

# Conversation Regression Testing: A Design Technique for Prototyping Generalizable Prompt Strategies for Pre-trained Language Models

*J.D. Zamfirescu-Pereira  
Björn Hartmann  
Qian Yang*

Electrical Engineering and Computer Sciences  
University of California, Berkeley

Technical Report No. UCB/EECS-2023-16

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-16.html>

February 7, 2023



Copyright © 2023, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

# Conversation Regression Testing: A Design Technique for Prototyping Generalizable Prompt Strategies for Pre-trained Language Models

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

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

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

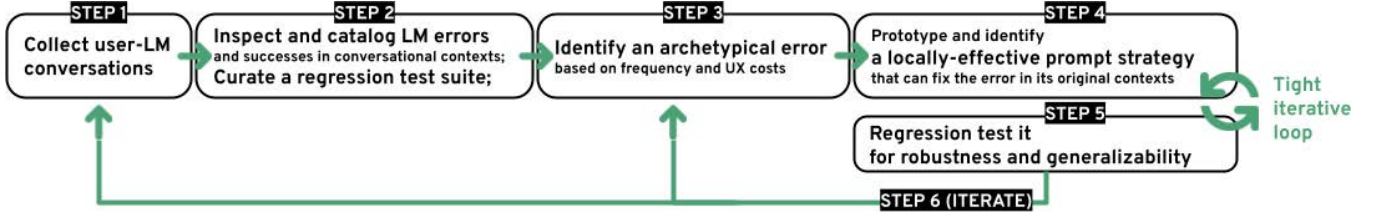


Figure 1: We propose *Conversation Regression Testing*, a workflow for chatbot designers to systematically experiment and evaluate various prompt strategies’ impact on pre-trained-language-model-powered conversational interactions. We also present *BotDesigner*, a prompt prototyping tool that operationalizes this workflow.

## ABSTRACT

Pre-trained language models (LLMs) such as GPT-3 can carry fluent, multi-turn conversations out-of-the-box, making them attractive materials for chatbot design. Further, designers can improve LLM chatbot utterances by prepending textual *prompts* – instructions and examples of desired interactions – to its inputs. However, prompt-based improvements can be brittle; designers face challenges systematically understanding how a prompt strategy might impact the unfolding of subsequent conversations across users. To address this challenge, we introduce the concept of Conversation Regression Testing. Based on sample conversations with a baseline chatbot, Conversation Regression Testing tracks how conversational errors persist or are resolved by applying different prompt strategies. We embody this technique in an interactive design tool, *BotDesigner*, that lets designers identify archetypal errors across multiple conversations; shows common threads of conversation using a graph visualization; and highlights the effects of prompt changes across bot design iterations. A pilot evaluation demonstrates the usefulness of both the concept of regression testing and the functionalities of *BotDesigner* for chatbot designers.

## 1 INTRODUCTION

The combination of pre-trained large language models (LLM) and prompts offers exciting new opportunities for chatbot design. Recent pre-trained LLMs (GPT-3 [9], GPT-J [39], Jurassic-1 [24], and T0 [35]) can engage in fluent, multi-turn conversations out-of-the-box. Removing the costs and data requirements for training supervised models, these models substantially lower the barrier of entry for creating a passable conversational user experience (UX) [7]. Further, chatbot designers can improve LLM outputs by prepending *prompts*—textual instructions and examples of designers’ desired interactions—to LLM inputs (Table 1.) Prompts directly

bias the model towards generating the desired outputs, raising the ceiling of what conversational UX is achievable with little or no labeled data. In the past two years, the promises of this new *pretrain-and-prompt* paradigm have been propelling a rapidly growing set of prompt design tools [3, 17, 27], research across HCI/UIST (e.g., [18, 23, 26, 40, 41]) and NLP (e.g., [22, 25, 31, 35, 36]), and even end-user-facing applications [18]. Some scholars even argued that this paradigm represents a “*sea change*” in NLP and will “*replace*” the fully-supervised and pre-train-and-fine-tune paradigms [25].

Prompt Strategy	Resulting Conversation
No prompt (baseline)	<p>🗣️ Ok hang on while I get a chair</p> <p>🛋️ Scoot to the front of your chair[...]</p>
Explicit instruction to wait	<p>🗣️ If the user asks you to wait, explain that this is not a problem [...]</p> <p>🗣️ Ok hang on while I get a chair</p> <p>🛋️ Once you have your chair, scoot to the front of it[...]</p>

Table 1: An example of how designers can directly improve chatbot interactions by modifying **prompt strategies** (🗣️). Note the change in the bot’s (🛋️) response to the user’s (🗣️) statement.

The pre-train-and-prompt paradigm also brings a series of new challenges to chatbot design, namely, the challenges of *designing robust prompt strategies*. To leverage prompts for production-level chatbots, designers need to first systematically assess (1) in what conversational contexts the pre-trained LLM is likely to fail and (2)

how frequent or damaging each failure or failure mode is, in order to identify the right problems to solve with prompts. Next, designers need to (3) identify a prompt strategy that can fix the target failure in its original conversational context, and finally, to assess its *generalizability* and *robustness* systematically, that is, assessing (4) whether it can fix similar failures in other conversational contexts, and whether it might cause new errors across the numerous ways the conversations can unfold subsequently for different users. These are challenging tasks [7, 25, 35].

A few HCI researchers have started to create workflows and tools that aid prompt strategy design, for example, for human-LLM collaborative writing [40]. However, such workflows and tools for chatbots are extremely rare. Instead, chatbot designers often experimented prompts *ad-hoc* using tools such as GPT-3 Playground [30]; Some even treated prompt strategy design as “*rolling the dice*” [41]. It remains unclear how designers can holistically analyze the highly-contextual errors LLMs make across conversations (challenges 1, 2), or how they can resolve the errors without unknowingly causing new errors in preceding or subsequent conversations (challenges 3, 4).

As a step toward more systematic and rigorous prompt strategy prototyping, we introduce the concept of *Conversation Regression Testing*. Taking inspiration from software regression testing, *Conversation Regression Testing* uses the conversational contexts where a baseline LLM has failed (or notably succeeded) as reusable test cases and helps designers track the effects of prompt strategy updates on these test cases. This approach allows designers to freely experiment with many prompt strategies to address a particular error in context, while ensuring the system’s overall stability and a trajectory of continuous improvements.

