"What's important here?": Opportunities and Challenges of Using LLMs in Retrieving Information from Web Interfaces

Faria Huq, Jeff Bigham, Nikolas Martelaro
School of Computer Science
Carnegie Mellon University

Abstract

Large language models (LLMs) that have been trained on large corpus of codes exhibit a remarkable ability to understand HTML code [1]. As web interfaces are mainly constructed using HTML, we design an in-depth study to see how the code understanding ability of LLMs can be used to retrieve and locate important elements for a user given query (*i.e.* task description) in web interface. In contrast with prior works, which primarily focused on autonomous web navigation, we decompose the problem as an even atomic operation - *Can LLMs find out the important information in the web page for a user given query?* This decomposition enables us to scrutinize the current capabilities of LLMs and uncover the opportunities and challenges they present. Our empirical experiments show that while LLMs exhibit a reasonable level of performance in retrieving important UI elements, there is still a substantial room for improvement. We hope our investigation will inspire follow-up works in overcoming the current challenges in this domain.

1 Introduction

Web assistants capable of understanding natural language command and retrieving relevant information from Web User Interface (UI) could significantly enhance human efficiency. With the recent advancement of LLMs and their potential in understanding raw HTML, research attempts have predominantly leaned on LLM-driven web assistants capable of autonomous navigation [2, 3, 4, 1, 5]. However, as demonstrated by [3] and [4], these assistants can not yet exceed 11% end-to-end success rate for real-world websites. Due to the lower accuracy, these autonomous agents are yet to be a practical solution for day-to-day usage.

Please note that, to successfully complete a user given task that requires navigating through a series of web-pages, an autonomous agent must be able to perform a more fundamental operation – retrieving the most important UI elements from each page. Motivated by this insight, in this paper, we aim to explore LLMs' cabability to perform this fundamental operation. We conduct a comprehensive investigation into four key components of input prompts: example selection for few-shot prompting, specificity of natural language command, HTML truncation strategy, and the role assumed by the LLM (*persona*). We find that the performance of LLM is intertwined with these components. Carefully prompting the few-shot examples can help LLMs to succeed as long as the input sequence size is reasonable. For instance– using lexico-semantically similar few-shot examples can boost the recall by 9.70% in 1-shot prompting; however, it decreases the performance by 13.17% in 2-shot prompting. We also show that an effective way to truncate the HTML document can alone lead to better performance with gains upto 11.54%. We unveil critical limitations of LLMs such as hallucination and failure to follow instruction in terms of UI element retrieval. We conclude by providing future direction and possible solutions to the limitations we observed in our study.

37th R0-FoMo: Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models at NeurIPS 2023(NeurIPS 2023).

2 Related Works

2.1 LLM for UI

LLM has recently gained popularity for many aspects of UI tasks. [2] shows an in-depth study on LLM for interacting with mobile UI — ranging from task automation to screen summarization. [6] performs an offline exploration and creates a transition graph, which is used to provide more contextual information to the LLM prompt. [7] introduces chain-of-action prompting that leverages previous action history and future action plans to decide the next action. These works primarily focus on Mobile UI – which has a significantly smaller search space than Web UI. Compared to Mobile UI, Web UI is much more complex and has more elements on each page – making it harder to process and incorporate with LLMs [3, 4]. Most of the early works on Web UI are based on synthetic frameworks, MiniWob [5] and WebShop [8]. [1] proposed a fine-tuned T5 model on WebShop. However, these datasets are not well representative of real-world web activities. To capture the complexity of real world tasks, [4] and [3] introduce two realistic environments and datasets encompassing real-world tasks. However, to the best of our knowledge, there has been no in-depth analysis of the true capabilities of LLMs in the context of Web UI. Specifically, there is a gap in understanding how to effectively prompt LLMs for Web UI tasks, which prompting strategies yield favorable results and the underlying reasons behind their success or failure. In this study, we explore the potential of LLMs for Web UI information retrieval through evaluating the effectiveness of state-of-the-art prompting techniques.

