

Enabling Conversational Interaction with Mobile UI using Large Language Models

Bryan Wang*
University of Toronto
Toronto, ON, Canada
bryanw@dgp.toronto.edu

Gang Li
Google Research
Mountain View, CA, USA
leebird@google.com

Yang Li
Google Research
Mountain View, CA, USA
liyang@google.com

ABSTRACT

Conversational agents show the promise to allow users to interact with mobile devices using language. However, to perform diverse UI tasks with natural language, developers typically need to create separate datasets and models for each specific task, which is expensive and effort-consuming. Recently, pre-trained large language models (LLMs) have been shown capable of generalizing to various downstream tasks when prompted with a handful of examples from the target task. This paper investigates the feasibility of enabling versatile conversational interactions with mobile UIs using a single LLM. We designed prompting techniques to adapt an LLM to mobile UIs. We experimented with four important modeling tasks that address various scenarios in conversational interaction. Our method achieved competitive performance on these challenging tasks without requiring dedicated datasets and training, offering a lightweight and generalizable approach to enable language-based mobile interaction.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**.

KEYWORDS

Large Language Models, Conversational Interaction, Mobile UI

ACM Reference Format:

Bryan Wang, Gang Li, and Yang Li. 2023. Enabling Conversational Interaction with Mobile UI using Large Language Models. 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, 17 pages. <https://doi.org/10.1145/3544548.3580895>

1 INTRODUCTION

Interacting with computing devices using natural language is a long-standing pursuit in human-computer interaction [6, 13, 22]. Language, as both the input and the output, allows users to efficiently communicate with a computing system and access its functionalities when other I/O modalities are unavailable or cumbersome. The interaction paradigm is particularly useful for users

with motor or visual impairments or situationally impaired when occupied by real-world tasks [48, 54]. Intelligent assistants, e.g., Google Assistants and Siri, have significantly advanced language-based interaction for performing simple daily tasks such as setting a timer. Despite the progress, these assistants still face limitations in supporting conversational interaction in mobile UIs, where many user tasks are performed [31]. For example, they cannot answer a user's question about specific information displayed on the screen [18]. Achieving such capabilities requires an agent to have a computational understanding of graphical user interfaces (GUIs), which is absent in existing assistants.

Prior research has investigated several important technical building blocks to enable conversational interaction with mobile UIs, including summarizing a mobile screen for users to quickly understand its purpose [50], mapping language instructions to UI actions [37, 41, 44] and modeling GUIs so that they are more amenable for language-based interaction [34, 39, 50, 55, 58]. However, each of them only addresses a limited aspect of conversational interaction and requires considerable effort in curating large-scale datasets and training dedicated models. [37, 38, 40, 50]. Furthermore, there is a broad spectrum of conversational interactions that can occur on mobile UIs, as Todi et al. revealed [49]. Therefore, it is imperative to develop a lightweight and generalizable approach to realize conversational interaction.

Recently, pre-trained large language models (LLMs) such as GPT-3 [7] and PaLM [9] have demonstrated abilities to adapt themselves to various downstream tasks when being *prompted* with a handful of examples of the target task. Such generalizability is promising to support diverse conversational interactions without requiring task-specific models and datasets. However, the feasibility of doing so is unclear. Little work has been conducted to understand how LLMs, trained with natural languages, can be adapted to GUIs for interaction tasks. Therefore, we investigate in this paper the viability and the how-to of utilizing LLMs to enable diverse language-based interactions with mobile UIs.

We categorized four mobile UI conversational interaction scenarios, which guided our experimental task selection. We developed a set of prompting techniques to prompt LLMs with mobile UIs. Since LLMs only take text tokens as input, we contribute an algorithm to generate the text representation of mobile UIs. Our algorithm uses depth-first search traversal to convert the Android UI's view hierarchy, i.e., the structural data containing detailed properties of UI elements, into HTML syntax. We conducted comprehensive experiments with four pivotal modeling tasks, including *Screen Question-Generation*, *Screen Summarization*, *Screen Question-Answering*, and *Mapping Instruction to UI Action*. The experimental results show that our approach achieves competitive performance

*Work done during an internship at Google Research.

Permission to make digital or hard copies of part or all 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. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
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/10.1145/3544548.3580895>

using only *two data examples* per task. Notably, our work is the first to investigate methods to enable the *Screen Question-Generation* and *Screen Question-Answering* tasks in literature, setting benchmark performances. Furthermore, the human evaluation of the *Screen Summarization* task demonstrates that our method generates more accurate screen summaries compared to Screen2Words [50], the benchmark model trained with tens of thousands of examples. In summary, the evaluation results from a suite of modeling tasks validate the feasibility and effectiveness of our approach in enabling conversational interaction with mobile UIs.

More broadly, our study demonstrates LLMs’ potential to fundamentally transform the future workflow of conversational interaction design. Using our prompting method, interaction designers and developers can quickly prototype and test novel language interactions with users, saving time and resources before investing in dedicated datasets and models. Our experiments are based on open-source Android UI datasets, including RICO [12], Screen2Words [50], and PixelHelp [37]. We also open-source the code¹ of our algorithm that converts the Android view hierarchy to HTML syntax to allow future work to replicate and build upon our work. Example prompts created using our algorithm can be found in the appendix. In summary, our paper makes the following contributions:

- Our work is the first investigation for using LLMs to enable conversational interaction on mobile UIs, which advances the understanding of using LLMs for interaction tasks.
- We designed a novel method for feeding GUIs to LLMs—that is pre-trained for natural language—and a set of techniques to prompt LLMs to perform a range of conversational tasks on mobile UI screens. These techniques produce competitive performance; we open-source the code so others can immediately use them in their work.
- We experimented with four pivotal modeling tasks, demonstrating the feasibility of our approach in adapting LLMs for conversational GUI interaction and potentially lowering the barriers to developing conversational agents for GUIs.

2 RELATED WORK

2.1 Bridging GUIs with Natural Language

There has been increasing interest in using machine learning to bridge graphical user interfaces and natural language for use cases such as accessibility and multimodal interaction. For example, Widget Captioning [38] and Screen Recognition [58] predict semantically meaningful alt-text labels for GUI components. Screen2Words [50] took a step further to predict text summaries that concisely describe the entire screen using multimodal learning. Leiva et al. proposed a vision-only approach to generate templated language descriptions of UI screenshots [28]. Li et al. [37] uses a transformer-based model to map natural language instructions to mobile UI action sequences. These prior works typically train a model dedicated to the task based on a sizeable dataset collected. In contrast, our work leverages the few-shot learning ability of LLMs to enable language-based UI tasks by providing a small number of examples. To achieve this, we propose a novel method to represent the UI so that an LLM pre-trained for natural language can efficiently process

it. Another relevant body of work is to develop a conversational or multimodal agent that can help the user accomplish mobile tasks [31–33, 35]. For example, SUGILITE [31] enables users to create task automation on smartphones by user demonstration and perform the tasks through a conversational interface. KITE [33] helps developers create task-oriented bots templates from existing apps. Our work shows that LLMs can enable versatile language-based interactions when prompted with exemplars for different tasks, lowering the threshold for developing versatile multimodal agents.

2.2 Prompting Pre-trained Large Language Models

Finetuning pre-trained task-invariant models such as BERT has been a common practice to adapt large models for specific tasks. However, GPT-3 [7] introduced a new norm for leveraging pre-trained language models for downstream tasks through in-context few-shot learning, i.e., prompting. By prompting a pre-trained model with only a few examples, it can generalize to various tasks without updating the parameters in the underlying model. Recently studies have shown that prompting is one of the emergent abilities that appear only when the model size is large enough [52]. While prompting LLMs may not always outperform benchmark models, it provides a lightweight method to achieve competitive performance on various tasks [7, 9]. When the prompt consists of N pairs of input and output exemplars from the target tasks, it is referred to as N -shot learning, and providing more shots typically improves performance [7, 9]. Additionally, various prompting paradigms have been proposed to elicit logical reasoning from the language model [24, 51, 53, 59], which is useful for tasks like solving math problems. For example, Chain-of-Thought prompting [53] proposes to use the models to generate intermediate results (i.e., a chain of thoughts) before generating the final output. The core idea resembles the divide-and-conquer method in algorithms, which breaks more complicated problems into subproblems that can be solved more easily. Prompting LLMs remains an ongoing research topic in the community. Our work builds upon prior work to contribute a set of prompting techniques designed to adapt LLMs to mobile UIs.

2.3 Interactive Applications of Large Language Models

LLMs have been applied to enable a broad range of language-related interactive applications in the HCI community [2, 10, 11, 20, 21, 23, 25–27, 42, 56]. For example, Chang et al. [10] proposed TaleBrush, a generative story ideation tool that uses line sketching interactions with a GPT-based language model for control and sensemaking of a protagonist’s fortune in co-created stories. Stylette [23] allows users to modify web designs with language commands and uses LLMs to infer the corresponding CSS properties. Lee et al. [25] present CoAuthor, a dataset designed to reveal GPT-3’s capabilities in assisting creative and argumentative writing. Since LLMs can encode a wealth of semantic knowledge, they have also been used to support physical applications. For example, SayCan [2] extracts and leverages the knowledge priors within LLMs to execute real-world, abstract, long-horizon robot commands. Our work represents the first contribution of applying LLMs to enable conversational interactions on mobile UIs.

