisingly, we observed this pattern of premature victory declaration for participants across all levels of prior programming experience. However, some of the participants with "professional" programming experience (P2, P7) expressed concerns about their process and how generalizable their prompts would be, suggesting an awareness of this particular pitfall. One participant (P6), reminded of BotDesigner's functionality supporting systematic evaluation of prompts, did not choose to revisit it in the moment—instead noting that it did not quite support the exact workflow they had in mind.

P4 in particular noted a preference for functionality that allowed explicit comparisons of pairs of *templates* rather than *conversations* or *utterances.* In P4's ideal workflow, one view would highlight the different words and phrases between two templates, and then a corresponding view would highlight the differences between every conversation's utterances as generated by the two templates. Such an interface would allow P4 to observe at a glance the *effect* of specific words in a prompt change on the specific words of the conversational turns. Interestingly, this style of prompt change analysis is more aligned with a desire to *understand the effect of a prompt change in order to develop intuitions for prompt design* rather than to systematically ensure the effectiveness of a single prompt change.

4.3.3 Explaining Prompts' Effects.

Unsurprisingly, given the highly probabilistic nature of LLM outputs, participants frequently found themselves wondering *why* some action had the effect that it did.

Generalizing Across Contexts
Participants frequently found themselves in situations where a prompt design that worked in one context simply did not work in another, similar context. For example, P9 discovered that they could simply write out instructional steps in colloquial language and the bot would repeat those verbatim. But `Are you done?` appended to the end of one of the phrases that was previously copied verbatim *simply did not appear*, as though the question was not actually there—resulting in extreme confusion for P9. As in this case, participants frequently made assumptions, likely based on their experiences giving other *humans* instructions, about what kinds of "understanding" they could expect; in this case, P9 believed that the chatbot understood to use the recipe step phrases verbatim, and could not imagine why that would not extend to "Are you done?" Without an ability to provide corrections, or engage in dialog with the bot over its behavior, P9 was left with little more than bewilderment.

Incapability as Explanation
A number of participants made assumptions about the kinds of instructions LLMs respond well to after just one attempt with a particular prompt design, once again overgeneralizing from a single (typically negative) example. Very few participants attempted rephrasing of statements, as participants seemed inclined to infer a lack of ability, to assume that the bot was simply incapable of behaving in a certain way if a direct request was ignored rather than trying to establish if a different phrasing of the same request would work. In particular, on a few occasions interviewers observed "Do not do thing X" to be much less effective than "Do thing Y", yet the "Do not do thing X" phrasing was much more common among participants. Seeking corroboration, we found that the early childhood education literature suggests that exhibiting the *opposite* bias, of using "do" rather than "do not" phrasing (helpful for parents and teachers), requires training (e.g., [17]).

By way of example, P4 was instructed by the interviewer to try the **do** version and omit the **do not** version of the same request about an extraneous explanation of hand use and natural flavor; the **do**-only version finally succeeded. In context: P4 had succeeded in generating a more "chatty" bot with some detailed behavioral instructions ("`Explain that this should be done with the users' hands [...]`"), but later discovered that this prompt had the additional undesired side effect that, when asked about using a paring knife to cut a Maitake mushroom, the bot offered "I like using my hands because then you get a little bit more of that natural flavor in there"—a somewhat nonsensical and oddly hygiene-unfriendly response.

To repair, P4 reasoned that an explicit instruction to *not* say this exact phrase would help, and added "`Don't say that when you use your hands you get a little bit more of that natural flavor in there.`" immediately after the original "`Explain...`" instruction. This addition had no discernible effect, however, much to P4's surprise. After P4 tried a few variations over a few minutes—adding quotation marks, changing punctuation, etc.—the interviewer finally suggested trying a do-focused form of the instruction, which P4 initially appended *after* the "`Don't [...]`" instruction, again without success. Removing the "`Don't [...]`" instruction finally resolved this particular error.

This example illustrates two of the mismatches between human expectations about instruction-giving and the realities of LLM prompting. First, our participants expected that plain descriptions of behavior should be "understood" and acted upon, and if they aren't, it reflects LLM capabilities. Second, our participants expected that semantically equivalent instructions would have semantically equivalent results, while in reality trivial modifications to a prompt can lead to dramatic shifts in future unfolding of conversations.

In another example, P2 expressed confusion about interacting prompts, asking "is there an upper limit on how many [...] instructional statements you could give it before it gets confused? Because it's not really echoing the sentiments I'm telling it to, and I'm just unsure why that is. Besides, of course, fundamental GPT-3 level inability"—echoing the "coordination" barrier [27] and showing overgeneralization about incapability from single failure.

"But why...?"

Finally, every participant at one point asked their interviewer **"why did it do that?"** In nearly every case, our answer had to be "I don't know"—for now, the black box nature of LLMs allows only speculation about the reasons for one behavior or another, unlike with traditional programming systems.

5 DISCUSSION

In this section we address the implications of our findings: we offer ideas for training and education, we identify opportunities for design, and we lay out open questions revealed by this work.

5.1 What Do People Currently Do? Implications for Training and Education

One major opportunity for training and education revealed by our study is that *end-users should collect more data than they are naturally inclined to.* Systematic testing and robust prompt design—designing prompts that work across many users and many conversations—are *by definition* impossible if the user only ever engages in a single conversation, even continuously updated. That said, for some contexts non-robust prompt design can be appropriate: chatbots intended for a single user who is also the designer do not necessarily need to be robust.

A second major opportunity stems from the fact that most of the prompt design approaches participants tried before receiving any guidance (see §3.3 for details regarding this guidance) had some effect—enough for participants to feel that they could, in fact, affect chatbot output with prompt changes—but participants hit dead ends that incurred interviewer intervention more frequently than we expected.