Operationalizing this concept, we then present BOTDESIGNER, a prompt strategy prototyping tool that integrates the *Conversation Regression Testing* workflow into an interactive machine learning analysis tool (one that tracks model performance across iterations and provides insights into what changes yield what performance improvements.) Such tools have shown remarkable traction with designers and developers in non-conversational domains (e.g., Weights and Biases [5]). BOTDESIGNER consists of four components:

- **Conversation Collector**, an interface for collecting sample conversations between a baseline LLM-based chatbot and real-world users (or crowd workers);
- **Annotator**, an interface for inspecting and cataloging the problematic (or successful) utterances made by the baseline bot, across many conversations with multiple users. These errors are opportunities for prompts to help, as well as test cases for *Conversation Regression Testing*;
- **Visualizer**, a graphic visualization that aids designers to identify archetypal errors by showing the baseline bot’s failures and successes against the backdrop of common end-user-LLM conversation patterns. These archetypal errors help designers to prioritize their prompt design efforts;
- **Regression Tester**, features that embody *Conversation Regression Testing*. When designers experiment with a new prompt strategy, these features enable them to track whether the target error persists or gets resolved, or if new errors have appeared, as a result of the new strategy.

This paper presents the concept of *Conversation Regression Testing*, the implementation of BOTDESIGNER, and a small user evaluation study that preliminarily demonstrates the usefulness of both for chatbot designers when designing instructional chatbots.

This paper makes two contributions, one conceptual and one technical. The primary contribution is the concept of *Conversation Regression Testing* for prompt strategy design. While most prior work focused on exploration-and-ad-hoc-testing stage of prompt design, *Conversation Regression Testing* offers an initial workflow to for assessing prompt strategies’ robustness and generalizability. Secondly, the technical contribution of this paper lies in the techniques for implementing BOTDESIGNER. It presents a novel conversation visualization technique that visualizes common conversation patterns across many discrete conversations between an LLM and various users. It can be useful for developing many other human-LLM interaction analysis or design tools. BOTDESIGNER also implements an interface for *Conversation Regression Testing*, a technique that can be valuable for prototyping prompts for many other LLM applications beyond conversational interactions.

## 2 RELATED WORK

We briefly review three threads of related work: 1) workflows and tools for interactively improving NLP model performance and 2) for improving conversational UX, and finally 3) prior conversation visualization techniques and analytical tools.

### 2.1 NLP Modeling Workflows and Tools

NLP modeling workflows and tools roughly fall under three categories [25]. **Fully supervised learning**, where a task-specific model is trained on a dataset of input-output examples for the task, has long played a central role in machine learning (ML) and natural language processing (NLP). Because fully labeled datasets are often insufficient for learning high-quality models, interactive NLP tools for improving model performance focused heavily on assisting feature engineering; providing models with the appropriate inductive bias to learn from this limited data. Towards this goal, supervised NLP tools most often embodied one of the two workflows:

- Tools such as LightSIDE [28] assist NLP modelers to define and extract salient features from raw data. These tools adopted a five-step workflow that many seminal interactive ML tools (e.g., Crayons [16], ModelTracker [2], Gestalt [32], and Weights and Biases [6]) have pioneered: Modelers (i) inspect raw data; (ii) label data or extract features from the data, sometimes with the assist of ML; (iii) train an initial model, (iv) classify, view, and correct the model’s outputs, and (v) iterate on this process while the tools track the model’s performance improvements and provide insight into what changes yield the improvements.
- The second workflow emerged in response to the criticism that the above workflow left out considerations of ML amateurs[1]. Researchers created “*human-centered ML tools*” that added end-users to every step of the first workflow (e.g., allowing them to provide traces of their natural interaction with the model for model training [43] and transfer learning [29], nominate features [11], demonstrate desired model behaviors [42], etc.) These tools demonstrated that integrating an understanding and natural

interaction data of end-users into ML workflow can improve both UX and model performance [1].

In 2017-2019, the standard way of NLP modeling shifted to “*pre-train and fine-tune*”, with fully supervised learning playing an ever-shrinking role [25]. This paradigm embodies a two-step-only, no-longer-task-specific ML workflow.

- Step i Modelers pre-train a model with a fixed architecture on large, unlabeled textual data. In this process, the pre-trained LLM learns general-purpose language features that can be used for a wide range of tasks (e.g., predicting the next line of code or prose, document summarization, biomedical question answering, translation, and more.) GPT [9, 39] and BERT [13] exemplify families of pre-trained LLMs.
- Step ii Modelers adapt the pre-trained LLM to the particular interaction task at hand through fine-tuning.

In this paradigm, the main focus of model tuning turned from feature to objective engineering, designing the training objectives for both pre-training and fine-tuning. As a result, most aforementioned interactive ML tools no longer apply. While a few commercial general-purpose ML tools (e.g., Azure [45]) can support this new workflow, we did not find interactive NLP tools tailored for this workflow in our literature search.

The past two years have been witnessing another paradigm shift in NLP: the rise of the “*pre-train, prompt, and predict*” paradigm [25]. This paradigm follows roughly the 2-step workflow above. However, instead of adapting pre-trained LLMs to particular tasks via objective engineering, modelers reformulate the tasks to look more like those solved during the original LLM training with the help of a textual *prompt*. For example, GPT-3 can automatically translate users’ natural language requests to html code using the prompt template/strategy “web code description: <natural language request> html:<html> css: <css> javascript:<js>” [18]. Modelers curate a large set of such prompts using a template and retrain the LLM with them [3, 14].

In this paradigm, the main focus of model tuning turned to prompt engineering, designing the appropriate prompts and prompt strategies that yield the desired model behaviors. Further, because many prompts are human-readable, prompts also present renewed opportunities to engage end-users in the modeling process. Tools have emerged to enable crowd workers or end-users to contribute queries and prompt strategies [3, 14].

Noteworthy, even for experts, identifying robust and generalizable prompt strategies requires extensive trial and error, where modelers iteratively experiment and assess the effects of various prompt strategies on concrete input-output pairs, before assessing them more systematically on large conversation datasets. A well-established prompt design workflow does not yet exist. How a prompt or a prompt strategy may directly impact model outputs, or how it modifies pre-trained LLM’s billions of parameters during re-training, are both active areas of NLP research [25, 35].

## 2.2 Prototyping Chatbot UX

A well-established workflow exists for designing and prototyping multi-turn conversational interactions and experiences (“*chatbot UX*”, for short) [10, 12, 19, 20, 34, 37]). Following this workflow,

chatbot 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., “*greeting message, questions to collect user intention, ...*”); and finally (iii) fill the template with supervised NLP models (e.g., user intention classifier, response generator, etc.) Many tools that support this process exist supporting this process, for example, Google Dialogflow and Facebook Messenger tools for step (ii) and (iii).