¹<https://github.com/google-research/google-research/tree/master/llm4mobile>

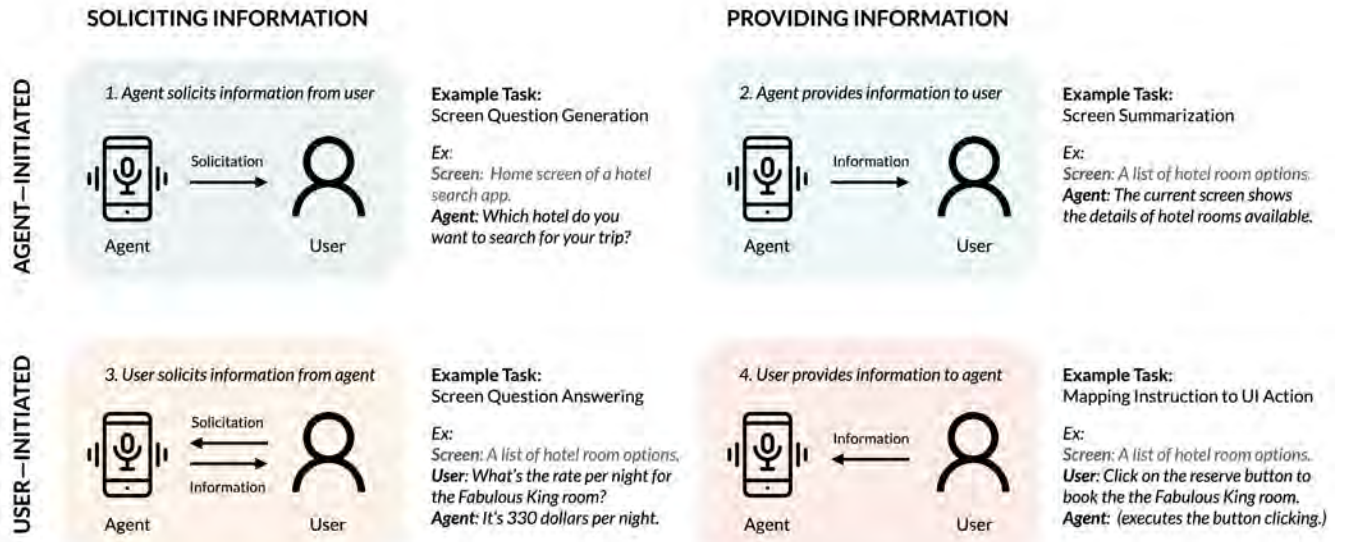


Figure 1: The categorization of different conversation scenarios when user and agent interact to complete tasks on mobile UIs. The categorization has two axes: initiative and purpose. An agent can take the initiative to solicit new information from the user or to provide information to the user, and vice versa. We included an example task for each category where the agent's action represents the target output from LLMs.

3 CONVERSATION FOR MOBILE UI TASKS

Conversational interaction with mobile devices is typically embodied as human users exchanging information with a conversational agent. We developed a conceptual framework categorizing four conversation scenarios between users and agents when performing mobile tasks. Our categorization has two dimensions: *Initiative* and *Purpose*. As shown in Figure 1, a conversation can be mixed-initiative [17], either initiated by the agent or the user. The purpose of initiated conversation can be either soliciting or providing information. The categorization lays the foundation for determining important modeling tasks to investigate. We focus on studying how LLMs can enable the interaction capabilities of a conversational agent based on a mobile UI, e.g., providing language responses or performing UI actions on behalf of the user. For simplicity, we limit our study to unit conversations which include, at most, a single turn from the user and agent. More complex, multi-turn conversations are beyond the scope of this paper and will be explored in future research, which we discuss in section 6.3. We now introduce each conversation category and the associated example modeling tasks, including 1) *Screen Question-Generation*, 2) *Screen Summarization*, 3) *Screen Question-Answering*, and 4) *Mapping Instruction to UI Action*.

3.1 Agent-initiated Conversation

When an agent initiates a conversation, it can be either soliciting or providing information essential for the user to perform tasks on a mobile UI.

3.1.1 Agent solicits information from user. Mobile UIs often request users to input information relevant to their goals. For example,

destination city or *travel dates* in the scenario of booking a hotel. On mobile UIs, the information request is typically made through input text fields. A conversational agent should be able to similarly solicit essential information from users using natural language. For example, asking users questions like "Which hotel do you want to search for?" or "What is the check-in date of your stay?" We refer to this type of task as *Screen Question-Generation* since the questions should be generated based on the UI screen contexts.

3.1.2 Agent provides information to user. A key function of GUIs is to convey information to users through visual means. Similarly, conversational agents should be able to articulate the information presented in GUIs using language. Given that UIs contain a vast amount of information and user needs vary [49], there are numerous ways to deliver screen information. An example is *Screen Summarization* [50], which provides a short description of the purpose of the current screen, e.g., "A list of hotel rooms available at W San Francisco.", or "A step-by-step recipes of butter chicken". The descriptions can help users quickly understand the UI when visual information is unavailable.

3.2 User-initiated Conversation

Users can also initiate conversations to request information or proactively provide information for the agent to process.

3.2.1 User solicits information from the agent. Users should be able to solicit screen information from the agent through conversation. When the information request is done through questions such as "What's the rate of the hotel room with a king-size bed?", the agent should respond with appropriate answers such as "\$330 per night",

based on the information presented on the screen. We call this type of conversational interaction *Screen Question-Answering*, similar to visual [5] or text question-answering [45] but instead based on a mobile UI. *Screen Question-Answering* is beneficial in situations where there is a large amount of text on a screen, such as search results, making it difficult to locate specific information. It is also useful for individuals with disabilities who rely on screen readers to access screen text. Instead of waiting for the screen reader to scan through all the content on the screen until the relevant information is found, they can simply ask for the needed information.

3.2.2 User provides information to the agent. Users can initiate conversations with the agent by providing new information. After receiving the messages, the agent should respond accordingly based on the current UI context using language and/or mobile actions. A representative task of this type of conversation is *Mapping Instruction to UI Action*. For example, when a user who is presented with a hotel booking screen says, "Click on the Reserve button to book the Fabulous King room", the agent should understand the user's intent and the UI contexts in order to click the corresponding button. This type of interaction allows users to control the devices when touch inputs are unavailable [54].

4 PROMPTING LARGE-LANGUAGE MODELS FOR MOBILE UI TASKS

We introduce a class of prompting techniques designed to adapt LLMs to mobile UIs to enable conversational interaction. LLMs support in-context few-shot learning via *prompting*—instead of finetuning or re-training models for each new task, one can prompt an LLM with a few input and output data exemplars from the target task [7, 9, 53, 59]. For many natural language processing tasks such as question-answering or translation, few-shot prompting performs competitively with benchmark approaches [7]. However, the methodology for prompting LLMs with mobile UIs has yet to be established and presents several challenges. Firstly, language models can only take text input, while mobile UIs are multimodal, containing text, image, and structural information in their view hierarchy data and screenshots. Moreover, directly inputting the view hierarchy data of a mobile screen into LLMs is not feasible as it contains excessive information, such as detailed properties of each UI element, which can easily exceed the input length limits of LLMs. Furthermore, the mobile UIs encapsulate the logic of target user tasks [36]. Therefore, logical reasoning based on UI contexts is essential for the model to support conversations toward task completion. These unique aspects of mobile UIs pose two open problems for designing prompts:

- (1) How to represent mobile UIs in texts to leverage the few-shot prompting capability of LLMs?
- (2) How to elicit reasoning based on the mobile UIs when needed?

We respond to these questions by describing our proposed prompting techniques and their design rationales. Prompting LLMs remains an ongoing research problem. As the first to investigate prompting with mobile UI, we provide a strong baseline approach and encourage future work to build upon our design and further study the open problems.

4.1 Screen Representation

4.1.1 Representing View Hierarchy as HTML. There are various ways to represent a mobile UI in text, e.g., concatenating all the text elements on the UI into a token sequence or using natural language sentences to describe UI elements, such as "a menu button in the top left corner." To design our screen representation, we leverage the insight that if a prompt falls within the training data distribution of a language model, it is more likely for few-shot learning to perform. This is because LLMs are trained to predict the subsequent tokens that maximize the probability based on the training data. LLMs' training data is typically scraped from the web, including both natural language and code. For example, 5% of PaLM's [9] training data was scraped from GitHub, including 24 common programming languages such as Java, HTML, and Python [9]. Therefore, we represent a mobile UI in texts by converting its view hierarchy data into HTML syntax. HTML is particularly suitable for representing mobile UIs as it is already a markup language representing web UIs. The conversion is conducted by traversing the view hierarchy tree using a depth-first search. We detail our conversion algorithm in the following sections. Note that since the view hierarchy is not designed to be represented in HTML syntax, a perfect one-to-one conversion does not exist. In contrast, our goal is to make the converted view hierarchy similar to the HTML syntax to generate a data representation closer to the training data distribution.

4.1.2 View Hierarchy Properties. Converting a mobile UI's view hierarchy into HTML syntax can preserve the detailed properties of UI elements and their structural relationship. The view hierarchy is a structural tree representation of the UI where each node, corresponding to a UI element, contains various properties such as the class, visibility-to-user, and the element's bounds. However, using all element properties will result in lengthy HTML text, which may exceed the input length limit of the language model, e.g., 1920 tokens for PaLM and 2048 tokens for GPT-3. Therefore, we use a subset of properties related to the text description of an element:

- **class:** Android object type such as TextView or Button.
- **text:** element text that is visible to the user.
- **resource_id:** text identifiers that describe the referenced resource.
- **content_desc:** content description that describes the element for accessibility purposes, i.e., the alt-text.

4.1.3 Class Mapping. We developed heuristics to map the Android classes to HTML tags with similar functionalities. We map TextView to the <p> tag as they are both used for presenting texts; all button-related classes such as Button or ImageButton are mapped to <button>. We map all image-related classes such as ImageView to , including icons and images. Lastly, we convert the text input class EditText to <input> tag. We focus on the most common element classes for simplicity, and the rest of the Android classes, including containers such as LinearLayout are mapped to the <div> tag.