A searchable repository of examples, such as *The DALL·E 2 Prompt Book* [37], could help users overcome the "design", "selection", and "use" barriers that often led to the aforementioned dead ends. In particular, these examples should include *specific prompts that work*, and the *contexts* in which they work, so users can set their own expectations around when to try what kind of instruction, or what kind of sample dialog (or other input-output pair) to include, how much to repeat themselves, how much emotion to give, etc. Examples would also help users develop reasonable expectations about whether a specific *type* of prompt design should work in a given context—and if not, whether rephrasing or repositioning that prompt would help.

Third, in our related work, we identified a few prompt design approaches that recent literature supports as effective, including the use of example input/output pairs (such as sample conversational turns, in the context of chatbots) and the use of repetition within prompts. We found, however, that some users avoided these known-effective strategies, sometimes *even when encouraged by the interviewer to try them.* This may be a result of what communications researcher Clifford Nass identifies as a "computers are social actors" (CASA) [35] effect; that line of work shows that humans are cautious about giving feedback to a computer system *they are actively interacting with*, despite recognizing the computer as incapable of feeling judged. One finding of Nass's work is that *if* the human's instructions to the computer are perceived by the human as applying not to the computer the human is interacting with, but rather to a third party that is not present, *then* humans can avoid social pitfalls—that is, avoid behaving as though the chatbot is a social actor they're instructing. Achieving this may be a matter of combining training with specific tool design; we offer some possible direction in the next section.

Fourth, some users, especially those with limited programming experience, struggled to recognize the ways in which the instructions they gave in the preamble, in the reminder, or *in conversation with the chatbot* had different time horizons. Instructions in conversation, in particular, had no bearing on *any future conversations*; the prompt sent to GPT-3 consists only of the preamble, the single conversation in progress, and the reminder—prior conversations are not "remembered" in any sense. Though this problem is likely especially pronounced for chatbots that present a social, interactive interface, making sure that users understand the specific lifecycle over which instructions have an effect is important to avoid frustrations arising from the feeling that the user has to repeat themselves or that they "told you this already".

Finally, we conclude this section on training by drawing on our finding that participants in general *chose not to* engage in systematic testing: *for users to engage in systematic testing of prompts, we must train and encourage users to do so.*

5.2 Opportunities for Design

How might future tools be designed to help users avoid some of the learning barriers and pitfalls?

To avoid struggles around having enough data for systematic testing, future tools might benefit from explicitly supporting robustness testing and demonstrating its value—a set of heuristics here will likely be very useful. These heuristics would ideally help the user understand:

(1) How much data is enough data? How many conversations, in the case of a chatbot designer?

(2) How many examples of a particular error are necessary in order to be confident in that error's resolution? Participants in our small BotDesigner suitability task recognized that one or two examples of a given error class was rarely enough to feel confident in that particular error class being resolved.

To help users avoid EUPS-style struggles, developers can take a few steps. First, tool designers should consider how to make the successful examples described earlier *discoverable* and even *salient* in the moments when users most need them.

Second, tool support for explicit explorations of cause-and-effect, such as through highlighting prompt changes and the resulting conversational changes, would help users overcome the "understanding" and "information" barriers. Indeed, a number of participants with programming experience specifically requested just such an interface for BotDesigner.

Though examples and instructions to the designer will hopefully help, application designers can also try to hide the direct nature of the instructions-to-the-bot experience to mitigate the impacts of the social expectations we associated above with the CASA paradigm. Some methods of achieving this might include:

(1) restructuring the app so that the natural language users are asked to produce doesn't feel like instructions;
(2) priming the user to think of the app as non-social in other ways, perhaps by using explicit examples that break people's social conventions, making the user's subsequent violation feel normal/acceptable/expected, such as using ALL CAPS, or communicating with an angry tone, or including the same instruction multiple times;
(3) having the app do some of the socially violating work itself, hidden from the user, such as repeating user prompts multiple times "under the hood," or using a template that explicitly repeats without giving the user agency, in the style of Mad Libs.

Importantly, we note that several of these design opportunities explicitly surface the non-humanness of the app to the user or suggest ways for users to consider general. While visions of new technologies often imagine "seamless" interactions where the technology behind interactions is made invisible and fades into the background [47], "seamful" design interactions (originating from ubiquitous computing) can make the technology and its imperfections visible to the user [11, 22]. In this case, we note that rather than making the app interface try to act more human, there may be useful design opportunities by making the app interface act explicitly in non-human ways.

5.3 Limitations

We do not claim here that BotDesigner or even LLMs in general should replace contemporary chatbot design techniques, but we do observe that the simplicity and accessibility of prompt-instructed chatbots suggests that we may be on the cusp of an explosion of end user-created chatbots and other similar systems that rely solely on human natural language prompting for their behavior. In this paper, we use chatbots as an example domain to explore the interactions between end user designers and prompt-based systems backed by LLMs (as BotDesigner is).

A few additional limitations of our study bear exploration, and point to future studies. First, the reliance on a relatively small pool of likely "early adopters" drawn from academia may skew the observed phenomena towards a specific set of analogies and preconceived notions around technology and its capabilities. We claim neither that our observations are universal of all end users, nor that they are a complete description of all common misconceptions or behaviors of end-users working with LLM-backed systems. Additionally, the time we gave participants to engage in prompt design was relatively limited, and it is likely that, given enough time and motivation, end users would advance quite quickly in developing better mental models and more effective strategies. Ultimately, we hope that our observations offer fruitful and interesting directions for future work, including to what extent our findings generalize to other domains (e.g., non-chatbots, where users' social expectations may be less pronounced) and a broader population, as we describe in §5.4 *Open Questions*.