2.2 Prompting LLM for Instruction following

LLMs are now largely used for many tasks related to instruction following [9, 10, 11, 12, 13, 14, 15]. Due to the large pool of works active in LLM, we only provide a high-level overview in this section. Especially for Vision-and-Language Navigation, LLM has been successful as a co-ordinator and/or planner [16, 17, 18, 19]. Some of the most recent works proposed new frameworks for holistic evaluation of LLMs [20, 21, 22, 23, 24]. However, these frameworks are mostly for analytic problem-solving for math and/or reasoning and may not be a proper reflection of the unique challenges encountered in tasks related to user interfaces. This study specifically focuses on exploiting some of these prompt techniques for Web UI and highlighting their feasibility in this domain.

3 Experiments

In our experiments, we use the `Claude2` model [1] by Anthropic [25]. `Claude2` has a context length of 100k tokens, which is the largest among all the LLMs to date. Having a large context window is especially beneficial for Web UI analysis considering the potentially thousands of elements on a webpage [4, 1].

3.1 Problem Formulation

We define each example as a set "$\{\{w, q\}, e\}$", where w, q, and e represent `Current HTML`, `user-query`, and `ground truth UI element`, respectively. A system, Ψ, has to retrieve the most important UI element as,

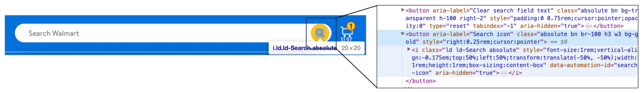
$$e = \Psi(w, q), \tag{1}$$

where $\hat{e}$ is the retrieved UI element, which is compared with e to compute the performance of Ψ.

3.2 Dataset

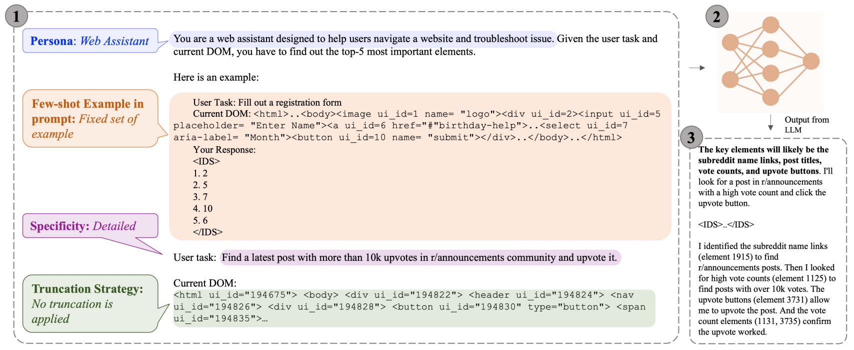
We use Mind2Web [4] dataset for our experiments. Mind2Web has 2,000 open-ended and real-world tasks collected from 137 websites – making it particularly suitable for our experiment. The dataset contains three different test sets, namely– 1) Cross-Task: examples from unseen tasks during training; 2) Cross-Website: examples from unseen websites; 3) Cross-Domain: examples from unseen domains. In our setting, these three variants of test set are especially helpful for understanding the

[1]For each experiment, we use a fixed value of temperature = 1. We use a high temperature to encourage exploration following [3].



```
<button aria-label="Clear search field text" class="absolute bn bg-tr
ansparent h-100 right-2" style="padding:0 0.75rem;cursor:pointer;opac
ity:0" type="reset" tabindex="-1" aria-hidden="true">…</button>
<button aria-label="Search icon" class="absolute bn br-100 h3 w3 bg-g
old" style="right:0.25rem;cursor:pointer"> == $0
  <i class="ld ld-Search absolute" style="font-size:1rem;vertical-ali
  gn:-0.175em;top:50%;left:50%;transform:translate(-50%, -50%);width:
  1rem;height:1rem;box-sizing:content-box" data-automation-id="search
  -icon" aria-hidden="true">…</i>
</button>
```

Figure 1: Example of a nested DOM layout where predicting either of parent and child can achieve identical result.