4.1.4 Text, Resource_Id, and Content Description. We insert the text properties of Android elements in between the opening and closing HTML tags, following the standard syntax of texts in HTML. The resource_id property contains three entities: package_name, resource_type, and resource_name. Among them, resource_name

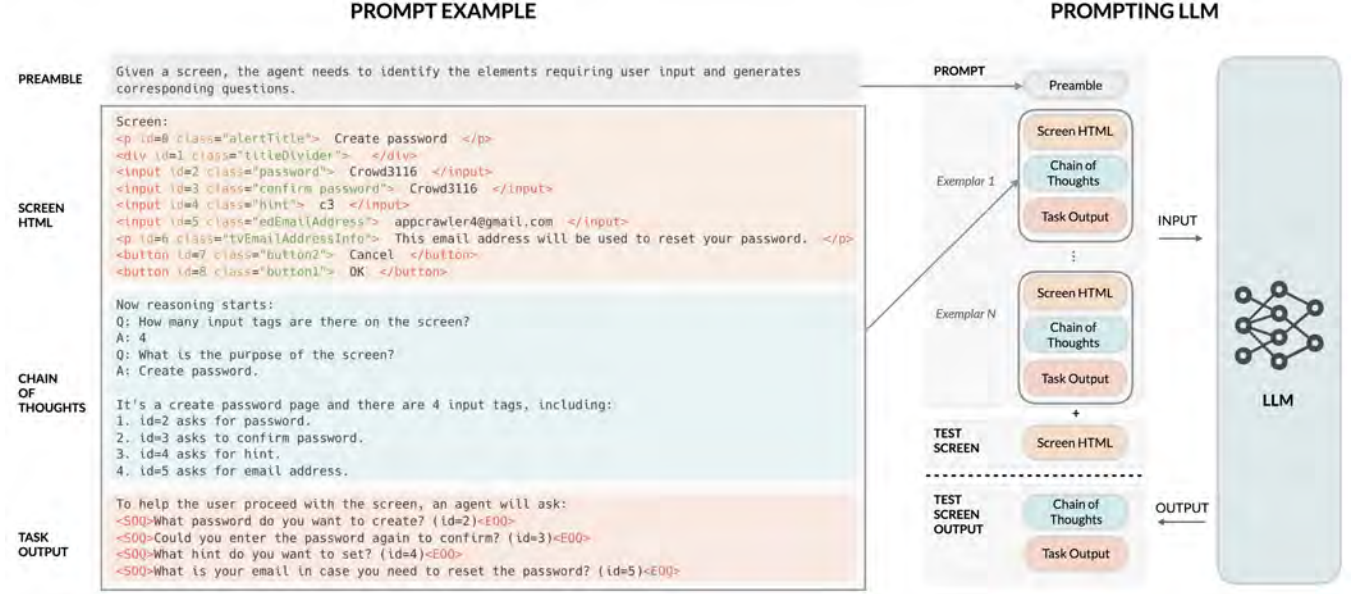


Figure 2: Left: An example illustrating the proposed prompt structure. A prompt starts with the preamble, which describes the task. Following the preamble, there will be zero or more task exemplars from the target tasks. Each exemplar consists of an input screen HTML, a chain of thoughts (if applicable), and a task-specific output. Additional exemplars are appended to the end of the previous ones. Right: An illustration of prompting large language models in our use cases. The prompt contains N exemplars from the target tasks. We append the screen HTML of the test screen to the end of the prompt. We then feed the prompt + test screen HTML as input into the LLM. The LLM then generates word tokens in an auto-regressive manner, which helps them capture the exemplars’ pattern to produce a chain of thoughts (if applicable) and task-specific output.

usually contains additional descriptions of an element’s functional-ity or purpose written by the developers. For example, in the Gmail app, an element with resource_name of "unread_count_textView" shows how many emails are unread, whereas a "date" means the element shows the date of receiving a mail. Such information helps the model to better understand the screen context. We insert the resource_name tokens that describe each element’s purpose as additional identifiers in the "class" attributes, which originally contain identifiers linked to a style sheet or used by JavaScript to access the element. Word tokens in resource_name are typically concatenated with underscores, which we replace as spaces when inserting. Lastly, we insert the content_desc as the "alt" attribute in the HTML tags when the property is present.

4.1.5 Numeric Indexes for Referencing. To help model referencing specific UI elements, we insert numeric indexes to each element as the "id" attribute. The indexes are generated with the depth-first search order in the view hierarchy tree. For tasks such as predicting which button to click based on language instructions, the model can refer to elements using numeric indexes, which is more efficient and space-saving than spelling out the complete HTML tag.

4.2 Chain-of-Thought Prompting

Mobile UIs encapsulate the logic of user tasks [36]; therefore, it is vital for models to perform reasoning when used for conversational interaction. LLMs have demonstrated abilities to reason [2, 7, 9] as they captured real-world knowledge during training with a large

number of texts. Recent work further shows that LLM’s reasoning ability can be improved by generating and chaining intermediate results to obtain the final answers, namely, Chain-of-Thought prompting [53]. The idea is straightforward, i.e., simply appending a chain of thoughts describing intermediate results before the answers in the prompt. The model would then follow the patterns to generate a chain of thoughts during inference. Chain-of-Thought prompting has been shown to be helpful for reasoning tasks. The results are also more interpretable as the model would articulate its thought process before coming up with the answer. However, prior work has not investigated whether it can facilitate reasoning in generating conversations based on mobile UIs. Therefore, we incorporate the method in our experiments. Chain-of-Thought prompting has been shown effective for tasks that require the model to reason across multiple steps. In our early investigations, we found that this technique does not improve performance for tasks where the output can be directly obtained from the input screen HTML. Therefore, in our experiments, we only used the technique for *Screen Question-Generation* whose task setup requires multi-step reasoning.

4.3 Prompt Structure

We follow a similar prompt structure proposed in [7]. Each prompt starts with a *preamble* which explains the prompt’s purpose. The preamble is followed by multiple exemplars consisting of the input, a chain of thought (if applicable), and the output for each task. Each exemplar’s input is a mobile screen in the HTML syntax. To better leverage few-shot learning while complying with LLM’s

input length limits, we only show the leaf nodes visible to the users, as non-leaf nodes are usually containers that do not contain textual information. Following the input, a chain of thoughts is provided to elicit logical reasoning from LLMs, if applicable to the task. The output is the desired outcome for the target tasks, e.g., a screen summary or an answer to the question asked by the user. Figure 2-left shows an example of a 1-shot prompt. Few-shot prompting can be achieved with more than one exemplar included in the prompt. During prediction, we feed the model with the prompt with a new input screen appended at the end. Therefore, for N -shot learning, the prompt will consist of a preamble, N exemplars, and the test screen for prediction, as shown in Figure 2-right.

5 FEASIBILITY EXPERIMENTS

As shown in Figure 3, we demonstrate the feasibility of using LLMs to enable conversations on GUIs through experiments with four tasks we introduced in section 3: 1) *Screen Question-Generation*, 2) *Screen Summarization*, 3) *Screen Question-Answering*, and 4) *Mapping Instruction to UI Action*. Following the common practices of few-shot prompting [7, 53], we select a handful of exemplar data to construct prompts for each task. We then evaluate the effectiveness using task-specific metrics detailed in each experiment. All the studies were conducted with the PaLM model [9], which performs similarly to other LLMs such as GPT-3 [53]. The PaLM model is trained with a maximum input length of 1920 tokens. Therefore, we limit the number of exemplars in a prompt to be two at most, excluding the test screen, to avoid exceeding the length limit. The experiments aim to understand what can be achieved by simply prompting LLMs with a few exemplars from the target tasks and compared to the baseline or benchmark if available.

5.1 Screen Question-Generation

5.1.1 Task Formulation. Given a mobile UI screen, the goal of screen question-generation is to synthesize coherent, grammatically-correct natural language questions relevant to the UI elements requiring user input. The task occurs when the agent requests user input to proceed on the UI.

5.1.2 Prompt Construction. Figure 2 shows an example prompt we used to generate questions. We use a preamble of "Given a screen, the agent needs to identify the elements requiring user input and generates corresponding questions.". We used chain-of-thought techniques to generate three intermediate results 1) input fields count, 2) screen summary, and 3) input enumeration. The input field counts were fed with a ground truth count extracted from the screen HTML. We found this step essential to prevent the model from omitting some input elements and only generating a subset of questions. Next, we asked the model to summarize the screen's purpose, which produces the screen context that helps provide details in the generated questions. Subsequently, the model enumerates which elements are asking for what information. After the chain of thoughts, the model generates the questions, enclosed by <SOQ> and <EOQ> tokens, representing start-of-question and end-of-question, respectively. The tokens are used as delimiters for conveniently parsing the questions from the output texts generated by the model. We use similar ways to insert special tokens for parsing model

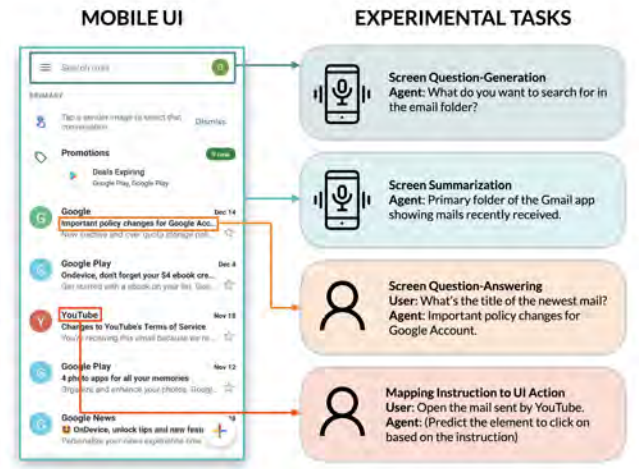


Figure 3: An illustration of the four UI modeling tasks we experimented on a single mobile UI. The tasks include *Screen Question-Generation*, *Screen Summarization*, *Screen Question-Answering*, and *Mapping Instruction to UI Action*. Each task is associated with a conversation category described in Section 3. Bounding boxes on the mobile UI highlight elements relevant to the example conversational interactions from each task.

output in the rest of the experiments. An example prompt can be found in appendix A.1.

5.1.3 Experimental Setup. We aim to understand the quality of LLMs for natural language generation based on UI elements. Since there is currently no existing dataset for screen question generation, we followed the common practice of evaluating language generation quality with human ratings. We randomly sampled 400 test screens from the RICO dataset [12]. Each of these screens contains at least one `EditText` element, representing the text input field for users to enter information on the UI. We randomly selected another two screens from the RICO dataset as exemplars to include in the prompt. An `EditText` element represents an input field for the user to enter information, and we generate questions for every input field. We generate questions from the test screens using a prompt constructed with two exemplars. Some screens contain multiple input fields, and sometimes several of them are relevant and can be asked collectively. For example, three fields asking for the birth year, month, and date can be combined into a single question as "when is your birthday?". Combining questions can lead to a more efficient conversation between an agent and a user. Therefore, we include an exemplar that combines relevant questions in the prompt to see if the model will also learn to combine relevant questions.