While highly valuable, these conversation-template-oriented tools are ill-fitted for pre-trained LLMs. However, chatbot design tools for the pre-train-and-prompt paradigm are extremely rare. The closest related work is *AI Chains*[41], a tool for exploring human-LLM collaborative writing interactions. It 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 enabled designers to explore prompt and chaining strategies more efficiently and strategically [40]. However, it is unclear whether the resulting strategies were effective or robust beyond the few interactions contexts that the designers experimented with.

## 2.3 Conversation Visualization and Analysis

Prior work on visualizing conversations has either focused on visualizing the structure of a dyadic (email) [38] or multi-party conversation [15, 44] over time (newsgroups, etc); or, they’ve sought to create a more abstract, higher-level picture of the topics covered in a conversation [4]. Our needs here are different, since we’re considering the unique settings of multiple independent conversations about the same topic—visualizing which pieces are shared and which are unique to each conversation. Some related work does also touch on the adjacent task of visualizing the structure of multiple *tutorials* (rather than conversations) covering a single topic, exploring which pieces are shared and which are unique to each tutorial [21, 33].

## 3 CONVERSATION REGRESSION TESTING

We wanted to help chatbot designers to freely prototype and systematically evaluate prompt strategies, thereby empowering them to leverage pre-trained LLMs and prompts in their design. To this end, we introduce the concept of *Conversation Regression Testing*.

### 3.1 Definition and Benefits

*Conversation Regression Testing* is an iterative workflow for prototyping and evaluating prompt strategies. Following this workflow, chatbot designers start by identifying a baseline prompt strategy (or an off-the-shelf pre-trained LLM, i.e. with no prompt strategy). They then carry out the following complementary activities (Figure 2):

- (1) *Collect human-LLM conversations*: Collect a diverse set of conversations between the baseline LLM and end-users through crowdsourcing or in-person user studies;
- (2) *Inspect and catalog LLM errors and successes in context*: Inspect the errors and successes both in the contexts where they occurred and in aggregate, across the myriad ways the baseline user-LLM conversations have unfolded; add noteworthy user-LLM conversation turns to a suite of regression test cases;

- (3) *Identify an archetypical error* based on how frequent or damaging each error or error pattern is; develop intuitions of possible new prompt strategies for addressing the error;
- (4) *Identify a locally-effective prompt strategy*: Experiment with new prompt strategies to fix a particular archetypical error in the conversational context where it originally occurred; Identify one locally effective prompt strategy;
- (5) *Regression test for robustness and generalizability*: Apply the locally effective prompt strategy to the entire regression test suite, inspecting its robustness (whether it has fixed similar failures in other conversational contexts) and generalizability (whether it has caused new errors across the numerous ways the conversations can unfold subsequently for different users). If not, iterate on step 4-5 or even collect more conversations (step 1-5) before proceeding. If positive for both, continue;
- (6) *Iterate while tracking*: Consider the robust and generalizable prompt strategy as a new baseline, iterate on the whole process (step 1-6) while tracking which errors have been resolved versus persisted.

Central to this workflow are the concepts of *conversation regression testing* and *prompt prototyping in human-LLM conversational contexts*. They highlight the benefits of *Conversation Regression Testing* over existing common practices.

**Benefits over current chatbot UX prototyping workflow.** Similar to software regression test suites [8], conversation regression test suites enable chatbot designers to track the effects of prompt strategy updates on many discrete conversations with different users. This approach is particularly valuable for prompt strategy design, because UX improvements and breakdowns caused by prompts are often brittle. In comparison to the current UX practice where designers tend to test their prompt strategies on the utterances they themselves authored in an ad-hoc manner [26, 40], conversation regression test cases enable designers to freely experiment with many prompt strategies, without unknowingly causing new errors in preceding or subsequent conversations.

Importantly, *Conversation Regression Testing* is not merely Regression Testing applied to prompt design. *Conversation Regression Testing* is a rapid and iterative prototyping process. Each iteration resolves an error or an error mode without regression. This is different from software regression tests, whose use is typically limited to when new program updates reintroduces *old* errors (hence the name *regression*.)

**Benefits over current NLP practice.** *Conversation Regression Testing* highlights the importance of the use of user-LLM conversation texts throughout the prompt strategy prototyping process. Designers inspect errors and test new strategies, both in the original user-LLM conversational contexts where errors (or successes) occurred. This is a departure from current common NLP practice, where modelers typically evaluated prompt strategies on pre-curated human-human conversation datasets. Taking a lesson from human-centered ML work, end-user interactions with a model – particularly their reactions to its errors – should not be an afterthought.

### 3.2 Conversation Regression Testing In Practice: An Example Design Process

Let us ground the concepts and workflow of *Conversation Regression Testing* and their benefits in a concrete example. Consider ourselves chatbot designers who are creating an *ExerciseBot*, a voice-based conversational agent that walks users through a set of physical exercises that they can perform at their desk. Following the *Conversation Regression Testing* workflow, we can rapidly prototype various prompt strategies in-context and systematically evaluate their robustness and generalizability:

We start by identifying a baseline prompt strategy. Here we use GPT-3’s text-davinci-001 model (setting TEMPERATURE = 0) out-of-the-box. We use the simple combination of a set of publicly available exercise instructions and a request to “*instruct the user in completing each exercise step-by-step*” as our baseline (Table 2);

- (1) *Collect human-LLM conversations*: We collect 30 conversations between the baseline bot and 10 Mechanical Turk workers, which yields many creative yet realistic utterances that we could hardly anticipate (“*At my age I’m going to have to break them up.*” “*Is it more effective to do all [exercises] at once?*”).
- (2) *Inspect and catalog errors and successes in context*: We found that the baseline prompt strategy is sufficient to create a passable chatbot that, most often, naturally walked users through the exercise step-by-step (e.g., User: “*At my age I’m going to have to break them up.*” Bot: “*That’s ok, just try to complete all 5 reps.*”) We also identified a number of error patterns. For example, the “skip a step” error is that the bot skips a step when walking users through the exercises. The “*unsympathetic*” error is where the bot routinely ignores user requests for help (“*Can we try an easier exercise?*”) or expressions of distress (“*Ow, that hurt!*”). We collected these conversations as substrates for our *Conversation Regression Testing* test suite.

---

#### Baseline prompt

---

Consider the following set of exercises:

1. Tricep Dips. Scoot to the front of your chair, with both hands facing forward, [...]
2. Seated Leg Lifts. Grab the sides of your chair [...]

Instruct the user in completing each exercise step-by-step.