1
Persona: Web Assistant
Few-shot Example in prompt: Fixed set of example
Specificity: Detailed
Truncation Strategy: No truncation is applied
2
Output from LLM
3

You are a web assistant designed to help users navigate a website and troubleshoot issue. Given the user task and current DOM, you have to find out the top-5 most important elements.

Here is an example:

User Task: Fill out a registration form
Current DOM: `<html>..<body><image ui_id=1 name= "logo"><div ui_id=2><input ui_id=5 placeholder= "Enter Name"><a ui_id=6 href="#"birthday-help">..<select ui_id=7 aria-label= "Month"><button ui_id=10 name= "submit"></div>..</body>..</html>`
Your Response:
<IDS>
1. 2
2. 5
3. 7
4. 10
5. 6
</IDS>

User task: Find a latest post with more than 10k upvotes in r/announcements community and upvote it.

Current DOM:
```
<html ui_id="194675"> <body> <div ui_id="194822"> <header ui_id="194824"> <nav
ui_id="194826"> <div ui_id="194828"> <button ui_id="194830" type="button"> <span
ui_id="194835">…
```

The key elements will likely be the subreddit name links, post titles, vote counts, and upvote buttons. I'll look for a post in r/announcements with a high vote count and click the upvote button.

<IDS>..</IDS>

I identified the subreddit name links (element 1915) to find r/announcements posts. Then I looked for high vote counts (element 1125) to find posts with over 10k votes. The upvote buttons (element 3731) allow me to upvote the post. And the vote count elements (1131, 3735) confirm the upvote worked.

Figure 2: Overview of our experiment setup. (1) Input prompt is crafted based on four components, namely user role (*i.e: persona*), few-shot examples, specificity and HTML truncation policy; (2) LLM (`Claude2` in this case) processes the given prompt; and (3) LLM generates the corresponding output.

generalizability of LLMs. We refer viewers to the original paper for more information about the dataset [4]. Please note that in our study, we use the "`Target Element`" provided in the dataset as e in Equation 1, since they are analogous. Similar to the original setting of Mind2web, we also take the rate limit of `Claude2` API into account and use a subset with 124 examples for our experiments.

3.3 Evaluation Metric

HTML layouts are often complicated and nested in a manner where multiple elements can point to the same information. For example, Figure 1 shows an example from `Walmart.com` where the search button has a single child element (the search icon). Predicting either the search button or the icon would yield identical results. To address this, we take the following approach – Given a UI element as the prediction, we expand the tree to get the *leaf nodes* under it and compare them with the same from our ground-truth labels. We report the evaluation in terms of *recall* (Eq. 2) and *element accuracy*. Similar to [3] and [26], we also emphasize using *recall* rather than *precision* as we have access to only one positively labeled UI element.

$$\text{Recall} = \frac{|\text{Predicted\_Elements} \cap \text{Ground\_Truth}|}{|\text{Ground\_Truths}|} \tag{2}$$

Element accuracy compares the predicted elements with all acceptable elements in the same manner as Mind2Web [4]. Please note that the other metrics (Success Rate, Step Success Rate, Operation F1) used in Mind2Web are not well-suited to our context – as we are not focusing on web automation.

3.4 Investigation on Different Components of a Prompt

In this section, we discuss how we probe LLM for each specific component and report the findings for each condition (denoted by 💡 icon).

3.4.1 Impact of Example Selection for In-Context Learning

In-context learning (ICL) refers to a particular approach to prompt engineering in which the model is given a few examples of the task as a component of the prompt [27, 15]. [28] showed that the choice of prompt examples can impact the effectiveness of ICL, and thus, the performance of LLMs. If the few-shot examples in the prompt are semantically close to the target sentence, they noticed the performance improved compared to passing a set of constant examples. Inspired by their findings, we also investigate the impact of example selection in the few-shot prompting in our task. To find relevant examples, we use the k-nearest neighbor search on the train set of Mind2Web, Specifically,