We compare LLM's results with a rule-based approach that uses words in `resource_id`, referred to as `res_tokens`, to fill in the template of "What is {`res_tokens`}?". We use `res_token` instead of `text` because most text input fields are blank by default, and `res_token` contains the most meaningful description of an input field. We recruited 17 raters who work as professional data labelers at Google to provide ratings. To ensure the quality of the labels, a group of

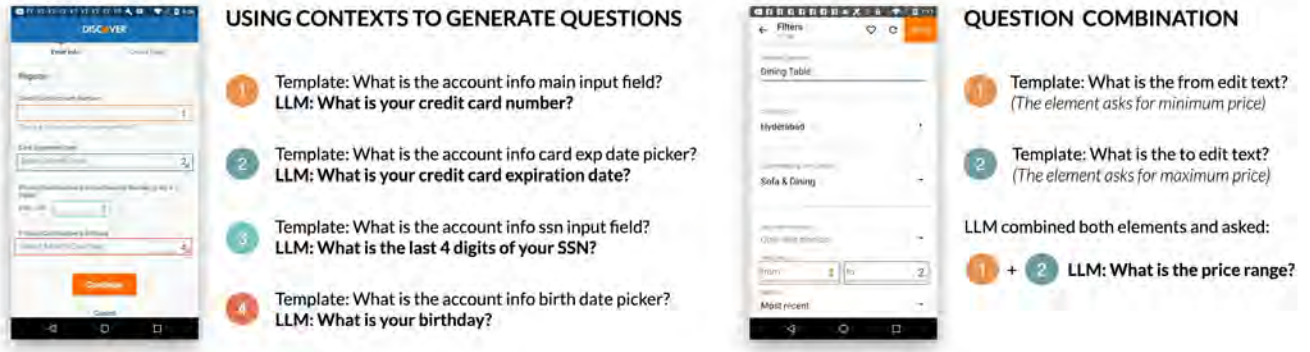


Figure 4: Example screen questions generated by the LLM. Left: The LLM can utilize screen contexts to generate grammatically-correct questions relevant to each input field on the mobile UI, while the template approach falls short. Right: We observed that the LLM could use its prior knowledge to combine multiple related input fields to ask a single question. The example shows the LLM combining the minimum and maximum prices fields into a single question asking about the price range. Elements relevant to each question are highlighted in corresponding colors and numbered indexes.

Table 1: Grammar correctness, UI relevance, and question coverage results from the screen question-generation experiment.

Method	Grammar	Relevance	Coverage F1
Template	3.60 ($\sigma=0.69$)	84.1%	100%
LLM	4.98 ($\sigma=0.07$)	92.8%	95.9%

quality audits sampled and reviewed 5% of the total number of questions answered by every rater. We provided a UI screenshot of a mobile UI and a generated question for each labeling task. The `EditText` element associated with the question is highlighted with a bounding box. We solicited human ratings on whether the questions were grammatically correct and relevant to the input fields for which they were generated. In addition to the human-labeled language quality, we automatically examined how well LLMs can cover all the elements that need to generate questions. The evaluation metrics include the following:

- **Grammar Correctness:** How correct is the grammar of a generated question? Are the sentences intelligible and plausible? This metric tests the language generation quality in general and is rated on a 5-point Likert scale with 1 as completely incorrect and 5 as completely correct.
- **UI Relevance:** Whether a generated question is relevant to the highlighted UI element. This metric tests whether the connection between a UI element and a question is correctly established by the model, which is rated on a binary scale as either relevant or not relevant.
- **Question Coverage:** How well can the model identify the elements on the screen that need question generation? This metric is automatically computed by comparing the indices of ground truth input elements with those identified by the model within the chain of thoughts.

5.1.4 Results. We evaluated 931 questions for both the LLMs and the template-based approach. Three different human raters examined each question to obtain aggregated scores. Table 1 shows the results of our evaluation. Our approach achieves an almost perfect average score of 4.98 on grammar correctness, while the rule-based approach receives a 3.6 average rating. A Mann-Whitney U test shows that the difference between the two methods is statistically significant ($p < 0.0001$). LLMs also generate 8.7% more relevant questions compared to the baseline. In terms of questions coverage, our approach achieves an F1 score of 95.9% (precision = 95.4%, recall = 96.3%). Since the rule-based method iterates through every input field to generate questions, its question coverage is naturally 100%. Altogether, the results show that our approach can precisely identify input elements and generate relevant questions that are intelligible. We further analyzed the model behaviors, and our results revealed interesting emergent abilities of LLMs. When generating a question for a field, the model considers both the input field element and the *screen context* (information from other screen objects). For example, Figure 4 shows how the model leveraged screen contexts to generate four questions for the input fields on a credit card register screen. While the baseline outputs use the *ref_tokens* to convey somewhat relevant information, they are less intelligible than the LLM output and do not articulate the specific information requested by the fields. In contrast, all four questions generated by LLM are grammatically correct and ask for relevant information. For Question 3 in Figure 4, the LLM additionally uses the texts above the input field to ask for the “last 4 digits of SSN”. The model also blends the screen contexts into the generated questions. For instance, Question 2 asks for “credit card expiration date,” while the texts above did not mention the word “credit.” We also observed that the model exhibits the behavior of combining relevant fields into a single question on three test screens. For example, Figure 4 shows the model can combine two input elements asking for minimum and maximum values for price into a single question “What is the price range?”. In practice, combining questions can lead to more efficient communication between users and agents.

Table 2: Screen summarization performance on automatic metrics.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr	ROUGE-L	METEOR
0-shot LLM	7.8	6.4	5.9	5.7	1.5	3.4	4.5
1-shot LLM	42.3	21.1	14.8	12.1	40.9	28.9	15.3
2-shot LLM	45.0	25.1	17.6	14.1	39.9	33.0	17.7
Screen2Words [50]	65.5	45.8	32.4	25.1	61.3	48.6	29.5

5.2 Screen Summarization

5.2.1 Task Formulation. Screen summarization was proposed in [50] as the automatic generation of descriptive language overviews that cover essential functionalities of mobile screens. The task helps users quickly understand the purpose of a mobile UI, which is particularly useful when the UI is not visually accessible.

5.2.2 Prompt Construction. We use a preamble of "Given a screen, summarize its purpose." We did not use chain-of-thought prompting as no intermediate result is needed to be generated for the task. We used N pairs of screen HTML and corresponding summary following the preamble, where $N = 0, 1, 2$ represents N -shot learning. The output summaries are enclosed by special tags <SOS> and <EOS>, meaning the start and end of a summary, respectively. An example prompt can be found in appendix A.2.

5.2.3 Experiment Setup. We use the Screen2Words dataset [50] to test LLM's ability to summarize screens. The dataset contains human-labeled summaries for more than 24k mobile UI screens, each with five summary labels. To gauge the quality of our approach, we test the performance of using LLMs to summarize screens with Screen2Words' test set, consisting of 4310 screens from 1254 unique apps, which was used by the benchmark. We randomly sampled two screens from the dataset and one of their corresponding summaries as the exemplars for prompt construction. We use the same automatic metrics reported in the original paper, including BLEU, CIDEr, ROUGE-L, and METEOR. As prior work [14, 50] has found that automatic scores may not correlate well with human perception of summary quality, we additionally conducted a human evaluation to solicit subjective feedback on the summaries generated by both LLM and the benchmark model Screen2Words [50]. We recruited 37 annotators using the same process specified in section 5.1.3. We presented annotators with a mobile UI screen and two screen summaries during the labeling process. To avoid bias, we randomly assigned the summaries generated by LLM and Screen2Words to be either Summary 1 or Summary 2, without revealing which summary was generated by which model. We instructed annotators to choose the summary that best *accurately* summarizes the mobile screen. The annotators were given three options: (a) Summary 1, (b) Summary 2, and (c) Equal or Very Similar. They were instructed to choose the third option only when it was difficult to judge the differences between the quality of the two summaries. The rating study was conducted on the full test dataset, with each screen receiving three ratings from different annotators.

5.2.4 Results. Table 2 shows the results of screen summarization on automatic metrics. The model could not generate meaningful summaries in the zero-shot setting (not providing any exemplar).

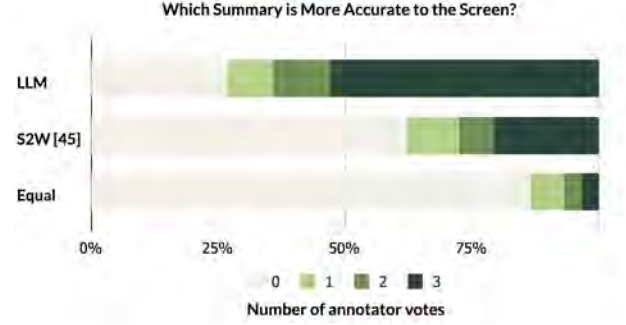


Figure 5: Annotator vote distribution across all test screens. LLM summaries are more accurate than those of the benchmark model for 64.1% of screens across the test set. This choice is unanimous between three labelers for more than half of the screens (52.8 %).

This result is expected as the LLM's training data may not have covered the task of screen summarization. When provided with the model one exemplar, the performance significantly boosted across all metrics. More examples in the prompt provide marginally higher scores. The average length of summaries generated by our two-shot LLM model was 7.15 words (STD=2.68). In comparison, the average length of summaries generated by the benchmark model was 6.64 (STD=1.98). Additionally, the LLM used a significantly larger number of unique words, at 3062, compared to the 645 unique words used by the benchmark model. The results indicate that LLM is capable of generating longer summaries with a wider range of language. When evaluating these models against automatic metrics, LLM generally scored less than the benchmark model. This is expected because the benchmark model was trained with the Screen2Words dataset, and the automatic metrics are based on token matching. A model trained on the dataset could achieve high scores by prioritizing the high-frequency words in the dataset. However, many high-frequency words and phrases, such as "display of," "screen", and "app", in the Screen2Words dataset do not meaningfully contribute to the summarization accuracy. As a result, a specialized model that learned to prioritize these frequent words may score well by synthesizing generic summaries, indicating the limitations of existing evaluation metrics.