---

#### New prompt (fixing the “skip a step error”)

---

Consider the following set of exercises:

1. Tricep Dips. Scoot to the front of your chair, with both hands facing forward, [...]
2. Seated Leg Lifts. Grab the sides of your chair [...]

Instruct the user in completing each exercise step-by-step.

[Don’t skip any steps.](#)

---

**Table 2: The baseline and improved prompts in the *ExerciseBot* design example.**



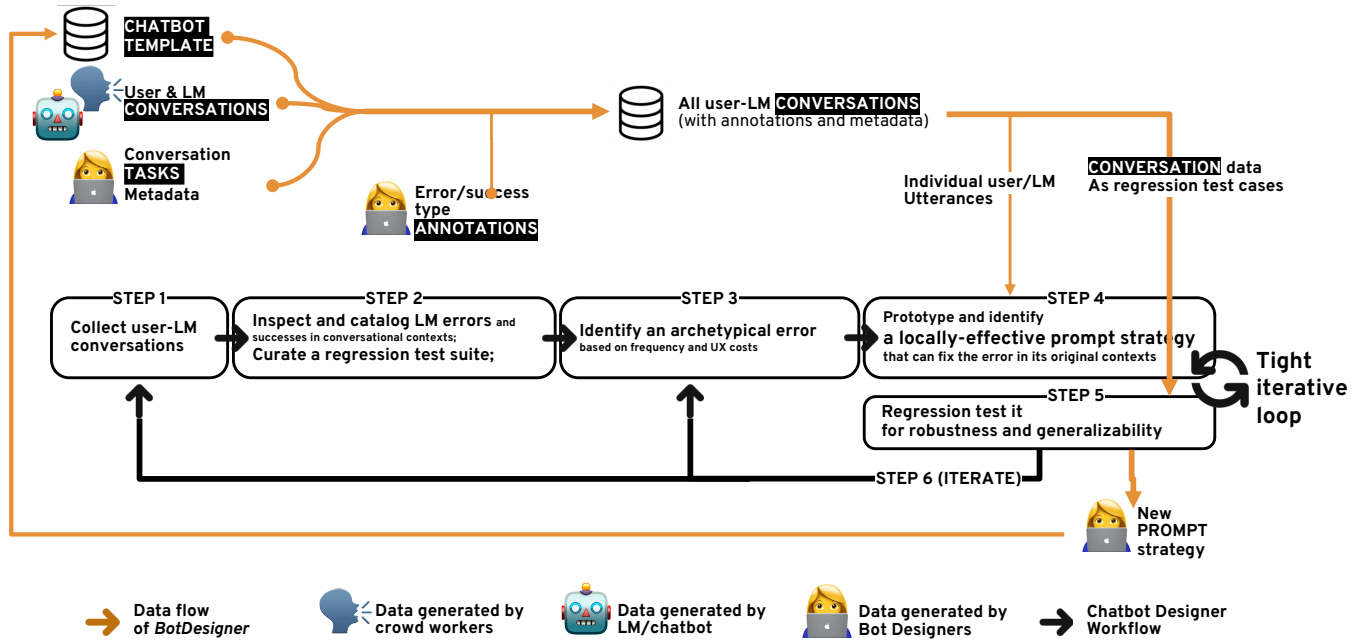


Figure 2: Conversation Regression Testing workflow and BOTDESIGNER data flow.

- (3) *Identify an archetypical error.* We chose to focus on the “skip a step” error, since it causes confusion if not physical danger during the exercises. It also has frequently caused breakdowns in subsequent conversations when users requested clarifications.
  - (4) *Identify a locally-effective prompt strategy:* After extensive experimentation, we resolved the “skip a step” error by simply appending the explicit instruction “Don’t skip any steps.” to the end of the baseline prompt, before the user-bot conversations begin. Another locally-effective strategy is to number the sub-steps within each step of the exercises in the initial prompt (Table 2.)
  - (5) *Regression test for robustness and generalizability:* Applying the two new strategies to the previously curated test cases, we noticed that the explicit instruction strategy consistent resolves the “skip a step” error, while the numbering-the-steps strategy only worked for some exercises. However, in some contexts, the explicit instruction strategy caused a side effect: It makes the bot’s stubbornly stick to the step-by-step exercise instructions, even when users said this step is too hard. It could worsen the “unsympathetic” error.
- With this trade-off in mind, we iterate on step 4-5, exploring additional prompt strategies that may work even better. We could also choose to collect additional conversations (for example, on a different set of exercises), thereby identifying new patterns of errors and success (steps 1 and 2). This approach allows us to fully understand the extent to which the new prompt strategy is robust and generalizable before adopting it.
- (6) *Iterate on this process to tackle additional errors while tracking ExerciseBot’s behavior changes using the Conversation Regression Testing test suite.*

## 4 BOTDESIGNER: A TOOL THAT OPERATIONALIZES CONVERSATION REGRESSION TESTING

We present BOTDESIGNER, a chatbot prompt strategy prototyping tool that operationalizes the *Conversation Regression Testing* workflow described in §3.2.

### 4.1 System Overview

BOTDESIGNER enables *Conversation Regression Testing* with the following functionality:

- (1) A **conversation collection** interface that enables the crowd-sourcing of a set of baseline conversations with a baseline GPT-3 based chatbot; this interface enables step (1) described in §3.2.
- (2) A **conversation visualization and annotation** interface that shows conversation flow across multiple users’ conversations (for a single *task*, defined in §4.2) using a graph interface, highlighting which utterances are common across conversations, and aiding in the categorization and tagging (annotation) of individual problematic or particular successful bot-provided utterances for targeted improvement or maintenance. This interface enables steps (2)-(3) from §3.2.
- (3) A **utterance testing** interface that situates individual problematic utterances in context and highlights changes to those utterances caused by updates to the bot. This interface enables steps (4)-(5) from §3.2.

In conjunction with a built-in code editor, these interfaces support iteration over chatbot prompt designs.

## 4.2 Inputs

BOTDESIGNER relies on three types of input data: **conversations**, **tasks**, and **templates**, representing, respectively, individual multi-turn user interactions with a specific bot (*conversation*), a set of structured instructions that make up the user’s task (*task*), and a set of prompts comprising a specific point design for a chatbot (*chatbot template*). Although we believe *Conversation Regression Testing* can be usefully applied to any type of chatbot, we chose to focus on *task-oriented instructional* interactions because of the opportunities for aggregation offered by similarities across multiple conversations by multiple users focused on the same task.