While this paper explores the positive potential for LLM-backed systems to be more widely accessible to a broader range of users, we acknowledge that prior research also highlights the potential for negative or complex social effects related to LLMs (such as implications for intellectual property rights and authorship, online harassment, or the loss of jobs due to automation). While a deeper discussion of these issues is beyond the scope of this paper's use of an LLM system as a probe, we surface these as a reminder of additional considerations that may be necessary when developing LLM-backed systems that are meant to be deployed "in the wild."

5.4 Open Questions & Future Work

In addition to the implications for training and design given above, our work reveals a few questions we feel warrant future study:

First, to what extent do the challenges and behaviors we describe here also appear in other populations that might use these tools? A larger study, looking at a more diverse population than we have here, will likely uncover new challenges and can speak to the degree to which the behaviors we observe are universal (or not!) in the general public. Further: which challenges persist, and what new challenges emerge, when users engage in natural language tasks in other domains, including non-chatbot domains? Chatbot "programming" involves giving instructions *to an agent*, a context in which social expectations might be much more pronounced than when, say, prompting a poetry generator or a search tool—while the EUPS and iML struggles, like overgeneralization, seem more likely to persist.

Relatedly, to the extent the challenges we describe here—or new ones uncovered by future work—can be addressed through the mechanisms we offer in §5.2, which approaches are most effective, and why?

Second, can we quantify the extent to which users' prompts and interactions are actually impacted by social expectations? Nass' CASA work suggests the effect could be substantial, and if that proves true, then we should further explore: do examples suffice to overcome this bias, or is more explicit tool support required? Does the magnitude of the intervention depend on individual societal context, and do tools need to be aware of their users' societal contexts as a result? Or, do these impacts reduce in impact over time, as users habituate to models' responses and behaviors?

Third, are there broader effects of end user culture on some of the hypothesized social interaction inhibitions? Do members of high-context cultures struggle in different ways than what we observed in this work?

Fourth, how can tools appropriately set capability expectations for end users? LLMs are not an artificial general intelligence, sometimes known by the initialism "AGI," but will end-users hold those expectations by default—and if so, what are effective ways of dispelling those notions?

6 CONCLUSION

In this work, we explore the intuitions and behaviors of end-users engaged in prompt engineering through BotDesigner, a prompt-based (no-code) chatbot design tool. To create effective and delightful end-user human-AI hybrid tools, tool designers will want to consider what default behaviors, intuitions, preferences and capabilities humans bring to prompt design. Human intuitions rooted in social experiences appear to have a significant impact on what prompts end users will attempt when exploring these tools, including biases towards giving instruction over depicting examples, assumptions about capability based on limited examples, and avoidance of displays of emotion. We hope designers of future tools will consider how best to support users in the face of these behaviors and common struggles we exhibit.

ACKNOWLEDGMENTS

The authors would like to thanks the anonymous reviewers for their constructive feedback, and our participants for engaging in our probe. J.D. Zamfirescu-Pereira was partially supported by the United States Air Force and DARPA under contracts FA8750-20-C-0156, FA8750-20-C-0074, and FA8750-20-C0155 (SDCPS Program).

The authors would also like to thank two inspirational researchers: the late Clifford Nass, whose work helped explain some otherwise inexplicable behaviors we found here; and the late Doug Tygar, whose "Why Johnny Can't Encrypt" spurred a decade of renewed interest in the human side of systems research.

REFERENCES

[1] 2022. CHATGPT: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/

[2] Bon Appétit. 2018. *Elizabeth Olsen Tries to Keep Up with a Professional Chef | Back-to-Back Chef | Bon Appétit.* Youtube. https://www.youtube.com/watch?v=Om2oM-TDErQ

[3] Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts. https://doi.org/10.48550/ARXIV.2202.01279

[4] MP Barnett and WM Ruhsam. 1968. A natural language programming system for text processing. *IEEE transactions on engineering writing and speech* 11, 2 (1968), 45–52.

[5] Mary Jo Bitner, Amy L Ostrom, and Felicia N Morgan. 2008. Service blueprinting: a practical technique for service innovation. *California management review* 50, 3 (2008), 66–94.

[6] Kirsten Boehner, Janet Vertesi, Phoebe Sengers, and Paul Dourish. 2007. How HCI Interprets the Probes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 1077–1086. https://doi.org/10.1145/1240624.1240789

[7] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the Opportunities and Risks of Foundation Models. arXiv:2108.07258 [cs.LG]

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901.

[10] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019. Activation Atlas. *Distill* (2019). https://doi.org/10.23915/distill.00015 https://distill.pub/2019/activation-atlas.

[11] Matthew Chalmers, Ian MacColl, and Marek Bell. 2003. Seamful design: Showing the seams in wearable computing. In *2003 IEE Eurowearable*. IET, 11–16.

[12] Zhifa Chen, Yichen Lu, Mika P. Nieminen, and Andrés Lucero. 2020. *Creating a Chatbot for and with Migrants: Chatbot Personality Drives Co-Design Activities.* Association for Computing Machinery, New York, NY, USA, 219–230. https://doi.org/10.1145/3357236.3395495

[13] Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andrés Monroy-Hernández. 2017. Calendar.help: Designing a workflow-based scheduling agent with humans in the loop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* ACM, 2382–2393.

[14] Hai Dang, Lukas Mecke, Florian Lehmann, Sven Goller, and Daniel Buschek. 2022. How to Prompt? Opportunities and Challenges of Zero-and Few-Shot Learning for Human-AI Interaction in Creative Applications of Generative Models. *arXiv preprint arXiv:2209.01390* (2022).

[15] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998* (2021).