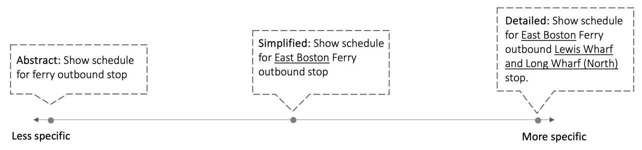

Abstract: Show schedule for ferry outbound stop
Simplified: Show schedule for East Boston Ferry outbound stop
Detailed: Show schedule for East Boston Ferry outbound Lewis Wharf and Long Wharf (North) stop.
Less specific
More specific

Figure 3: Example of a task description based on its level of specificity

given the training examples S and the target sentence t, we aim to find a subset $\mathcal{N}_{s,t} \subset \mathcal{S}$ with k examples that are the closest to t,

$$\mathcal{N}^*_{s,t} = \underset{\mathcal{N}_{s,t} \subset \mathcal{S}:|\mathcal{N}_{s,t}|=k}{\arg\min} \sum_{e \in \mathcal{N}_{s,t}} \sum_{i=1}^{|\mathrm{Emb}(e)|} (\mathrm{Emb}(e)_i - \mathrm{Emb}(t)_i)^2 \tag{3}$$

where $\mathrm{Emb}(\cdot) \in \mathbb{R}^{384}$ is the embedding space of MiniLMv2 [29], and $\mathcal{N}^*_{s,t}$ is the most optimal subset.

Table 1 reports the performance of `Claude2` based on the choice of examples. 💡 Providing the kNN examples from the train set significantly helps in the Cross-Task test set. This might be due to the presence of similar tasks in the train set that boosts the performance [28]. However, for Cross-Website and Cross-Domain splits, the kNN examples did not help much. We hypothesize this might be due to the unfamiliarity of website and domain between the train and test sets. We also noticed that for the kNN example selection, the 1-shot prompt yielded better performance than 2-shot prompt. One probable reason might be the increased length of input sequence causing a decay in performance, as also demonstrated by [30]. Our experiment to find the impact of HTML length on the LLM's performance support this possibility as well (see Section A.5 in appendix). However, for fixed prompt examples, the accuracy consistently improved throughout the three test sets when we used 2-shot prompting.[2] We can thus understand that providing semantically relevant examples in the prompt is not enough for Web UI, we also need to be mindful of the context length.

Model	Prompt Example	Prompt Setup	Cross-Task		Cross-Website		Cross-Domain	
			Recall	**Ele. Acc**	**Recall**	**Ele. Acc**	**Recall**	**Ele. Acc**
`Claude 2`	Fixed	1-shot	43.243%	36.364%	*40.909%*	**40.909%**	11.429%	17.647%
		2-shot	**58.621%**	**50.0%**	**48.235%**	30%	**51.515%**	27.273%
	kNN (mind2web)	1-shot	*52.941%*	40%	36.364%	*38.235%*	25.806%	**29.412%**
		2-shot	45.455%	*42.105%*	12.698%	23.529%	13.725%	26.667%

Table 1: Comparison of Example selection strategy for few-shot prompting on Mind2Web Test sets.

3.4.2 Impact of Specificity in User Query

Recent studies have shown that generic users find it difficult to prompt effectively and often opt for *abstract* or *higher-level* description while interacting with LLMs [31]. However, most of the publicly available UI datasets including Mind2Web include very detailed task descriptions which can be unrealistic and unnatural. To explore the effect of specificity in LLM's performance, we augment Mind2Web task description in three ways - 1) *Detailed*: the original task description from the test set; 2) *Simplified*: simplified task description which may still include essential details 3) *Abstract*: High-level task description which does not include any details to mimic real-world queries by users [32, 31]. Figure 3 shows an example from the Mind2web dataset and its three variants of description. We use GPT-4 to construct the corresponding simplified and abstract descriptions. See appendix (section A.3) for more details regarding the description creation process. To the best of our knowledge, this is the first work to investigate the impact of specificity for UI related task for LLM.