**Conversations** are specific multi-turn interactions collected by BOTDESIGNER, consisting of a dialog data structure that includes each conversation partner’s utterances as well as any error annotations provided *post facto* by the designer or human conversation partner. Each conversation is attached to the specific template and recipe used to generate the bot’s utterances.

**Tasks** are specific structured task descriptions comprised of a name, description, and set of steps the user is expected to complete. Some tasks may also include metadata such as a list of the items required to complete the task.

**Chatbot templates** describe the set of prompts that are sent as a prefix to the backing LLM (GPT-3 in the case described here). Each template contains instructions for (1) how to convert a structured **task** of the appropriate type into plain text, suitable for inclusion into the LLM text prompt, and (2) code describing how to lay out, in the prompted text, the turn-by-turn dialog-in-progress that is stored in the **conversation**. Templates also describe how the LLM output should be parsed and the bot’s response utterance extracted. See Fig. 3 for an example.

## 4.3 Using BOTDESIGNER

BOTDESIGNER supports each of the four steps of *Conversation Regression Testing*:

In **conversation collection** mode, BOTDESIGNER requests utterances from the user, generates a full prompt, sends it to GPT-3’s API requesting a prediction for the following tokens, receives GPT-3’s response, extracts the predicted bot utterance, and displays it to the user. See Figure 4 for an example of this interface.

Supporting **utterance annotation and conversation discovery**, BOTDESIGNER allows designers to identify problematic and successful utterances and then attach single-word tags to those utterances for easier aggregation of errors by type (see Fig. 5). A separate view, the *conversation visualizer*, shows all collected conversations (optionally filtered by data source and the presence of specific errors), making use of a graph data structure and visualization. This graph structure shows which utterances and tagged error types are common to many conversations—typically these are specific steps within the instructions, but they can also be common questions asked by users.

Figure 6 shows an example of the conversation visualizer aggregating the conversation flows of 12 conversations collected from AMT workers using a baseline version of ExerciseBot. The red border and edge coloring highlights the flow of a single conversation embedded within the full set of conversations. Nodes that have been tagged or identified as problematic or especially strong

```

1  template = {
2
3    id: "exercise-1",
4
5    ctype: "exercise",
6
7    preamble: exercise => `Consider this set of exercises:
8
9    ==Exercises==
10   ${exercise.name}
11
12   ${exercise.exercises.map((e,i) => `=${i+1}: ${e.name}=
13   ${e.description}
14   `).join('\n')}
15   Next, walk the user through these exercises one at a time, in conversation. For
16   each exercise, break the steps down into individual sentences and walk the user
17   through each step.
18
19   firstTurns: exercise => [
20     {who: 'bot', said: "Hi, today I'm going to help you with "+exercise.name},
21     {who: 'user', said: "Great, let's get started. What's the first exercise?"}
22   ],
23
24   formatTurn: (exercise, turn) => `${
25     { bot: 'Bot', user: 'User' }[turn.who]}: ${turn.said}`,
26
27   dialogPrompt: function dialogPrompt() {
28     return this.turnSeparator + "Bot: ";
29   },
30
31   extractResponse: (response, prompt) => response.data.choices[0].text.substr
32   (prompt.length).trim(),
33
34   turnSeparator: "\n",
35
36   generationStop: "\nUser:",
37 }

```

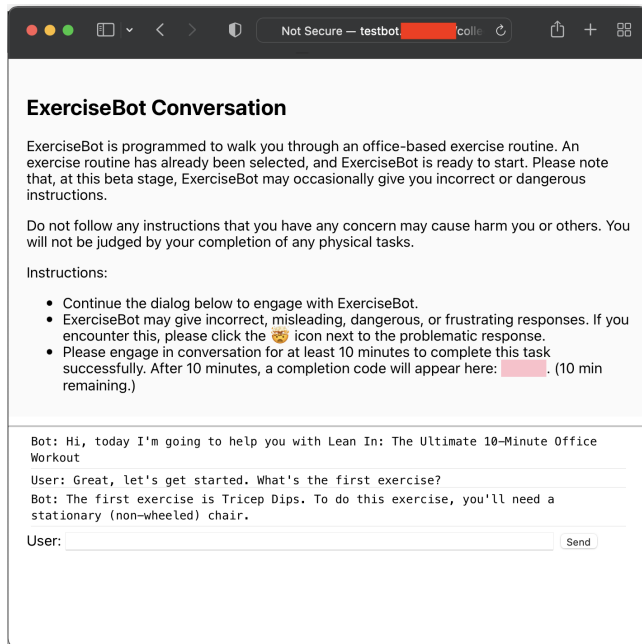
**Figure 3: The various properties of this chatbot template describe the prompt *preamble* (the text prepended to the conversation dialog), instructions for *formatting* prior conversational turns into the LLM prompt, instructions for *prompting* and *extracting* the chatbot utterance from the LLM’s prediction, and other assorted parameters.**

have orange backgrounds and are overlaid with CATEGORY tags for easy identification. This particular example illustrates how the conversation visualizer shows a few useful properties of this set of conversations:

- The different ways this set of conversations arrives at the “Step forward until your butt clears the chair and your knees...” utterance, labeled with error tags SKIP and LANGUAGE.
- The context-sensitivity of errors, like the aforementioned SKIP tag, which indeed indicates that the first step was skipped in the rightmost 5 of the 6 conversation threads (shown in the top half of Fig. 6), but *not* to the leftmost thread, which includes the only utterance with the correct first step, “Scoot to the front of the chair...”
- The different utterances that different users use to push the conversation forward from step to step, as well as the requests they make, such as “Ok hang on while I get a chair”, that go heeded or unheeded by the chatbot.

In **prompt strategy development and testing** mode, BOTDESIGNER shows all problematic utterances (again optionally filtered by source and the presence of specific error tags) in context and allows the user to test a new template on any specific (or on all) utterances and see how the template changes affect problematic conversational turns.