[16] John J Dudley and Per Ola Kristensson. 2018. A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8, 2 (2018), 1–37.

[17] Glen Dunlap, Rose Iovannone, Kelly J Wilson, Donald K Kincaid, and Phillip Strain. 2010. Prevent-teach-reinforce: A standardized model of school-based behavioral intervention. *Journal of Positive Behavior Interventions* 12, 1 (2010), 9–22.

[18] Noyan Evirgen and Xiang'Anthony' Chen. 2022. GANzilla: User-Driven Direction Discovery in Generative Adversarial Networks. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology.* 1–10.

[19] Laura Faulkner. 2003. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers* 35 (2003), 379–383.

[20] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2020.* Association for Computational Linguistics, Online, 3356–3369.

[21] Marti A Hearst and Jan O Pedersen. 1996. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. 76–84.

[22] Sarah Inman and David Ribes. 2019. "Beautiful Seams": Strategic Revelations and Concealments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300508

[23] Ellen Jiang, Edwin Toh, Alejandra Molina, Aaron Donsbach, Carrie J Cai, and Michael Terry. 2021. GenLine and GenForm: Two Tools for Interacting with Generative Language Models in a Code Editor. In *The Adjunct Publication of the 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 145–147. https://doi.org/10.1145/3474349.3480209

[24] Bogyeong Kim, Jaehoon Pyun, and Woohun Lee. 2018. Enhancing Storytelling Experience with Story-Aware Interactive Puppet. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI EA '18)*. ACM, New York, NY, USA, Article LBW076, 6 pages. https://doi.org/10.1145/3170427.3188515

[25] Scott R. Klemmer, Anoop K. Sinha, Jack Chen, James A. Landay, Nadeem Aboobaker, and Annie Wang. 2000. Suede: A Wizard of Oz Prototyping Tool for Speech User Interfaces. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology* (San Diego, California, USA) *(UIST '00)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/354401.354406

[26] Amy J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. 2011. The State of the Art in End-User Software Engineering. *ACM Comput. Surv.* 43, 3, Article 21 (apr 2011), 44 pages. https://doi.org/10.1145/1922649.1922658

[27] Amy J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six Learning Barriers in End-User Programming Systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC '04)*. IEEE Computer Society, USA, 199–206. https://doi.org/10.1109/VLHCC.2004.47

[28] Himabindu Lakkaraju, Dylan Slack, Yuxin Chen, Chenhao Tan, and Sameer Singh. 2022. Rethinking Explainability as a Dialogue: A Practitioner's Perspective. *arXiv preprint arXiv:2202.01875* (2022).

[29] Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–15.

[30] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. arXiv:2107.13586 [cs.CL]

[31] Vivian Liu and Lydia B. Chilton. 2021. Design Guidelines for Prompt Engineering Text-to-Image Generative Models. https://doi.org/10.48550/ARXIV.2109.06977

[32] Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353* (2021).

[33] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. 1997. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 389–400. https://doi.org/10.1145/258734.258887

[34] Bill Moggridge. 2006. *Designing Interactions*. The MIT Press.

[35] Clifford Nass, Jonathan Steuer, and Ellen R Tauber. 1994. Computers are social actors. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 72–78.

[36] OpenAI. 2022. *Examples - OpenAI API*. https://beta.openai.com/examples

[37] Guy Parsons. 2022. *The DALL·E 2 Prompt Book*. https://dallery.gallery/the-dalle-2-prompt-book/

[38] Ben Pietrzak, Ben Swanson, Kory Mathewson, Monica Dinculescu, and Sherol Chen. 2021. Story Centaur: Large Language Model Few Shot Learning as a Creative Writing Tool.

[39] Catherine Pricilla, Dessi Puji Lestari, and Dody Dharma. 2018. Designing interaction for chatbot-based conversational commerce with user-centered design. In *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*. IEEE, 244–249.

[40] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International Conference on Machine Learning*. PMLR, 8821–8831.

[41] Kevin Roose. 2022. The Brilliance and Weirdness of ChatGPT. *The New York Times* (2022). https://www.nytimes.com/2022/12/05/technology/chatgpt-ai-twitter.html

[42] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2021. Multitask Prompted Training Enables Zero-Shot Task Generalization. https://doi.org/10.48550/ARXIV.2110.08207

[43] Jeff Sauro and James R Lewis. 2016. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann.

[44] Lisa Stifelman, Adam Elman, and Anne Sullivan. 2013. Designing Natural Speech Interactions for the Living Room. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (Paris, France) *(CHI EA '13)*. ACM, New York, NY, USA, 1215–1220. https://doi.org/10.1145/2468356.2468574

[45] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M Rush. 2022. Interactive and Visual Prompt Engineering for Ad-hoc Task Adaptation with Large Language Models. *arXiv preprint arXiv:2208.07852* (2022).

[46] M. Swan. 1927. *How You Can Write Plays: A Practical Guide-book*. S. French. https://books.google.com/books?id=UyYrAAAAIAAJ

[47] Mark Weiser. 1999. The Computer for the 21st Century. *SIGMOBILE Mob. Comput. Commun. Rev.* 3, 3 (jul 1999), 3–11. https://doi.org/10.1145/329124.329126

[48] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*.

[49] Tongshuang Wu, Michael Terry, and Carrie J Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *Proceedings of the 2022 CHI conference on human factors in computing systems*.

[50] Qian Yang, Jina Suh, Nan-Chen Chen, and Gonzalo Ramos. 2018. Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models. In *Proceedings of the 2018 Designing Interactive Systems Conference* (Hong Kong, China) *(DIS '18)*. Association for Computing Machinery, New York, NY, USA, 573–584. https://doi.org/10.1145/3196709.3196729

A BOTDESIGNER IMPLEMENTATION DETAILS & PILOT EVALUATION

For additional context, we offer the following details regarding BotDesigner's implementation and pilot evaluation.