Table 2 reports the performance of `Claude2` based on the level of specificity in user query. 💡 For kNN based prompt example, the performance gradually decayed from detailed to abstract specificity across three test sets. However, when the examples were fixed, the result was relatively consistent throughout all levels of specificity. We hypothesize that this is due to the particular choice of the fixed examples. The task descriptions of the fixed examples are more abstract while the examples from Mind2Web train set are more detailed. This abstract task description in the fixed examples perhaps helped `Claude2` to generalize better across the three levels. This also highlights the sensitivity of LLMs towards the design of few-shot examples.

[2]The few-shot examples chosen in the experiment (referred as *Fixed*) are shown in Figure 5.

Model	Prompt Example	Level of Specificity	Prompt Setup	Cross-Task		Cross-Website		Cross-Domain	
				Recall	**Ele. Acc**	**Recall**	**Ele. Acc**	**Recall**	**Ele. Acc**
Claude2	Fixed	Detailed	1-shot	43.243%	36.364%	40.909%	*40.909%*	11.429%	17.647%
			2-shot	**58.621**%	**50.0%**	*48.235%*	30%	**51.515**%	27.273%
		Simplified	1-shot	*54.701%*	38.202%	44.384%	36.490%	36.087%	*29.474%*
			2-shot	53.125%	43.333%	**49.412**%	**46.341**%	34.211%	**44**%
		Abstract	1-shot	44.545%	*47.619%*	47.963%	40.456%	*39.720%*	31.579%
			2-shot	50%	38.095%	36.047%	30.952%	9.523%	14.286%
	kNN (mind2web)	Detailed	1-shot	**52.941**%	*40%*	36.364%	**38.235**%	*25.806%*	**29.412**%
			2-shot	*45.455%*	**42.105**%	12.698%	23.529%	13.725%	26.667%
		Simplified	1-shot	44.444%	31.25%	15.278%	30.556%	**30.0**%	*23.529%*
			2-shot	21.429%	20.833%	**38.961**%	35.135%	10.345%	12.5%
		Abstract	1-shot	14.286%	6.667%	21.212%	36.842%	6.667%	13.333%
			2-shot	27.273%	26.316%	19.298%	32.353%	3.030%	4.545%

Table 2: Comparison of user description specificity on Mind2Web Test sets.

Model	HTML Truncation	Prompt Setup	Cross-Task		Cross-Website		Cross-Domain	
			Recall	**Ele. Acc**	**Recall**	**Ele. Acc**	**Recall**	**Ele. Acc**
Claude2	No Truncation	1-shot	43.243%	36.364%	40.909%	40.909%	11.429%	17.647%
		2-shot	**58.621**%	50.0%	*48.235%*	30%	**51.515**%	27.273%
	Truncation at Top-10	1-shot	38.297%	46.667%	34.783%	37.5%	20%	*33.333%*
		2-shot	43.75%	*51.613%*	34.375%	38.462%	22.807%	37.5%
	Truncation at Top-50	1-shot	40%	43.75%	43.182%	**50**%	30.556%	30%
		2-shot	*48.649%*	**51.724**%	**59.77**%	*45.238%*	*47.368%*	**43.333**%

Table 3: Comparison of Truncation strategies for few-shot prompting on Mind2Web Test sets.

3.4.3 Impact of HTML Truncation

Raw HTML can include thousands of elements per page and such a large bulk of information can become difficult to process using LLM models [4]. Even in our reported accuracy from Table 2, we notice that 2-shot prompting hurts the performance - most likely due the the increased length of input sequence. Thus, it might be useful to filter out uninformative elements and truncate the HTML before passing it to the LLM. To test this hypothesis, we experiment with a pretrained HTML filtering module proposed by Mind2Web. This module returns the top-k elements from the HTML and creates a smaller snapshot of the DOM. In our experiments, we use $k = [10, 50]$ as majority of the data samples have the ground truth elements among this region (see Section A.4 in appendix).

Table 3 reports the results for varying levels of truncation. 💡 Truncation significantly improved the performance compared to cases when no truncation was performed. It also resolved the issue with 2-shot prompting observed in table 2 by effectively reducing the input token size in the LLM prompt.