**Figure 4: The conversation collection interface, used to collect sample conversations with a baseline chatbot from AMT workers.**

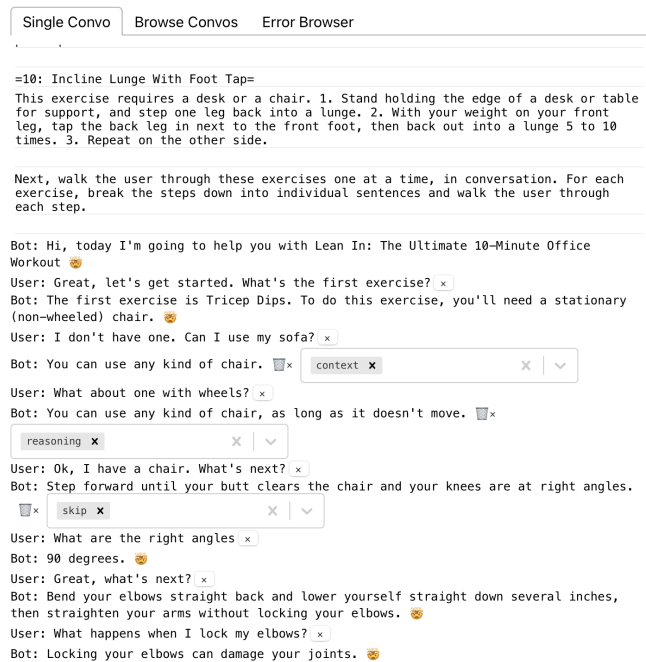
Figure 7 shows a screenshot of BOTDESIGNER’s *prompt testing* interface being used to evaluate a new prompt template. This view groups all tagged utterances (by tag) and displays the utterances in each group with two lines of context before and after each tagged utterance. Utterances with multiple tags are duplicated in each group. In the specific screenshot in Fig. 7, a new prompt template is being applied to baseline conversations for utterances bearing the `skip` tag. In this example, every utterance now includes the correct first step; additionally, the second conversation snippet’s utterance has also changed to explicitly address the user’s prior utterance requesting that the bot “[...]hang on while I get a chair”.

The ability to quickly see the effects of prompt changes allows designers to rapidly iterate on ideas and quickly eliminate approaches that don’t work for a specific utterance, or don’t work across a whole class of utterances, to converge on approaches that offer the most “bang for the buck” in terms of improved outcomes while avoiding regressions.

#### 4.4 Implementation Details

BOTDESIGNER is implemented as a React-based web application with a node.js-based backend, relying on OpenAI’s GPT-3 API as the underlying pre-trained language model. For consistency across tests, BOTDESIGNER always uses GPT-3’s `test-davinci-001` model with `TEMPERATURE`<sup>1</sup> set to 0.

<sup>1</sup>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.



**Figure 5: The annotation interface, supporting tagging of any chatbot utterance for later aggregation, examination, and prompt effect testing. This annotator view also supports extending any recorded conversation with new utterances, or forking a conversation: creating a new conversation from an existing one, but rolled back to an earlier user utterance in the conversation, and then continued with a new user utterance.**

Much of the implementation of the application consists of standard CRUD-style techniques, but a few specific implementation details bear mentioning:

**Conversation Visualizer.** To aid users in discovering common patterns across multiple discrete conversations between a bot and different users, BOTDESIGNER includes a conversation visualizer. We take inspiration from visualizing shared structure in written step-by-step tutorials [21, 33] and apply similar techniques to dialogues. Our visualization models a full set of conversations for a given recipe as individual paths through a Directed Acyclic Graph (DAG). Each conversational turn is modeled as a single node; where multiple conversations have identical utterances, those utterances are merged together into a single DAG node. (Short utterances of fewer than 20 characters, such as “OK” or “What’s next?” are not merged; these do not typically indicate any kind of useful similarity across conversations, as they too often occur in different contexts.)

Merging nodes in this manner, however, has the downside of introducing cycles into the conversation graph, if multiple conversations yield two merged nodes which appear in the opposite order across the conversations. For example, if in conversation 1 utterance A follows utterance B, but in conversation 2 it is B that follows A, then the merging algorithm will create a cycle: a path from A to B exists in conversation 1, while a path from B to A exists

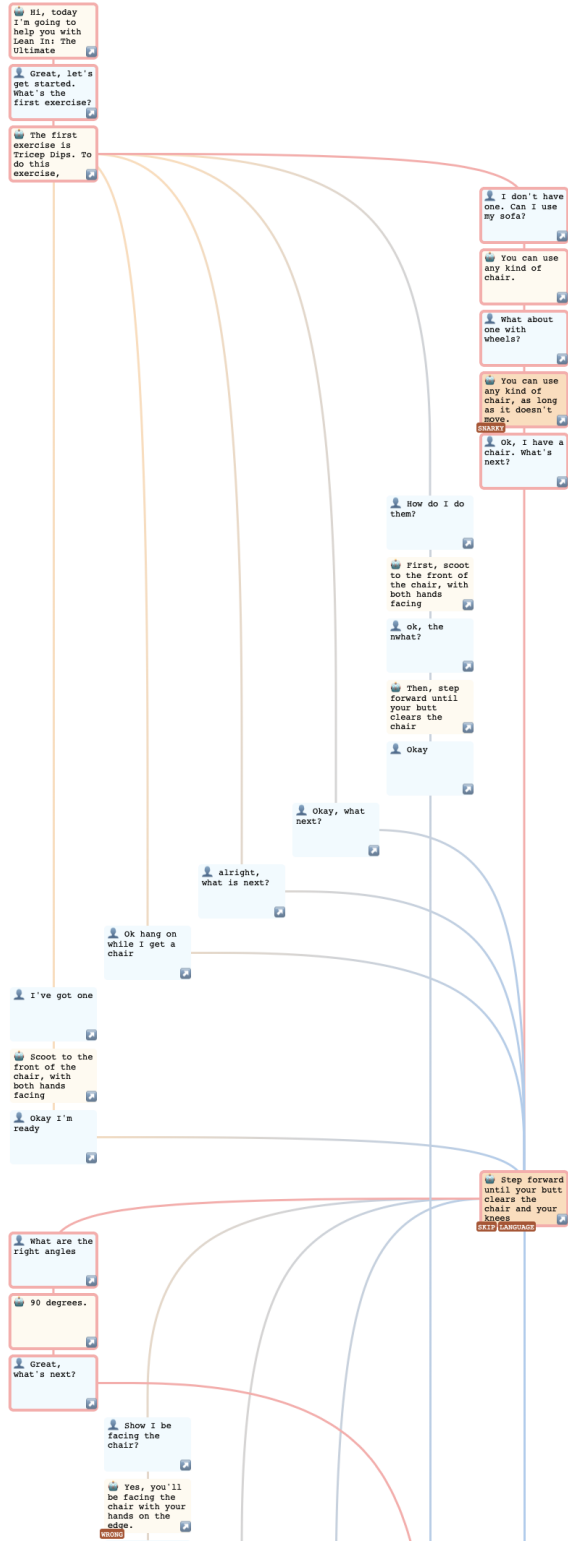


Figure 6: A sample of the *conversation visualizer* reflecting the first few turns of 12 conversations, half of which were “forked” and thus share a substantial prefix of turns.

in conversation 2. To resolve these, a “decycling” operation splits one of the two merged nodes back into separate nodes and updates the graph edges to preserve the original conversational flows.