In contrast, our human evaluation revealed that LLM generates summaries with higher perceived quality than those generated by the benchmark model, Screen2Words. Among all the human annotations, 63.5% of them rated LLM summaries as more accurate, while 28.6% voted for benchmark model summaries. Moreover, 7.9%

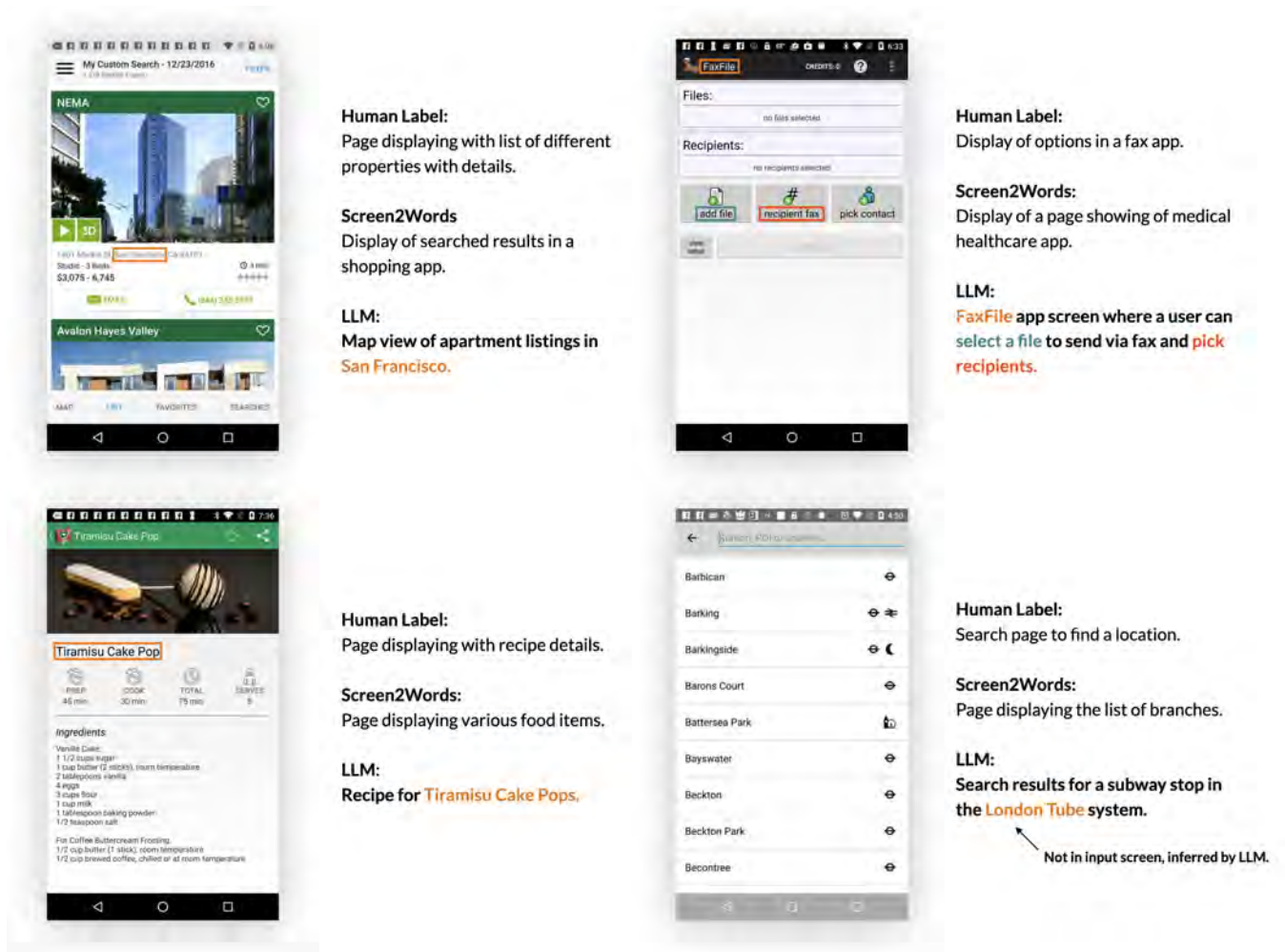


Figure 6: Example summaries generated by prompting the LLM with two exemplars (2-shot learning). The LLM is likelier to use specific texts on the screen to compose summaries (top-left and bottom-right). Moreover, the LLM is more likely to generate more extended summaries that leverage multiple vital elements on the screen (top-right). We also observed that the LLM would use its prior knowledge to help summarize the screens. For example, the bottom-right shows the LLM had inferred the screen is for the London Tube system from the station names displayed on the screen. UI elements relevant to the highlighted phrases in the summaries are called out by bounding boxes with corresponding colors.

of annotations indicated that LLM and Screen2Words generate summaries with equal or very similar accuracy. We further examine the annotator agreement across all individual screens. As shown in Figure 5, LLM summaries are deemed as more accurate for 64.1% of screens according to the majority vote (> 2 votes). Remarkably, LLM summaries were rated as more accurate unanimously by three annotators for 52.8 % of screens. However, our study also showed the LLMs summaries are not always better than the benchmark model, whose summaries received unanimous votes for 20.5% of screens. The results suggest that improving our approach may involve incorporating visual information [3] (as Screen2Words does), as current language models do not have access to this type of information. Therefore, our approach may not perform well for screens without texts and have a strong focus on visuals.

Figure 6 shows example screens with summaries annotated by human labelers and the output from both Screen2Words and the LLM model. We found that the LLM is more likely to use specific texts on the screen to compose summaries, such as San Francisco (top-left) and Tiramisu Cake Pop (bottom-left), while the Screen2Words dataset and the benchmark model output tend to be more generic. Moreover, the LLM is more likely to generate more extended summaries that leverage multiple vital elements on the screen. For example, the top-right screen shows that the LLM composes a longer summary by leveraging the app name, the send file button, and the recipient fax button: "FaxFile app screen where a user can select a file to send via fax and pick recipients." We also observed that the LLM's prior knowledge is beneficial for screen summarization. For example, the bottom-right photo shows

Table 3: The LLM’s performance on the screen QA task.

Model	Exact Matches	Contains GT	Sub-String of GT	Micro-F1
0-shot LLM	30.7%	6.5%	5.6%	31.2%
1-shot LLM	65.8%	10%	7.8%	62.9%
2-shot LLM	66.7%	12.6%	5.2%	64.8%
DistilBERT [47]	36.0%	8.5%	9.9%	37.2%

a station search results page for London’s tube system. The LLM predicts “*Search results for a subway stop in the London tube system.*” However, the input HTML does not contain the words “*London*” nor “*tube*.” Therefore, the model has utilized its prior knowledge about the station names, learned from large language datasets, to infer that they belong to the London Tube. This type of summary may not have been generated if the model is trained only on the Screen2Words dataset and shows the benefit of leveraging LLM’s prior knowledge for summarizing UIs.

5.3 Screen Question-Answering (QA)

5.3.1 Task Formulation. Given a mobile UI and an open-ended question asking for information regarding the UI, the model should provide the correct answer. We focus on factual questions, which require answers based on facts presented on the screen.

5.3.2 Prompt Construction. We use a preamble of “*Given a screen and a question, provide the answer based on the screen information.*” We did not use chain-of-thought prompting as no intermediate result is needed to be generated for the task. In our experiments, N sets of screen HTML and question-answer pairs follow the preamble, where $N = 0, 1, 2$ represents N -shot learning. The output answers are enclosed by special tags <SOA> and <EOA>, meaning the start and end of an answer, respectively. The prompt we used in the study can be found in appendix A.3.

5.3.3 Experiment Setup. We use a dataset of 300 human-labeled question-answer pairs from 121 unique screens in the RICO dataset [12]. It is a preliminary dataset obtained from the authors of an ongoing large-scale data collection for screen question-answering [18]. The data labeling process involves two stages: question annotation and answer annotation. For question annotation, the annotators were asked to frame questions given a screenshot as the context. The annotators were expected to compose questions that only inquire about information that requires no logical reasoning and can be directly read off from the screen. After that, another set of annotators answers the previously annotated questions given the associated screenshots. We randomly held out three screens for prompt construction. The three screens were associated with 12 QA pairs, and we randomly sampled one pair from each screen as exemplars to include in the prompt. In the remaining 288 question-answer pairs, 57 ground truth answers are not present in the view hierarchy data. This is expected because the labeling is based on screenshots instead of view hierarchies, and many screens in the RICO dataset contain inaccurate view hierarchy data [29]. In this work, we focus on the answers that are present in the view hierarchy data, and incorporating screenshot information in the LLM will be a critical direction to investigate in the future.

Since the answers are generated instead of retrieved from screen HTML, some correct answers may not completely match the labels. For example, “2.7.3” versus “*version 2.7.3*”. Therefore, we report performances on four metrics: 1) *Exact Matches*: the predicted answer is identical to the ground truth. 2) *Contains GT*: the answer is longer than the ground truth and fully contains it. 3) *Sub-String of GT*: the answer is a sub-string of the ground truth. 4) *Micro-F1*: the micro F1 scores, calculated based on the number of shared words between the predicted answer and the ground truth across the entire dataset. We consider *Exact Match* answers correct, and those fall within *Contains GT* and *Sub-String of GT* relevant. The three metrics are exclusive to each other, and examples of each can be found in Figure 7-right. We compare the LLM with an off-the-shelf text QA model DistilBERT [47]. We use the distilbert-base-cased-distilled-squad implementation on huggingface.co, which achieved 79.6% Exact Match and 87.0% F1 score on the SQuAD dataset [45]. Unlike existing QA models that extract answers from the input text, the LLM may generate answers in aliases equivalent to the ground truth, for example, 9/15 and September 15th. However, our metrics, which rely on string matching, may not detect these aliases and could result in under-reported scores for our approach.

5.3.4 Results. Table 3 shows the QA results of different settings. Unlike screen summarization, we found that the LLM can already perform screen QA with the zero-shot setting. 30.7% of the generated answers match exactly with the ground truth, 6.5% answers contain the ground truth, and 5.6% answers are sub-strings of the ground truth. The zero-shot performance might be because the training data of LLMs already contain many QA-related data from the internet. Therefore, the model had already learned to perform question-answering. The off-the-shelf DistilBert model achieves 36% Exact Match, 8.5% Contains GT scores, and 9.9% Contains GT scores, slightly better than the zero-shot performance of LLMs. DistilBert model performs much poorly on our tasks compared to standard question-answering benchmarks, which might be because it was not trained with HTML data. Similar to screen summarization, by providing a single exemplar, the performance boosted significantly, achieving 65.8% Exact Match, 10% Contains GT, and 7.8% Sub-String of GT—summing up to 83.6% answers relevant to the ground truth. 1-shot LLM achieved a 62.9% Micro-F1 score, significantly outperforming the baseline’s score of 37.2%. Once again, we observed that the 2-shot setting provided only a modest performance improvement, with relevant answers reaching 84.5% and Micro-F1 achieving 64.8%. Figure 7-left shows example QA results from our experiment using 2-shot learning. The LLM can effectively understand the screen and generate a correct or relevant answer. For the shown screen, the LLM correctly answers Q1, Q2, and Q4. For Q3, the LLM generates an answer containing the ground truth

Input Screen



Screen QA

Q1: What is the name of the author?
 Baseline: Jonathan Neman
 LLM: Jonathan Neman
 Ground Truth: Jonathan Neman

Q2: What is the headline of the article?
 Baseline: <img id=0 class="article loading
 LLM: Sweetgreen will stop accepting cash payments in 2017.
 Ground Truth: Sweetgreen will stop accepting cash payments in 2017.