3.4.4 Impact of the 'Role' Assumed by the LLM

This analysis was motivated by the fact that– the perception of "relevant UI elements" might vary from user to user [33]. For example, a UI/UX designer cares more about the interaction flow of the application whereas a general user might only be interested in finding their information as soon as possible. To simulate these roles, we investigate three different personas in this paper - 1) Generic User; 2) Web Assistant; 3) UI Designer. Persona is a linguistic representation of a fictional individual, enabling LLM to mimic the actions of this imaginary character [34]. For each persona, we generate multiple candidate prompts and empirically pick the best prompt by observing the performance on a subset of the validation set (see Section A.2 in appendix for details). Our final prompts for each persona are shown in Table 4

Persona	**Prompt**
Web Assistant	You are a web assistant designed to help users navigate a website and troubleshoot issue. Given the user task and current DOM, you have to find out the top-5 most important elements.
Generic User	You are a user surfing a website. You have a specific goal for which you want to find out the top-5 most important elements in the current DOM.
UI Designer	You are a UI designer working to improve the user experience of a website. Given the user task and current DOM, you have to find out the top-5 most important elements.

Table 4: Prompt message for each persona.

Model	Persona	Prompt Setup	Cross-Task Recall	Cross-Task Ele. Acc	Cross-Website Recall	Cross-Website Ele. Acc	Cross-Domain Recall	Cross-Domain Ele. Acc
Claude2	Web Assistant	1-shot	35.135%	33.333%	**61.25%**	**60%**	15.625%	24.138%
		2-shot	40.741%	*43.478%*	29.069%	41.463%	**60.417%**	*25%*
	Generic User	1-shot	*41.935%*	39.286%	37.349%	25.641%	7.813%	13.793%
		2-shot	**73.077%**	**55.556%**	*48.529%*	40%	13.235%	22.581%
	UI Designer	1-shot	35.714%	30.769%	15.294%	31.707%	12.069%	20%
		2-shot	32.143%	32%	42.857%	*45%*	*30.159%*	**29.630%**

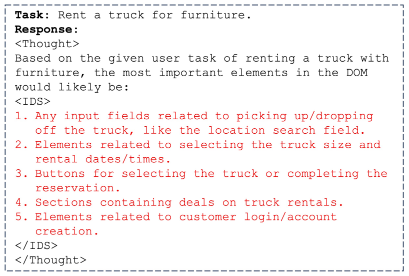
Table 5: Comparison of choice of persona on Mind2Web Test sets.

Table 5 reports the performance of `Claude2` based on the choice of persona in input prompt. Web Assistant persona performed significantly better (except for the User Persona for 2-shot prompting on Cross-Task) than the rest of the persona.

3.4.5 Error Patterns in LLM Response

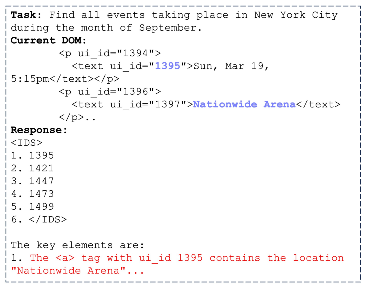
During our experiments, we noticed two types of errors most frequently - 1) Failure to follow instruction; and 2) Hallucination. Here we provide two such examples in Figure 4.

In the first example (Figure 4a), the model failed to follow the instruction to respond with the IDs within the `<ID>` tag, rather it provided the reasoning of its choice. In the second example (Figure 4b), the model hallucinated a non-existing DOM element by merging the information (*i.e*, nodeID and text description) from two existing elements.



```
Task: Rent a truck for furniture.
Response:
<Thought>
Based on the given user task of renting a truck with
furniture, the most important elements in the DOM
would likely be:
<IDS>
1. Any input fields related to picking up/dropping
   off the truck, like the location search field.
2. Elements related to selecting the truck size and
   rental dates/times.
3. Buttons for selecting the truck or completing the
   reservation.
4. Sections containing deals on truck rentals.
5. Elements related to customer login/account
   creation.
</IDS>
</Thought>
```