The resulting conversational DAG is laid out and displayed using the d3-dag extension to d3.js.

**Regression Testing.** To evaluate whether a particular **template** change affects any of the identified problematic utterances, BOTDESIGNER replays conversations containing errors and displays any modified responses. Two implementation approaches are possible for this task: a system could either perform an “individual 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, 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 “individual 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.

## 5 EVALUATION

To evaluate the effectiveness of BOTDESIGNER in aiding conversational agent design, and to understand the value of Conversation Regression Testing, we ran a small ( $N = 3$  participants) qualitative pilot study with a design researcher (P1), a conversational agent designer (P2), and an NLP researcher (P3).

We ran this study primarily looking at two outcomes: how effectively could participants identify common or particular severe bugs or errors in a baseline chatbot, and how effectively could participants evaluate a new template for improvements.

### 5.1 Method

**Participants.** Since prompt-based chatbot design is not yet a common practice in industry, we recruited academic researchers with an interest in and experience with conversational agent design.

**Tasks.** We asked participants to perform two parts of the Conversation Regression Testing pipeline. We collected conversations in advance from AMT workers, and then asked participants to (1) browse the collected conversations to find errors and annotate them with categorization tags; (2) evaluate a “new” template, provided by us, with modified prompts, and report whether this new template resolved any of the errors participants had previously identified.

Participants were introduced to the tool and its basic use for about 10 minutes, asked to create some baseline conversations, and then asked to spend 10-15 minutes on each of the tasks above.

We recorded participants’ responses to using the tool and measured whether they detected a set of 5 error categories we previously identified in this dataset: (1) skipped steps, (2) ignorance of user expressions of pain, (3) ignorance of user expressions to wait until the

The screenshot shows the BOTDESIGNER interface for Conversation Regression Testing. At the top, there's a navigation bar with tabs for 'Single Convo', 'Browse Convo's', and 'Error Browser'. Below this, there's a 'Prefix' field and a 'show tags' button. The main area is divided into three columns. The left column shows a series of chatbot utterances, each with a 'SKIP' button and a 'Test' button. The center column shows the results of applying the modified chatbot template to the corresponding 'baseline' utterance. The right column contains the 'Template Editor' with a code editor showing the template code.

**Figure 7: An example of the *Conversation Regression Testing* panel of BOTDESIGNER. The left column shows individual *original* tagged chatbot utterances with individual **TEST** buttons, while the highlighted utterances in the center column show the results of applying the modified chatbot template (right-hand side code panel) to the corresponding “baseline” utterance (left column).**

user had completed some task (i.e., “hang on, let me get a chair”), (4) factually incorrect responses to questions, and (5) otherwise unhelpful responses. We also measured whether participants could identify which particular error categories were improved by the new template.

It bears noting that we did *not* ask users to engage in the task of prompt engineering; despite recent work exploring its potential, and our confidence in the value of large pre-trained LLMs as a design material, the pool of designers making use of prompt engineering and large pre-trained LLMs in the design of chatbots is small. Further, we did not want to spend time training participants in prompt engineering or depend on participants’ intuitions about

prompt changes to understand whether the *technique* of *Conversation Regression Testing* is effective at helping designers understand the impacts of *particular* prompt changes.

## 5.2 Findings

Overall, we found that each of our participants could effectively (1) find errors across conversations using BOTDESIGNER, and (2) evaluate whether a new prompt template improved outcomes across the identified errors. Here, we report some of the insights we gathered from our participants.

**5.2.1 Identifying Errors.** From our first participant (P1), we learned of an interest in tagging *effective* conversational turns in addition

to errors; this motivated the “regression testing” we use, and we subsequently found that **all** our participants were interested in tagging strong responses in addition to errors.

Two of our participants found all 5 categories of error (P1, P2), while one participant (P3) did not understand that tag names were for human use (not used as descriptions in some training process), and thus found only 3 of the 5 categories of error. All 3 participants found the tagging process straightforward, and P1 in particular appreciated the ways in which the conversations could be modified and rolled back: “oh, that’s useful!” (P1).

P1 also noted that determining whether some utterances were logically sound sometimes required substantial understanding of the underlying instructional task, which made catching errors a function of the willingness to manually scroll between the exercise template and the conversation.

Regarding the specific functionality used to understand the flow of conversations, P2 noted that the “I think this diagram [Ed: the conversation visualizer] has a real potential to help me understand what’s going on in the conversation [...] having a graph of all the conversations is really something valuable that I would appreciate.”

**5.2.2 Testing New Prompts.** All 3 participants were able to identify which classes of error were improved by the new bot template.

Our chatbot designer participant in particular (P2) interrupted the study halfway through to ask whether we could instead load up conversations *they had collected* and import a bot template *they had constructed* and to *continue the study with their template and data*: “You know I do have real life data, and we can use this [to improve my prompts.]” (P2) Though of course anecdotal, we consider this request to be a strong endorsement of the effectiveness of *Conversation Regression Testing* as a technique and BOTDESIGNER as a method for applying it.

After using the testing interface shown in Fig. 7 for the evaluation task, P2 notes:

I think you found out very interesting things. I didn’t think really about how I can control all the interactions...you know I use a high temperature for the chatbot, and I really like it because the conversations are becoming awesome with the new models, just fantastic, but I don’t have control. I don’t know what is produces, you know. **This kind of tool, as a plug-in for an AI system, that shows you a log of what happened on the system, and then you can this data to fine tune the user experience.**

Our observations of participants using BOTDESIGNER hint at the substantial value of systematizing the typical trial-and-error approach that makes it very challenging to assess prompt changes across multiple conversations rather than single turns at a time.

## 6 LIMITATIONS & FUTURE WORK

One fundamental assumption of the approach described here is that there is common structure across multiple dialogs. In step-by-step instructions, this is straightforward. In other conversation domains, how to align different conversations to common structure might be a research topic in itself.

Though probably helpful, the tested implementation of BOTDESIGNER does not present an aggregate picture of what classes of annotated utterances are improved or get worse, nor whether the changes in the produced utterances are meaningfully different or merely textually distinct.

We don’t yet offer tools for tracking evolution of utterances over time - if the interactive loop is about changing prompts, some changes will make things better, others will make things worse, and maybe some changes are modular, some aren’t. This probably requires tracking prompt state and responses over time

Future improvements to BOTDESIGNER could also include the use of large pre-trained LLMs to automate some tasks the designer currently performs, such as comparing baseline utterances with new utterances produced by an updated bot, or finding utterances with identical content but distinct text across conversations.