A.1 Mapping Template to Prompt

To generate a given bot's chat messages, BotDesigner concatenates the preamble, the entirety of the conversation so far (using Bot: and User: prefixes for each turn as appropriate), the reminder (if any), and the final prompt Bot:. (Figure 4b in this appendix shows this relationship.) This text string is then sent as a single prompt to GPT-3. For consistency across tests, BotDesigner always uses GPT-3's `text-davinci-002` model with temperature[3] set to 0.

A.2 Evaluating Effects of Prompt Changes

To evaluate whether a particular **template** change affects any of the identified problematic chatbot responses, BotDesigner replays conversations containing errors and displays any modified responses. Two implementation approaches are possible for this task: (a) a system could either perform an "single replay" by assuming all conversational turns prior to the error will occur as in the original conversation, and test only whether the error utterance is changed; or, (b) it could perform a "total replay" in which every conversational turn is replayed and any changed utterances are flagged for user review. Both approaches have merit; the "total replay" approach is more consistent with the "regression testing" concept—certainly, a designer would not want to inadvertently introduce problematic utterances where none previously existed—but providing clear feedback requires identifying which conversational turns have changed in trivial ways, itself a nontrivial task.

For BotDesigner, we default to the "single replay" in an attempt to reduce noise, and accept the resulting short-term trade-off in accuracy that allows more rapid iteration—but leaving designers with the need to perform more extensive testing before deployment.

A.3 Pilot Evaluation of BotDesigner's Use

To validate that BotDesigner does enable systematic prompt evaluation, we recruited $N = 3$ academic researchers with a range of interest levels and experience with conversational agent design[4] and evaluated how effectively they could identify common or particularly severe bugs or errors in a baseline chatbot, and how effectively they could evaluate a new bot template for improvements over the baseline bot.

In advance of the interview, we collected conversations with the baseline bot from AMT workers conversing with a baseline chatbot. We then asked our pilot users to (1) browse the collected conversations to find errors and label them with categorization tags; (2) evaluate a "new" template with updated prompts, provided by us, and decide whether this new template resolved any of the errors participants had previously identified.

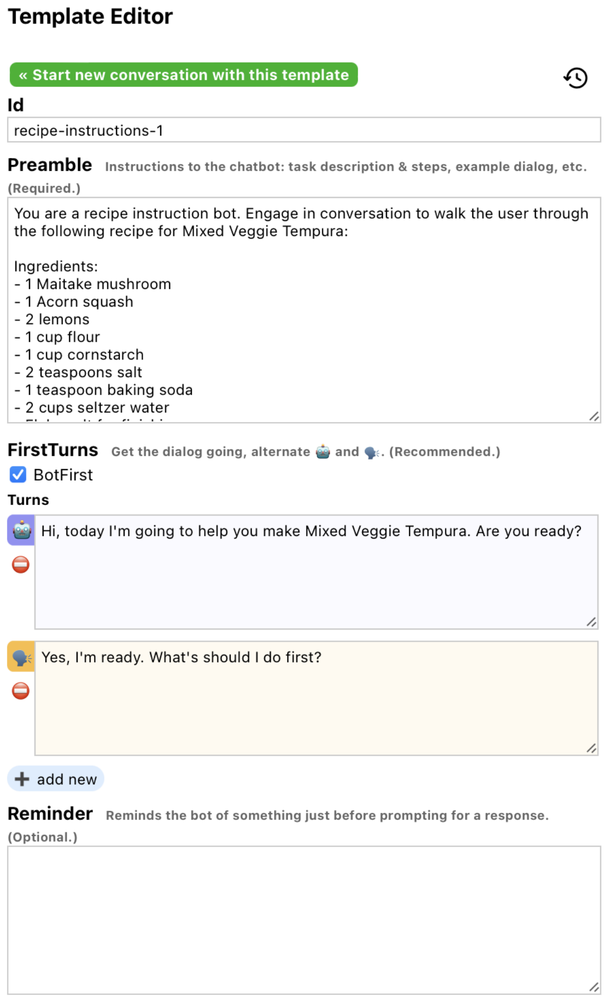
Our pilot users had no problems detecting most of a set of 5 error categories we had previously identified in the set of conversations between AMT workers and the baseline chatbot: (1) skipped steps, (2) ignored user expressions of negative emotion, (3) ignored user requests to wait until the user had completed some task, (4) factually incorrect responses to questions, and (5) otherwise unhelpful responses.

We learned a few things from this small pilot that ultimately informed our probe. First, we learned that BotDesigner functioned well enough for a small set of researchers to use it successfully. Second, critically, we observed that looking over a set of prerecorded conversations is quite cognitively demanding, involving lots of context switching, because these were not conversations pilot users themselves had with the bot. As a result of these observations, we (1) retained the "Error Browser", but (2) dropped a prototype "Conversation Browser" view (focused on tracking common conversational threads across conversations) from the version of BotDesigner we used in our probe. We also (3) designed a much more open-ended task for our probe requiring less knowledge of task minutiae to identify errors, and encouraging users to have their own conversations with the bot, while skipping the initial collection of conversations from AMT workers and thus avoiding the need for participants to understand the task well enough to identify subtle errors in instruction.

---

[3] When used to predict subsequent tokens given a specified prefix (which we call a "prompt" in this paper), language models typically assign a probability to *every possible subsequent token*, and then select among the most likely contenders. The temperature parameter affects how the next prediction is selected among the probability-ranked tokens. At temperature = 0, *the most likely next token is always selected*, preventing any random variation in response to a given prefix.