(a) LLM gave proper reasoning, but didn't give the ID of the HTML elements as instructed.



```
Task: Find all events taking place in New York City
during the month of September.
Current DOM:
       <p ui_id="1394">
         <text ui_id="1395">Sun, Mar 19,
5:15pm</text></p>
       <p ui_id="1396">
         <text ui_id="1397">Nationwide Arena</text>
       </p>..
Response:
<IDS>
1. 1395
2. 1421
3. 1447
4. 1473
5. 1499
6. </IDS>

The key elements are:
1. The <a> tag with ui_id 1395 contains the location
"Nationwide Arena"...
```

(b) Reference to wrong DOM element that was hallucinated from other elements in the DOM.)

Figure 4: Examples of frequently occurring errors. We show only the relevant fractions of the outputs for the ease of the readers.

4 Conclusions and Future Work

In this study, we investigate the capacities of LLM for retrieving important information from a webpage given user task description. Our current study is solely based on Anthropic's `Claude2` model. In the future, we would like to extend our study to other large language models such as GPT-4[35], Llama V2[36], Vicuna[37], PaLM 2[38] and so on. One of the primary challenges in extending our study to these models lies in fitting a substantial HTML context within their comparatively limited context lengths. Future work can explore how to efficiently encode this information while preserving the syntactical and contextual integrity of the DOM. The truncation strategy we used in Section 3.4.3 can be further modified for this purpose.

A major finding of our work is the co-relation of the specificity of user query and the performance. Future work can learn from our experiment and work on making these models more responsive. i.e: knowing when to intervene or ask for further clarification. A very interesting direction we are interested in is - even if the prompts are abstract, how can we make LLMs understand the user intention better? This could entail refining the models' ability to interpret and respond to nuanced user intentions. The model can also benefit from personalization and contextualized information about the user and/or other metadata such as current location, time, or data. However, future research should also be mindful of the security concerns when using the personal information of the users.

References

[1] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding html with large language models, 2023.

[2] Bryan Wang, Gang Li, and Yang Li. 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, New York, NY, USA, 2023. Association for Computing Machinery.

[3] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[4] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023.

[5] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.

[6] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Empowering llm to use smartphone for intelligent task automation, 2023.

[7] Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents, 2023.

[8] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023.

[9] Jiaming Han, Renrui Zhang, Wenqi Shao, Peng Gao, Peng Xu, Han Xiao, Kaipeng Zhang, Chris Liu, Song Wen, Ziyu Guo, Xudong Lu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Xiangyu Yue, Hongsheng Li, and Yu Qiao. Imagebind-llm: Multi-modality instruction tuning, 2023.

[10] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[11] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models, 2023.

[12] Scott L. Fleming, Alejandro Lozano, William J. Haberkorn, Jenelle A. Jindal, Eduardo P. Reis, Rahul Thapa, Louis Blankemeier, Julian Z. Genkins, Ethan Steinberg, Ashwin Nayak, Birju S. Patel, Chia-Chun Chiang, Alison Callahan, Zepeng Huo, Sergios Gatidis, Scott J. Adams, Oluseyi Fayanju, Shreya J. Shah, Thomas Savage, Ethan Goh, Akshay S. Chaudhari, Nima Aghaeepour, Christopher Sharp, Michael A. Pfeffer, Percy Liang, Jonathan H. Chen, Keith E. Morse, Emma P. Brunskill, Jason A. Fries, and Nigam H. Shah. Medalign: A clinician-generated dataset for instruction following with electronic medical records, 2023.

[13] Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization, 2023.

[14] Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning, 2023.

[15] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[16] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2998–3009, October 2023.

[17] Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action, 2022.

[18] Kaizhi Zheng, Kaiwen Zhou, Jing Gu, Yue Fan, Jialu Wang, Zonglin Di, Xuehai He, and Xin Eric Wang. Jarvis: A neuro-symbolic commonsense reasoning framework for conversational embodied agents, 2022.