## 7 CONCLUSION

The combination of pre-trained large language models (LLM) and prompts offers exciting new opportunities for chatbot design. However, identifying robust and generalizable prompt strategies that can effectively improve conversational interactions has so far been challenging. Designers face challenges in both holistically analyzing the highly-contextual errors LLMs make across conversations, and in resolving the errors without unknowingly causing new errors in preceding or subsequent conversations. This paper advances on these critical challenges.

The primary contribution of this paper is the concept of *Conversation Regression Testing* for prompt strategy design. Without model retraining, UX improvements from prompts tend to be brittle. Identifying truly effective prompt strategies requires systematic methods for assessing their robustness and generalizability. Such methods have been missing in prompt-related HCI research. *Conversation Regression Testing* offers a first step in filling this critical gap.

The technical contribution of this paper lies in the techniques for implementing BOTDESIGNER. It presents a novel conversation visualization technique that visualizes common conversation patterns across many discrete conversations between an LLM and various users. This technique not only enabled BOTDESIGNER to aggregate LLM errors without losing error contexts, it can be useful for developing many other human-LLM interaction analysis or design tools. BOTDESIGNER ultimately implements an interface for *Conversation Regression Testing*, a technique that can be valuable for prototyping prompts for many other pre-trained LLM applications beyond conversational interactions.

## REFERENCES

- [1] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine* 35, 4 (2014), 105–120. <https://doi.org/10.1609/aimag.v35i4.2513>
- [2] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 337–346.
- [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 Alyafei, 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] Tony Bergstrom and Karrie Karahalios. 2009. Conversation Clusters: Grouping Conversation Topics through Human-Computer Dialog. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) (CHI '09). Association for Computing Machinery, New York, NY, USA, 2349–2352. <https://doi.org/10.1145/1518701.1519060>
- [5] Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. <https://www.wandb.com/> Software available from wandb.com.
- [6] Lukas Biewald. 2020. Experiment tracking with weights and biases. *Software available from wandb.ai* 2 (2020).
- [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 Doumouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kavin 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, Fereshete Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kudithipudi, 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 Muniyikwa, Suraj Nair, Avnika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Nieves, 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](https://arxiv.org/abs/2108.07258) [cs.LG]
- [8] Frederick P Brooks Jr. 1995. *The mythical man-month: essays on software engineering*. Pearson Education.
- [9] 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.
- [10] 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>
- [11] Justin Cheng and Michael S. Bernstein. 2015. Flock: Hybrid Crowd-Machine Learning Classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*. ACM Press, New York, New York, USA, 600–611. <https://doi.org/10.1145/2675133.2675214>
- [12] Justin Cranshaw, Emad Elwamy, 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.
- [13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- [14] 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).
- [15] Judith Donath. 2002. A Semantic Approach to Visualizing Online Conversations. *Commun. ACM* 45, 4 (apr 2002), 45–49. <https://doi.org/10.1145/505248.505271>
- [16] Jerry Alan Fails and Dan R Olsen Jr. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*. 39–45.
- [17] 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.
- [18] 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>
- [19] Bogyong 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>
- [20] 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>
- [21] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. 2012. Delta: A Tool for Representing and Comparing Workflows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) (CHI '12). Association for Computing Machinery, New York, NY, USA, 1027–1036. <https://doi.org/10.1145/2207676.2208549>
- [22] Hunter Lang, Monica Agrawal, Yoon Kim, and David Sontag. 2022. Co-training Improves Prompt-based Learning for Large Language Models. [arXiv:2202.00828](https://arxiv.org/abs/2202.00828) [cs.CL]
- [23] Mina Lee, Percy Liang, and Qian Yang. 2022. CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities. In *Proceedings of the 2022 CHI conference on human factors in computing systems*.
- [24] Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical Details and Evaluation.
- [25] 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](https://arxiv.org/abs/2107.13586) [cs.CL]
- [26] 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>
- [27] 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).
- [28] Elijah Mayfield and Carolyn Penstein Rosé. 2013. Lightside: Open source machine learning for text accessible to nonexperts. invited chapter in the handbook of automated essay grading.
- [29] Swati Mishra and Jeffrey M Rzeszutarski. 2021. Designing Interactive Transfer Learning Tools for ML Non-Experts. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.
- [30] OpenAI. 2020. OpenAI GPT-3 playground: GPT-3 demo. <https://gpt3demo.com/apps/openai-gpt-3-playground>
- [31] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Preprint* (2022).
- [32] Kayur Patel, Naomi Bancroft, Steven M. Drucker, James Fogarty, Andrew J. Ko, and James Landay. 2010. Gestalt: Integrated Support for Implementation and Analysis in Machine Learning. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology - UIST '10*. ACM Press, New York, New York, USA, 37. <https://doi.org/10.1145/1866029.1866038>
- [33] Amy Pavel, Floraine Berthouzoz, Björn Hartmann, and Maneesh Agrawala. 2013. Browsing and Analyzing the Command-Level Structure of Large Collections of Image Manipulation Tutorials. *EECS Technical Report* (2013).
- [34] 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.
- [35] 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, Abheesh Sharma, Andrea Santilli, Thibault Fèvre, 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>
- [36] Timo Schick and Hinrich Schütze. 2021. True Few-Shot Learning with Prompts—A Real-World Perspective. *arXiv preprint arXiv:2111.13440* (2021).
- [37] 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>
- [38] Gina Danielle Venolia and Carman Neustaedter. 2003. Understanding sequence and reply relationships within email conversations: a mixed-model visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 361–368.
- [39] Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>.

- [40] 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*.
- [41] 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*.
- [42] 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>
- [43] Qian Yang, John Zimmerman, Aaron Steinfeld, and Anthony Tomasic. 2016. Planning Adaptive Mobile Experiences When Wireframing. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) (*DIS '16*). Association for Computing Machinery, New York, NY, USA, 565–576. <https://doi.org/10.1145/2901790.2901858>
- [44] Ou Jie Zhao, Tiffany Ng, and Dan Cosley. 2012. No forests without trees: Particulars and patterns in visualizing personal communication. In *Proceedings of the 2012 iConference*. 25–32.
- [45] Xiaoyong Zhu. 2018. *Fine-tune natural language processing models using Azure Machine Learning service*. <https://azure.microsoft.com/en-us/blog/fine-tune-natural-language-processing-models-using-azure-machine-learning-service/>