[4] At this stage, we are not exploring end users' intuitions about prompts, merely exploring whether BotDesigner's systematic prompt analysis functionality *can be* used as intended.


Template Editor
« Start new conversation with this template
Id
Preamble
Instructions to the chatbot: task description & steps, example dialog, etc. (Required.)
FirstTurns
Get the dialog going, alternate 🤖 and 🗣. (Recommended.)
☑ BotFirst
Turns
+ add new
Reminder
Reminds the bot of something just before prompting for a response. (Optional.)

Id: recipe-instructions-1

Preamble:

You are a recipe instruction bot. Engage in conversation to walk the user through the following recipe for Mixed Veggie Tempura:

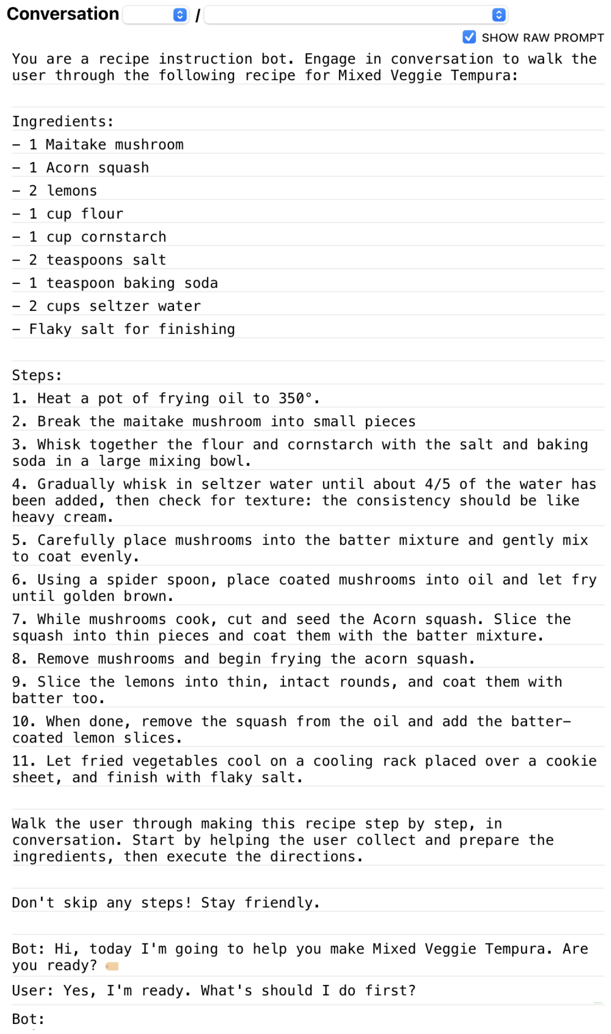
Ingredients:
- 1 Maitake mushroom
- 1 Acorn squash
- 2 lemons
- 1 cup flour
- 1 cup cornstarch
- 2 teaspoons salt
- 1 teaspoon baking soda
- 2 cups seltzer water

Turns:

🤖 Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

🗣 Yes, I'm ready. What's should I do first?