Q3: Define the posting date of the article?
 Baseline: 2016
 LLM: Dec 23rd, 2016 4:50 am
 Ground Truth: Dec 23rd, 2016

Q4: Who is the CEO of the company?
 Baseline: Jonathan Neman
 LLM: Jonathan Neman
 Ground Truth: Jonathan Neman

Input Screen



Prediction vs. Ground Truth

Exact Match

Q1: What is the email address of support?
 Prediction: support@wdtinc.com
 Ground Truth: support@wdtinc.com

Contains Ground Truth

Q2: What is the help link?
 Prediction: http://www.imapweatherradio.com/help/
 Ground Truth: http://www.imapweatherradio.com/help/

Sub-String of Ground Truth

Q3: What is the version number?
 Prediction: 3.3.5
 Ground Truth: 3.3.5 (137)

Figure 7: Left: Example results from the screen QA experiment. The LLM significantly outperforms the off-the-shelf QA model DistillBert (Baseline). Right: Example of the three metrics used to assess the LLM’s performance on screen QA, including Exact Match, Contains Ground Truth, and Sub-String of Ground Truth. For each example question, the UI element related to its ground truth is highlighted with a bounding box with the corresponding color and question index.

"Dec 23rd, 2016" but includes the time within the day "4:50" am. In contrast, the baseline model trained with standard text question-answering corpus only managed to answer Q4 correctly; for Q3, its answer only contains "2016", a sub-string of the ground truth. Q2 shows that the baseline model sometimes incorrectly retrieves HTML code from the input screen.

5.4 Mapping Instruction to UI Action

5.4.1 Task Formulation. Given a mobile UI screen and natural language instruction to control the UI, the model needs to predict the id of the object to perform the instructed action. For example, when instructed with "Open Gmail," the model should correctly identify the Gmail icon on the home screen. This task is useful for controlling mobile apps using language input such as voice access.

5.4.2 Prompt Construction. We use a preamble of "Given a screen, an instruction, predict the id of the UI element to perform the instruction.". The preamble is followed by N exemplars consisting of the screen HTML, instruction, and ground truth id. Here $N = 0, 1, 2$ representing N -shot learning. We did not use chain-of-thought prompting as no intermediate result is needed to be generated for the task. The output answers are enclosed by special tags <SOI> and <EOI>, meaning the start and end of the predicted element id, respectively. An example prompt can be found in appendix A.4.

5.4.3 Experiment Setup. We use the PixelHelp dataset [37], which contains 187 multi-step instructions for performing everyday tasks on Google Pixel phones, such as switching Wi-Fi settings or checking emails. We randomly sampled one screen from each unique app package in the dataset as prompt modules. We then randomly sampled from the prompt modules to construct the final prompts. We conducted experiments under two conditions: 1) in-app and

Table 4: Mapping Instruction to UI Action Results

Model	Partial	Complete
0-shot LLM	1.29	0.00
1-shot LLM (cross-app)	74.69	31.67
2-shot LLM (cross-app)	75.28	34.44
1-shot LLM (in-app)	78.35	40.00
2-shot LLM (in-app)	80.36	45.00
Seq2Act [37]	89.21	70.59

2) cross-app. In the former, the prompt contains a prompt module from the same app package as the test screen, while in the latter, it does not. Following the original paper, we report the percentage of partial matches and complete matches of target element sequences.

5.4.4 Results. Our experimental results (Table 4) show that the 0-shot setting cannot perform the task with nearly zero partial or complete accuracy. In the cross-app condition, one-shot prompting significantly achieves 74.69 partial and 31.67 complete, meaning 75% of elements associated with the instructions were correctly predicted, and more than 30% tasks are entirely correct. The 2-shot setting offers incremental boosts for both metrics. In the in-app condition, both the 1-shot and 2-shot settings achieve higher scores than their counterparts in the cross-app condition. Our best-performing setting is the 2-shot LLM & in-app, which achieves 80.36 partial and 45.0 complete accuracy scores. Our approach underperforms the benchmark results from the Seq2Act model [37], which was trained on several dedicated datasets with hundreds of thousands of examples. We do not expect prompting LLMs to consistently

outperform benchmarks across all tasks, as the model can only view a handful of data examples and does not update its underlying parameters [7]. Nevertheless, our approach still demonstrated competitive performances for the task using only two examples.

6 DISCUSSIONS AND FUTURE WORK

In this section, we discuss the implications of our investigation, the limitations of our approach, and how future work can build upon our work.

6.1 Implications for Language-Based Interaction

An important takeaway from our studies is that *prototyping novel language interactions on mobile UIs can be as easy as designing a data exemplar*. As a result, an interaction designer can rapidly create functioning mock-ups to test new ideas with end users. Moreover, developers and researchers can explore different possibilities of a target task before investing significant efforts into developing new datasets and models. For example, as discussed in Section 5.2, the summaries in the Screen2Words dataset follow a particular sentence structure, while there are many other ways to summarize a screen. Our approach can allow researchers to quickly inspect various types of summaries and how they work in different scenarios before spending efforts developing dedicated datasets and models.

The conversational interaction enabled by our approach is beneficial for accessibility. It can also be blended with other input/output modalities, such as touch input and screen readers, to offer new possibilities for multi-modal interaction. Another use case of our approach is *end-user prompting*. When a conversational agent fails to understand or carry out the tasks associated with the user's commands, prior work has leveraged programming by demonstration technique that asks users to provide tasks demonstrations to teach the agent [31, 35]. Our work could augment these techniques to allow users to teach the model by prompting. With the assistance of our algorithm, users only need to provide examples of the desired output, such as which button to click given a language command. For more complex prompts, such as ones using Chain-of-Thought prompting, future work can explore using LLMs to assist users in prompt writing.

6.2 Feasibility Experiments

Our approach showed promising results on each task in the feasibility experiments. Notably, our work contributes to the literature as the first to investigate methods to enable *Screen Question-Generation* and *Screen Question-Answering*. Moreover, our approach can generate screen summaries that are more accurate than the benchmark model, according to the human evaluation of the *Screen Summarization* task. Although our method underperformed the benchmark for the *Mapping Instruction to UI Action* task, it is worth noting that our method still achieved competitive accuracy with only two examples instead of extensive modeling on dedicated datasets like the benchmark. One possible reason for the underperformance is that LLMs are trained to generate text instead of selecting (predicting) an element id, and finetuning LLMs can potentially significantly boost the accuracy of this task. As the first work in this direction, our work offers a set of new techniques and findings for using

LLMs to enable conversational interaction. A natural next step is further examining these techniques by integrating them into an actual conversational agent.

6.3 Extending to Multi-Screen, Multi-Turn Conversational Interaction

We focus on unit conversation tasks involving single-turn interactions in this work. Future research can build upon our techniques to enable multi-screen [8] and multi-turn [30] conversational interactions. Multi-screen interactions can allow the agent to carry out complex tasks across various screens. For example, to help a user book a hotel room, an agent could start by asking which hotel the user wants to book and then search for that hotel (Figure 1-1: *Screen Question-Generation*). Once the search results are obtained, the agent could announce a summary of the search result page (Figure 1-2: *Screen Summary*). After that, the user could then ask for specific information from the search results, such as the room rates (Figure 1-3: *Screen Question-Answering*), and finally, the user could command the agent to book the desired room (Figure 1-4: *Mapping Instruction to UI Actions*). Multi-turn interactions can enable the agent to ask for clarifications on ambiguous cases [32] or recover from errors. For example, given the screen in Figure 3 with two emails sent from Google Play, if the user says, "Open the email sent by Google Play", our current approach would likely select one of them directly without clarifying with the users. A sophisticated agent should identify ambiguity and ask the user for clarification, such as, "Do you mean the one sent on Dec. 4th or Nov. 12th?" To achieve multi-screen and multi-turn interactions with the proposed approach, future work will need to investigate methods to store and represent past screens and interactions as contexts [43].

6.4 Shots, Input Lengths, and Model Performance

Our findings indicate that the first prompt exemplar usually significantly boosts LLM performance in mobile UI tasks. In contrast, the inclusion of a second example only results in a marginal improvement. Previous research by Reynolds and McDonnell [46] suggests that the effectiveness of few-shot prompting lies within guiding LLMs to locate specific task locations in the model's existing space of learned tasks. Therefore, the first shot may be the most helpful, and additional examples may provide only marginal benefits in narrowing the model's focus. Relevantly, language models often have input length constraints, which limit the number of exemplars that can be included in the prompt. The length of a screen HTML has a considerable variance, depending on how much information was conveyed through the view hierarchy and the inherent complexity of the screen. Future work could explore approaches to prevent exceeding the input length limit. One potential approach is to dynamically select prompt screens with different lengths. When the test screen HTML is lengthy, shorter prompt screens could be selected. However, it is unclear whether imbalanced lengths between prompt and test screens would lead to inferior performances. Future research could investigate the trade-off between the number of shots, the length of each shot, and their impact on the model's performance in different UI tasks.

As modeling techniques advance, the input length restrictions of language models would likely be lifted [57], allowing for more information to be included in prompts. Another direction for future work is to develop methods for condensing screen HTML into a more concise syntax while still achieving comparable performance. In addition, to enable real-time applications, future research may investigate strategies such as model distillation [16] and model compression [15] for more efficient inference of LLMs.

6.5 Screen Representation

Mobile UI screens contain multiple modalities, including pixels, texts, and even audio when media content is present. A limitation of our investigation is that we only use the view hierarchy information, which is converted to an HTML representation, and leave other modalities unused. This limitation is imposed by the type of input expected by LLMs. While our studies showed LLMs could perform decently on various UI tasks, they could fail in cases requiring information not present in the view hierarchy but available as pixels. For instance, many icons or images on UI screens have missing captions or alt texts (text description of a visual element), and LLMs may not be able to perform tasks based on these elements. Moreover, visual information is particularly crucial for some apps, e.g., photo editing tools, and our approach may fall short of enabling conversational interaction based on these apps. Many models have started using multiple modalities, including visual and text information of UIs [8, 38, 50, 58]. Future work could exploit these prior models to generate missing captions or alt-text of elements, which can lead to more comprehensive screen information in the HTML input to LLMs. Our approach can also be extended by leveraging large-scale vision language models such as Flamingo [4] to encode a screen's visual and structural information for few-shot learning.