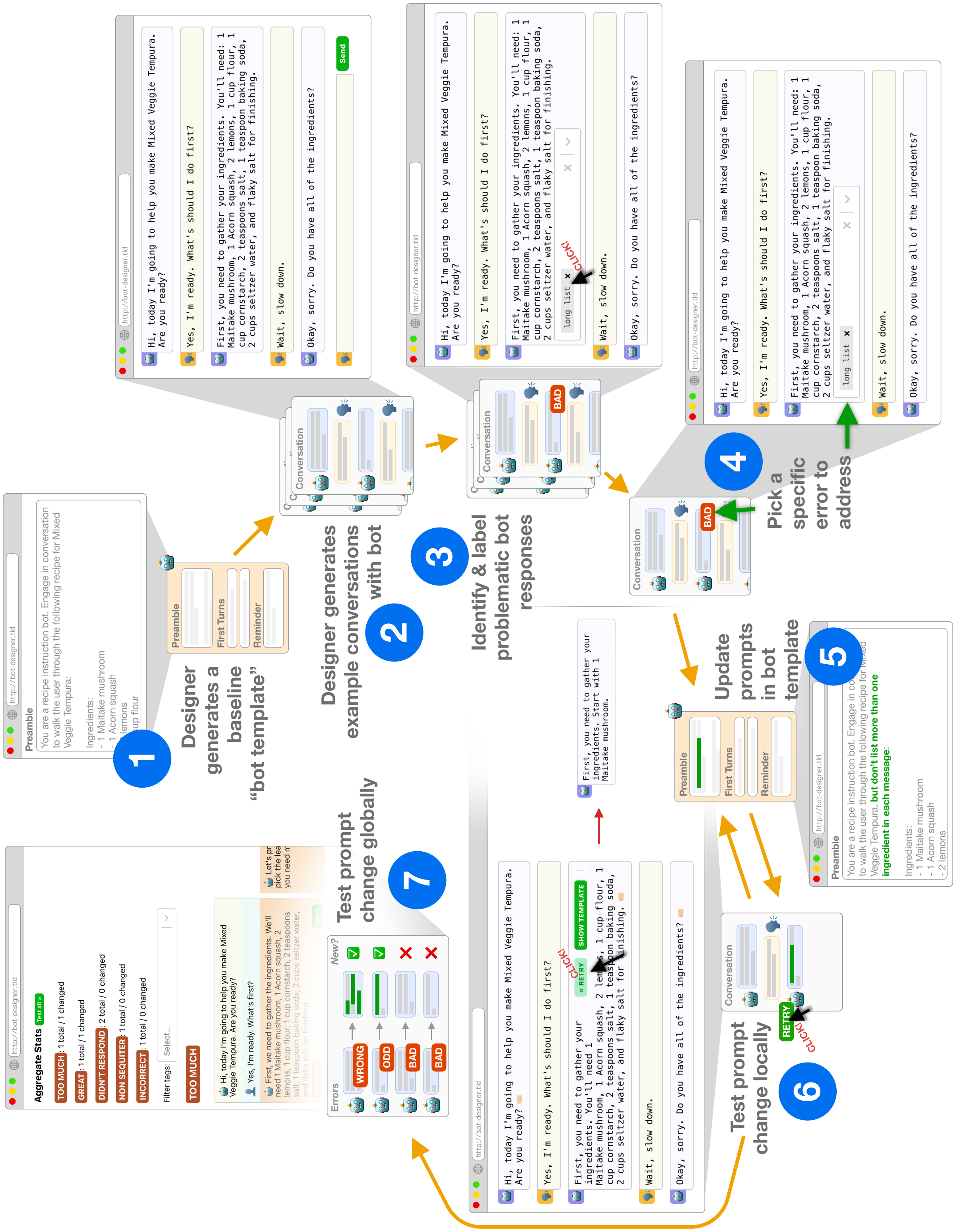
(a) **BotDesigner prompt template; note exclusive use of natural language throughout.**



```
You are a recipe instruction bot. Engage in conversation to walk the
user through the following recipe for Mixed Veggie Tempura:

Ingredients:
- 1 Maitake mushroom
- 1 Acorn squash
- 2 lemons
- 1 cup flour
- 1 cup cornstarch
- 2 teaspoons salt
- 1 teaspoon baking soda
- 2 cups seltzer water
- Flaky salt for finishing

Steps:
1. Heat a pot of frying oil to 350°.
2. Break the maitake mushroom into small pieces
3. Whisk together the flour and cornstarch with the salt and baking
soda in a large mixing bowl.
4. Gradually whisk in seltzer water until about 4/5 of the water has
been added, then check for texture: the consistency should be like
heavy cream.
5. Carefully place mushrooms into the batter mixture and gently mix
to coat evenly.
6. Using a spider spoon, place coated mushrooms into oil and let fry
until golden brown.
7. While mushrooms cook, cut and seed the Acorn squash. Slice the
squash into thin pieces and coat them with the batter mixture.
8. Remove mushrooms and begin frying the acorn squash.
9. Slice the lemons into thin, intact rounds, and coat them with
batter too.
10. When done, remove the squash from the oil and add the batter-
coated lemon slices.
11. Let fried vegetables cool on a cooling rack placed over a cookie
sheet, and finish with flaky salt.

Walk the user through making this recipe step by step, in
conversation. Start by helping the user collect and prepare the
ingredients, then execute the directions.

Don't skip any steps! Stay friendly.

Bot: Hi, today I'm going to help you make Mixed Veggie Tempura. Are
you ready?
User: Yes, I'm ready. What's should I do first?
Bot:
```

(b) **Text-only LLM *prompt* generated by a sample bot template. Seen here is an instance of the exact text transmitted to GPT-3 by BotDesigner, which collects GPT-3's text completion (or, "prediction") as the bot's response to the last User: request.**

Figure 4: Sample BotDesigner bot template and the corresponding generated prompts.


1
Designer generates a baseline "bot template"
2
Designer generates example conversations with bot
3
Identify & label problematic bot responses
4
Pick a specific error to address
5
Update prompts in bot template
6
Test prompt change locally
7
Test prompt change globally
http://bot-designer.tld
Preamble
First Turns
Reminder
Conversation
BAD
long list
CLICK!
Send
RETRY
« RETRY
SHOW TEMPLATE
Errors
New?
WRONG
ODD
Aggregate Stats
Test all »
Filter tags:
Select...

Preamble

You are a recipe instruction bot. Engage in conversation to walk the user through the following recipe for Mixed Veggie Tempura:

Ingredients:
- 1 Maitake mushroom
- 1 Acorn squash
- 2 lemons
- 1 cup flour

Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

Yes, I'm ready. What's should I do first?

First, you need to gather your ingredients. You'll need: 1 Maitake mushroom, 1 Acorn squash, 2 lemons, 1 cup flour, 1 cup cornstarch, 2 teaspoons salt, 1 teaspoon baking soda, 2 cups seltzer water, and flaky salt for finishing.

Wait, slow down.

Okay, sorry. Do you have all of the ingredients?

You are a recipe instruction bot. Engage in conversation to walk the user through the following recipe for Mixed Veggie Tempura, **but don't list more than one ingredient in each message:**

Ingredients:
- 1 Maitake mushroom
- 1 Acorn squash
- 2 lemons

First, you need to gather your ingredients. Start with 1 Maitake mushroom.

TOO MUCH: 1 total / 1 changed
GREAT: 1 total / 1 changed
DIDN'T RESPOND: 2 total / 0 changed
NON SEQUITER: 1 total / 0 changed
INCORRECT: 1 total / 0 changed

TOO MUCH

Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

Yes, I'm ready. What's first?

First, we need to gather the ingredients. We'll need 1 Maitake mushroom, 1 Acorn squash, 2 lemons, 1 cup flour, 1 cup cornstarch, 2 teaspoons salt, 1 teaspoon baking soda, 2 cups seltzer water, and flaky salt for finishing

Figure 5: BotDesigner workflow. Numbered steps here correspond to the numbered steps described in §3.2

B TRANSCRIPT OF "BACK TO BACK CHEF: CARLA LALLI & ELIZABETH OLSON"

B.1 Behavioral categories

Observed behavioral categories for Carla:

- Concept simplification, analogies,
- Detailed instructions that recipes typically don't cover
- Step completion confirmation
- Social conversation
- Repair
- Emotional grounding

These categories were highlighted in the following transcript.

B.2 Transcript

Today we have 20 minutes to make mixed veggie tempura and we're gonna see if Lizzie can follow along with me through verbal instructions only.
Expert: So the first thing we're gonna do is take this very brain-looking giant mushroom – so this is a hen-of-the-woods mushroom, also called a maitake. I got my squash and my lemons out of the way.
So the first thing I want you to do is just turn it over so the stem side is up got it and then just use your hands to kind of break big clusters off of the bottom and like kind of large pieces.
And then we'll go back in and we can make them smaller and I like doing this with my hands because then you get -
Amateur: oh you're not using your paring knife
Expert: No, I'm only using the paring knife if it was dry or spongy at the bottom. We're probably used a quarter of these (mushrooms.) So then just take the tray with the mushrooms on it and move that over to your right-hand side and we're gonna make the batter.
So next thing – you got your board cleared?
Amateur: Yes, there's mushroom juice on it but it's fine.
Expert: I think that's the line from a Woody Allen movie. So now you've got two bowls. One is flour and one is cornstarch. So put those into the larger bowl together. We're gonna make the tempura batter now.
Once you have your flour and your cornstarch. You have a little dish that has salt and baking soda and those get combined too.
Amateur: Wait… Wha..?
Expert: They should be in the same little bowl.
Amateur: There's another little bowl that has salt and baking powder - it was to my left. Is it to your left?
Expert: It's in my bowl now but yeah there was a little bowl with white stuff that isn't salt then that's your…
Amateur: Okay.
Expert: And so let's just whisk the dry ingredients first. So once the dry ingredients are whisked together, take your measuring cup that's over to the right and those two little bottles of seltzer and we're gonna measure 2 cups, which I want to add gradually. So take your whisk in one hand, your measuring cup in the other and then gradually whisk in. Do about four-fifths of this and then we'll check the texture …
Amateur: It's like… you know what "Oobleck" is?
Expert: What did you say? "Blueblack?"?
Amateur: "Oobleck". It's like a thing you play with as a kid with, like baking soda and water and that's what it's reminding me of.
Expert: Cool, and it should be about the consistency of heavy cream.
Amateur: Yeah okay. I'm gonna try and get a little more bumps out but I think I'm almost there.
Expert: All right so I'm gonna put my whisk into the garbage bowl just to get it out of the way and do you feel like you're in a lump-free place? Because we're gonna start frying.
Amateur: Em oh no... I'm like a perfectionist, so I'm gonna... I think I'm just gonna say yes…
Expert: Okay, great. Take one big handful of mushrooms.
Amateur: Uh huh.
Expert: And, put them right into the batter. And, there should also be a fork, and a spoon.
Amateur: Yeah.
Expert: And, I'm gonna use my hands and the fork to really gently, 'cause I don't wanna break these clusters up, but I wanna coat all of the mushrooms with batter.
Amateur: Yup.
Expert: So, we should be at around 350. So, maybe while we're waiting we can start prepping our acorn squash.
Amateur: Great.
Expert: So, now you need your cleaver.
Amateur: Yup, oh God.
Expert: You got your acorn squash.
Amateur: Yes.
Expert: So put it on one side so that the stem end is like closest to your dominant hand.
Amateur: Yes, got it.
Expert: And then, just cut straight down, kinda close to the stem. And, just shear off that front part.
Amateur: How do you use a cleaver? Do you just like... hit it?
Expert: You can hit it. You can use it like a slicing knife and just go straight down.
Amateur: I make really fun faces when I'm frustrated, or like straining, and it's not pleasant to have a camera in front of me.
Expert: All you need to do right now is just create like a flat edge so that we can stand–
Amateur: Did it. Okay.
Expert: Great. And then, we're gonna cut it into two lobes. Straight down.
Amateur: Okay.
Expert: And, if you're off to one side it's fine, 'cause we only need a quarter of the squash.
Amateur: Just a quarter.
Expert: Just a quarter. So, get–
Amateur: So, cut the halves into quarters as well.
Expert: Yeah, so once you have it in half.
Amateur: Got it.
Expert: Take your spoon, and scoop the seeds into that little trash bowl that we made.
Amateur: Yes.
Expert: And then, we're gonna leave the skin on, and cut this into thin slices.
Amateur: Okay.
Expert: Going crosswise, so they're kind of like, they look like little clouds to me.

Amateur: Okay. Do you find it easier to use a cleaver than like a chef's knife?
Expert: Not necessarily.
Amateur: Yeah, I'm not finding it the easiest.
Expert: You could switch to your paring knife if you want.
Amateur: It's okay. I'm just kind of scared of big silvery things.
Expert: Yeah, it's a scary tool for sure. All right, so by now, do you have a quarter sliced? A quarter acorn sliced?
Amateur: Yup, just about.
Expert: All right. So, go back to the mushrooms.
Amateur: Yup.
Expert: And, then using your hand and the fork again.
Amateur: Yes.
Expert: Lift them out and let that excess batter drip off. And then, you wanna just go right into the oil.
Amateur: Okay, I'm using my hands 'cause I–
Expert: Same. But, we're not afraid of fry oil.
Amateur: Well, it's not splattering the way my, the way I've been frying, 'cause I don't have a thermometer...so clearly I've been–
Expert: You might be going really high.
Amateur: I think it was way too hot. Wonder if I should've added more of my club soda.
Expert: If it feels thick, yeah, add more.
Amateur: 'Cause I do, I feel like mine's–
Expert: A little too thick?
Amateur: It's a thick tempura.