6.6 Steerability and Reliability of LLM Predictions

Our experiments uncovered several promising capabilities of LLMs for mobile UI tasks. For example, the model showcased the behavior of question combination, i.e., merging relevant input fields into a single question, in the *Screen Question-Generation* task. Furthermore, our study shows that the model can utilize its embedded prior knowledge to provide additional information in the *Screen Summarization* task. While these capabilities are intriguing and potentially useful, there is currently a lack of direct controls for steering LLMs in terms of when and how these behaviors should occur. Additionally, LLMs, as other natural language generation models, could sometimes produce unintended text, known as hallucinations [19]. This means that the model may produce incorrect or irrelevant information. Our current method does not explicitly address this issue. As the research surrounding LLMs develops, it is crucial to find ways to improve the steerability and reliability of their predictions and behaviors, especially when incorporating LLMs into user-facing technology.

6.7 Generalizing Beyond Mobile UIs

This paper focuses on mobile UIs, but the proposed approach with LLMs can also be applied to other UI types, such as web UIs [1] (which are already in HTML syntax) and popular UI systems like

iOS and macOS that possess view hierarchy data or equivalence. However, for more sophisticated and feature-rich UIs like video editors, a direct adaptation of our approach may be challenging due to the difficulty in representing key features in text format and ensuring the representation complies with model length constraints, as highlighted in the previous discussions. Nonetheless, our research provides a solid foundation for further studies to build upon and develop innovative methods that enable language interaction with graphical user interfaces.

7 CONCLUSION

We investigated the feasibility of prompting LLMs to enable various conversational interactions on mobile UIs. By categorizing the conversation scenarios between users and agents during mobile tasks, we identified four crucial UI tasks to study. We proposed a suite of prompting techniques for adapting LLMs to mobile UIs. We conducted extensive experiments with the four selected tasks to evaluate the effectiveness of our approach. The results showed that compared to traditional machine learning pipelines that consist of expensive data collection and model training, one could rapidly realize novel language-based interactions using LLMs while achieving competitive performance.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback during the R&R process that enhanced the paper's quality. We appreciate the discussions and feedback from our team members Chin-Yi Cheng and Tao Li. We also acknowledge the early-stage feedback from Michael Terry and Minsuk Chang. Special thanks to the Google Data Compute team for their invaluable assistance in data collection.

REFERENCES

- [1] Adept. 2022. *ACT-1: Transformer for Actions*. <https://www.adept.ai/act>
- [2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. <https://doi.org/10.48550/ARXIV.2204.01691>
- [3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. <https://doi.org/10.48550/ARXIV.2204.14198>
- [4] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. <https://doi.org/10.48550/ARXIV.2204.14198>
- [5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*. 2425–2433.

- [6] Richard A Bolt. 1980. “Put-that-there” Voice and gesture at the graphics interface. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*. 262–270.
- [7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *arXiv:2005.14165* [cs.CL]
- [8] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. 2022. A Dataset for Interactive Vision-Language Navigation with Unknown Command Feasibility. <https://doi.org/10.48550/ARXIV.2202.02312>
- [9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).
- [10] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. 2022. TaleBrush: Visual Sketching of Story Generation with Pretrained Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI EA '22). Association for Computing Machinery, New York, NY, USA, Article 172, 4 pages. <https://doi.org/10.1145/3491102.3519873>
- [11] 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. <https://doi.org/10.48550/ARXIV.2209.01390>
- [12] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschan, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual Symposium on User Interface Software and Technology (UIST '17)*.
- [13] Asbjørn Følstad and Petter Bae Brandtæg. 2017. Chatbots and the new world of HCI. *interactions* 24, 4 (2017), 38–42.
- [14] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News Summarization and Evaluation in the Era of GPT-3. <https://doi.org/10.48550/ARXIV.2209.12356>
- [15] Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. <https://doi.org/10.48550/ARXIV.1510.00149>
- [16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. <https://doi.org/10.48550/ARXIV.1503.02531>
- [17] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 159–166.
- [18] Yu-Chung Hsiao, Fedir Zubach, Maria Wang, and Chen Jindong. 2022. ScreenQA: Large-Scale Question-Answer Pairs over Mobile App Screenshots. <https://doi.org/10.48550/ARXIV.2209.08199>
- [19] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* (nov 2022). <https://doi.org/10.1145/3571730> Just Accepted.
- [20] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-Based Prototyping with Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI EA '22). Association for Computing Machinery, New York, NY, USA, Article 35, 8 pages. <https://doi.org/10.1145/3491102.3503564>
- [21] Ellen Jiang, Edwin Toh, Alejandra Molina, Kristen Olson, Claire Kayacik, Aaron Donsbach, Carrie J Cai, and Michael Terry. 2022. Discovering the Syntax and Strategies of Natural Language Programming with Generative Language Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 386, 19 pages. <https://doi.org/10.1145/3491102.3501870>
- [22] Clare-Marie Karat, John Vergo, and David Nahamoo. 2002. Conversational interface technologies. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*. 169–186.
- [23] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 5, 17 pages. <https://doi.org/10.1145/3491102.3501931>
- [24] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. <https://doi.org/10.48550/ARXIV.2205.11916>
- [25] 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* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 388, 19 pages. <https://doi.org/10.1145/3491102.3502030>
- [26] Yoonjoo Lee, John Joon Young Chung, Tae Soo Kim, Jean Y Song, and Juho Kim. 2022. Promptverse: Scalable Generation of Scaffolding Prompts Through Human-AI Hybrid Knowledge Graph Annotation. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 96, 18 pages. <https://doi.org/10.1145/3491102.3502087>
- [27] Yoonjoo Lee, Tae Soo Kim, Minsuk Chang, and Juho Kim. 2022. Interactive Children’s Story Rewriting Through Parent-Children Interaction. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*. 62–71.
- [28] Luis A. Leiva, Asutosh Hota, and Antti Oulasvirta. 2022. Describing UI Screenshots in Natural Language. *ACM Trans. Intell. Syst. Technol.* 14, 1, Article 19 (nov 2022), 28 pages. <https://doi.org/10.1145/3564702>
- [29] Gang Li, Gilles Baechler, Manuel Tragut, and Yang Li. 2022. Learning to Denoise Raw Mobile UI Layouts for Improving Datasets at Scale. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 67, 13 pages. <https://doi.org/10.1145/3491102.3502042>
- [30] Tao Li, Gang Li, Jingjie Zheng, Purple Wang, and Yang Li. 2022. MUG: Interactive Multimodal Grounding on User Interfaces. <https://doi.org/10.48550/ARXIV.2209.15099>
- [31] Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. 2017. SUGILITE: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 6038–6049.
- [32] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M Mitchell, and Brad A Myers. 2020. Multi-modal repairs of conversational breakdowns in task-oriented dialogs. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1094–1107.
- [33] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M. Mitchell, and Brad A. Myers. 2018. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Natural Language Instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 105–114. <https://doi.org/10.1109/VLHCC.2018.8506506>
- [34] Toby Jia-Jun Li, Lindsay Popowski, Tom Mitchell, and Brad A Myers. 2021. Screen2vec: Semantic embedding of gui screens and gui components. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.
- [35] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent That Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) (UIST '19). Association for Computing Machinery, New York, NY, USA, 577–589. <https://doi.org/10.1145/3332165.3347899>
- [36] Toby Jia-Jun Li and Oriana Riva. 2018. Kite: Building Conversational Bots from Mobile Apps. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services* (Munich, Germany) (MobiSys '18). Association for Computing Machinery, New York, NY, USA, 96–109. <https://doi.org/10.1145/3210240.3210339>
- [37] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. Mapping Natural Language Instructions to Mobile UI Action Sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 8198–8210. <https://doi.org/10.18653/v1/2020.acl-main.729>
- [38] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei (Wei) Guan. 2020. Widget Captioning: Generating Natural Language Description for Mobile User Interface Elements. <https://www.aclweb.org/anthology/2020.emnlp-main.443.pdf>
- [39] Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021. VUT: Versatile UI Transformer for Multi-Modal Multi-Task User Interface Modeling. *arXiv preprint arXiv:2112.05692* (2021).
- [40] Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021. VUT: Versatile UI Transformer for Multi-Modal Multi-Task User Interface Modeling. <https://doi.org/10.48550/ARXIV.2112.05692>
- [41] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement Learning on Web Interfaces Using Workflow-Guided Exploration. *arXiv:1802.08802* [cs.AI]
- [42] Yihe Liu, Anushk Mittal, Diyi Yang, and Amy Bruckman. 2022. Will AI Console Me When I Lose My Pet? Understanding Perceptions of AI-Mediated Email Writing. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 474, 13 pages. <https://doi.org/10.1145/3491102.3517731>
- [43] OpenAI. 2022. CHATGPT: Optimizing language models for dialogue. <https://openai.com/blog/chatgpt/>
- [44] Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. 2018. Mapping natural language commands to web elements. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 4970–4976. <https://doi.org/10.18653/v1/D18-1145>

- //doi.org/10.18653/v1/D18-1540
- [45] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).
- [46] Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. <https://doi.org/10.48550/ARXIV.2102.07350>
- [47] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. <https://doi.org/10.48550/ARXIV.1910.01108>
- [48] Zhanna Sarsenbayeva, Niels van Berkel, Chu Luo, Vassilis Kostakos, and Jorge Goncalves. 2017. Challenges of situational impairments during interaction with mobile devices. In *Proceedings of the 29th Australian Conference on Computer-Human Interaction*. 477–481.
- [49] Kashyap Todi, Luis A. Leiva, Daniel Buschek, Pin Tian, and Antti Oulasvirta. 2021. Conversations with GUIs. In *Proceedings of the ACM SIGCHI Conference on Designing Interactive Systems (DIS '21)*. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/3461778.3462124>
- [50] Bryan Wang, Gang Li, Xin Zhou, Zhouong Chen, Tovi Grossman, and Yang Li. 2021. Screen2Words: Automatic Mobile UI Summarization with Multimodal Learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (UIST '21). Association for Computing Machinery, New York, NY, USA, 498–510. <https://doi.org/10.1145/3472749.3474765>
- [51] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-Augmented Ensembles in Language Models. <https://doi.org/10.48550/ARXIV.2207.00747>
- [52] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).
- [53] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. <https://doi.org/10.48550/ARXIV.2201.11903>
- [54] Jacob O Wobbrock. 2019. Situationally aware mobile devices for overcoming situational impairments. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 1–18.
- [55] Jason Wu, Xiaoyi Zhang, Jeff Nichols, and Jeffrey P Bigham. 2021. Screen Parsing: Towards Reverse Engineering of UI Models from Screenshots. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (UIST '21). Association for Computing Machinery, New York, NY, USA, 470–483. <https://doi.org/10.1145/3472749.3474763>
- [56] Tongshuang Wu, Michael Terry, and Carrie Jun 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* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 385, 22 pages. <https://doi.org/10.1145/3491102.3517582>
- [57] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. (2020). <https://doi.org/10.48550/ARXIV.2007.14062>
- [58] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 275, 15 pages. <https://doi.org/10.1145/3411764.3445186>
- [59] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. <https://doi.org/10.48550/ARXIV.2205.10625>

A EXAMPLE PROMPTS FOR FEASIBILITY EXPERIMENTS TASKS

We show the prompt examples used in each task from the feasibility experiments. For screen summarization, screen QA, and mapping instruction to UI action, we include 1-shot prompts for simplicity. For the screen question-generation study, we include the 2-shot prompt—one exemplar uses question combination (the second screen), and the other does not. Both exemplars contain a chain of thoughts consisting of intermediate results.

A.1 Screen Question-Generation

```

1 Given a screen, the agent needs to identify the elements
  requiring user input and generates corresponding
  questions.
2
3 Screen:
4 <p id=0 class="alertTitle"> Create password </p>
5 <div id=1 class="titleDivider"> </div>
6 <input id=2 class="password"> Crowd3116 </input>
7 <input id=3 class="confirm password"> Crowd3116 </input
  >
8 <input id=4 class="hint"> c3 </input>
9 <input id=5 class="edEmailAddress"> appcrawler4@gmail.
  com </input>
10 <p id=6 class="tvEmailAddressInfo"> This email address
   will be used to reset your password. </p>
11 <button id=7 class="button2"> Cancel </button>
12 <button id=8 class="button1"> OK </button>
13
14 Now reasoning starts:
15 Q: How many input tags are there on the screen?
16 A: 4
17 Q: What is the purpose of the screen?
18 A: Create password.
19
20 It's a create password page and there are 4 input tags,
  including:
21 1. id=2 asks for password.
22 2. id=3 asks to confirm password.
23 3. id=4 asks for hint.
24 4. id=5 asks for email address.
25
26 To help the user proceed with the screen, an agent will
  ask:
27 <SQ>What password do you want to create? (id=2)<EQ>
28 <SQ>Could you enter the password again to confirm? (id
  =3)<EQ>
29 <SQ>What hint do you want to set? (id=4)<EQ>
30 <SQ>What is your email in case you need to reset the
  password? (id=5)<EQ>
31
32 Screen:
33 <p id=0> IRS2Go, </p>
34 <button id=1 alt="Open navigation drawer"> </button>
35 <p id=2 class="titleRefund"> Refund Status </p>
36 <p id=3 class="refundHeaderText"> Check your refund
   status by entering your information as shown on your
   2015 tax return. This tool is updated no more than
   once every 24 hours, usually overnight. </p>
37 <input id=4 class="taxId3Edit" alt="First 3 Digits of
   Social Security Number"> </input>
38 <p id=5 class="dash1"> - </p>
39 <input id=6 class="taxId2Edit" alt="Middle 2 Digits of
   Social Security Number"> </input>
40 <p id=7 class="dash2"> - </p>
41 <input id=8 class="taxId4Edit" alt="Last 4 Digits of
   Social Security Number"> </input>
42 <p id=9> Filing Status </p>
43 <input id=10 class="refundAmountEdit"> </input>
44 <button id=11 class="privacyNoticeButton" alt="Privacy
   Notice"> Privacy Notice, </button>
45 <button id=12 class="getStatusButton" alt="Get Status">
   GET STATUS, </button>
46 <div id=13 class="navigationBarBackground"> </div>
47 <div id=14 class="statusBarBackground"> </div>
48
49 Now reasoning starts:
50 Q: How many input tags are there on the screen?

```

```

51 A: 4
52 Q: What is the purpose of the screen?
53 A: Check your refund status.
54
55 It's a check refund status page and there are 4 input
    tags, including:
56 1. id=4 asks for first 3 digits of SSN
57 2. id=6 asks for middle 2 digits of SSN
58 3. id=8 asks for last 4 digits of SSN
59 4. id=10 asks for the amount of refund.
60
61 To help the user proceed with the screen, an agent will
    ask:
62 <SOQ>What is your SSN? (id=4, id=6, id=8)<EOQ>
63 <SOQ>What is the refund amount? (id=10)<EOQ>
64 ""

```

Listing 1: 2-shot example prompt for screen question-generation.

A.2 Screen Summarization

```

1 Given a screen, summarize its purpose.
2
3 Screen:
4 <img id=0> </img>
5 <p id=1 class="cliv name textview"> Create new contact <
    /p>
6 <img id=2> </img>
7 <p id=3 class="cliv name textview"> Add to a contact </p>
8 <img id=4> </img>
9 <p id=5 class="cliv name textview"> Send SMS </p>
10 <button id=6 class="floating action button" alt="dial pad
    "> </button>
11 <button id=7 class="search back button" alt="stop
    searching"> </button>
12 <input id=8 class="search view"> 18773312998 </input>
13 <img id=9 class="search close button" alt="Clear search">
    </img>
14 <div id=10 class="navigationBarBackground"> </div>
15 <div id=11 class="statusBarBackground"> </div>
16
17 Summary: <SOS>Screen of contact settings options<EOS>

```

Listing 2: 1-shot example prompt for screen summarization.

A.3 Screen Question-Answering

```

1 Given a mobile screen and a question, provide the answer
    based on the screen information.
2
3 Screen:
4 <p id=0> Invite for T20 Fans Live Chat </p>
5 <button id=1 alt="Choose account"> </button>
6 <p id=2 class="menu send" alt="Send"> </p>
7 <p id=3 class="message header"> Message </p>
8 <input id=4 class="message"> Join me on T20 Fans Live
    chat. </input>
9 <div id=5 class="message separator"> </div>
10 <p id=6 class="message limit"> </p>
11 <div id=7 class="separator"> </div>
12 <p id=8 class="selection"> Add recipients </p>
13 <div id=9 class="separator"> </div>
14 <p id=10 class="text"> Suggestions from Google </p>
15 <p id=11> A, </p>
16 <p id=12 class="name"> appcrawler5@gmail.com </p>
17 <p id=13 class="contact detail"> appcrawler5@gmail.com <
    /p>

```

```

18 <img id=14 class="contact method"> </img>
19 <div id=15 class="divider"> </div>
20 <p id=16> A, </p>
21 <p id=17 class="name"> appcrawler4@gmail.com </p>
22 <p id=18 class="contact detail"> appcrawler4@gmail.com <
    /p>
23 <img id=19 class="contact method"> </img>
24 <div id=20 class="divider"> </div>
25 <p id=21 class="text"> Everyone </p>
26 <img id=22> </img>
27 <p id=23 class="name"> App Crawler </p>
28 <p id=24 class="contact detail"> (415) 336-5454 </p>
29 <img id=25 class="contact method"> </img>
30 <img id=26 class="channel switcher icon"> </img>
31 <div id=27 class="divider"> </div>
32 <p id=28> T, </p>
33 <p id=29 class="name"> test, </p>
34 <p id=30 class="contact detail"> (415) 336-5454 </p>
35 <img id=31 class="contact method"> </img>
36 <img id=32 class="channel switcher icon"> </img>
37 <div id=33 class="divider"> </div>
38 <div id=34 class="navigationBarBackground"> </div>
39 <div id=35 class="statusBarBackground"> </div>
40
41 Q: What email addresses are there?
42 A: <SOA>appcrawler5@gmail.com<EOA>

```

Listing 3: 1-shot example prompt for screen question-answering

A.4 Mapping Instruction to UI Action

```

1 Given a screen, an instruction, predict the id of the UI
    element to perform the instruction.
2
3 Screen:
4 <div id=0 alt="Apps list"> </div>
5 <img id=1 class="g icon"> </img>
6 <img id=2 class="mic icon" alt="Voice search"> </img>
7 <p id=3 class="icon" alt="Calculator"> Calculator </p>
8 <p id=4 class="icon" alt="Calendar"> Calendar </p>
9 <p id=5 class="icon" alt="Camera"> Camera </p>
10 <p id=6 class="icon" alt="Chrome"> Chrome </p>
11 <p id=7 class="icon" alt="Clock"> Clock </p>
12 <p id=8 class="icon" alt="Contacts"> Contacts </p>
13 <p id=9 class="icon" alt="Custom Locale"> Custom Locale
    </p>
14 <p id=10 class="icon" alt="Dev Tools"> Dev Tools </p>
15 <p id=11 class="icon" alt="Drive"> Drive </p>
16 <p id=12 class="icon" alt="Files"> Files </p>
17 <p id=13 class="icon" alt="Gmail"> Gmail </p>
18 <p id=14 class="icon" alt="Google"> Google </p>
19 <p id=15 class="icon" alt="Hangouts"> Hangouts </p>
20 <p id=16 class="icon" alt="Maps"> Maps </p>
21 <p id=17 class="icon" alt="Messages"> Messages </p>
22 <p id=18 class="icon" alt="Phone"> Phone </p>
23 <p id=19 class="icon" alt="Photos"> Photos </p>
24 <p id=20 class="icon" alt="Play Movies & TV"> Play Movies
    & TV </p>
25 <p id=21 class="icon" alt="Play Music"> Play Music </p>
26 <p id=22 class="icon" alt="Settings"> Settings </p>
27 <p id=23 class="icon" alt="WebView Browser Tester">
    WebView Browser Tester </p>
28 <p id=24 class="icon" alt="YouTube"> YouTube </p>
29 <p id=25 class="icon" alt="Photos"> Photos </p>
30 <p id=26 class="icon" alt="Maps"> Maps </p>
31 <p id=27 class="icon" alt="Contacts"> Contacts </p>
32 <p id=28 class="icon" alt="Settings"> Settings </p>
33 <p id=29 class="icon" alt="Clock"> Clock </p>

```

```
34 <div id=30 class="fast scroller"> </div>
35 <div id=31> </div>
36 <div id=32 class="hotseat"> </div>
37
38 Instruction: Open your device's Clock app.
```

```
39 Prediction: id=<S0I>29<E0I>
```

Listing 4: 1-shot example prompt for mapping instruction to UI action.