[19] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. *arXiv preprint arXiv:2301.13166*, 2023.

[20] Jianing Wang, Chengyu Wang, Chuanqi Tan, Jun Huang, and Ming Gao. Boosting in-context learning with factual knowledge, 2023.

[21] Kumar Shridhar, Harsh Jhamtani, Hao Fang, Benjamin Van Durme, Jason Eisner, and Patrick Xia. Screws: A modular framework for reasoning with revisions, 2023.

[22] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023.

[23] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models, 2023.

[24] Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents, 2023.

[25] Anthropic. Model card and evaluations for claude models, Jul 2023.

[26] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020.

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

[28] David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George Foster. Prompting PaLM for translation: Assessing strategies and performance. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15406–15427, Toronto, Canada, July 2023. Association for Computational Linguistics.

[29] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online, August 2021. Association for Computational Linguistics.

[30] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023.

[31] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[32] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. Stylette: Styling the web with natural language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[33] Lisandra Maioli. *Fixing bad ux designs: Master proven approaches, tools, and techniques to make your user experience great again*. Packt Publishing Ltd., 2018.

[34] Myra Cheng, Esin Durmus, and Dan Jurafsky. Marked personas: Using natural language prompts to measure stereotypes in language models, 2023.

[35] OpenAI. Gpt-4 technical report, 2023.

[36] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[37] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

[38] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023.

[39] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

A Appendix

A.1 Design of Fixed Examples Used in Few-shot Prompting

Figure 5 shows the fixed examples we used in our experiment. To craft these examples, we preserved the bare-minimum information in the DOM and replaced everything else with ‘. .’.

```
Here are two examples -
User Task:  fill out a registration form
Current DOM: <html>..<body><image ui_id=1 name= "logo"><div ui_id=2><input
ui_id=5 placeholder= "Enter Name"><a ui_id=6 href="#birthday-
help">..<select ui_id=7 aria-label= "Month"><button ui_id=10 name= "sub-
mit"></div>..</body>..</html>
Your Response:
<Thought>The most important elements for this task would be input fields
(to put in the details), submission buttons (to submit the form), and help
instructions (to read instructions)</Thought>
<IDS>
1.  2
2.  5
3.  7
4.  10
5.  6
</IDS>
User Task:  place an order
Current DOM: <html>..<body><image ui_id=3 name= "Welcome to our
website!"><div ui_id=8><input ui_id=5 placeholder= "Enter Product
Name"><button ui_id=10 name= "Search"></div>..<a ui_id=25 href="#new
deals"><div ui_id=29 href="#sales"><span ui_id=25>"Account Set-
ting"</span>...</body>..</html>
Your Response:
<Thought>The most important elements for next step might be to fill up the
search field and search button, and relevant deals.</Thought>
<IDS>
1.  5
2.  10
3.  8
4.  25
5.  29
</IDS>
You must always include the <Thought> and <IDs> open/close tags or else
your response will be marked as invalid.  <IDs> must include the ui_id num-
ber of the elements.
```

Figure 5: Fixed examples used in the input prompt.

A.2 Exploration of Persona Prompt

All scores are reported on a small validation set from Cross-Task comprising on 2 tasks and 14 steps. These tasks are chosen randomly and kept constant for all the variants of prompt shown below. The performance scores on this subset, along with the candidate prompts, are shown in Tables 6, 7, and 8.

Prompt	Recall
You are a web assistant designed to improve the user experience of a website. Given the user task and current DOM, you have to find out the top-5 most important elements.	33.333%
You are a web assistant designed to help users navigate a website. Given the user task and current DOM, you have to find out the top-5 most important elements.	27.778%
You are a web assistant designed to help users navigate a website and troubleshoot issue. Given the user task and current DOM, you have to find out the top-5 most important elements.	**50.0%**
You are a web assistant designed to help users find relevant information. Given the user task and current DOM, you have to find out the top-5 most important elements.	44.444%

Table 6: Initial Candidate prompts for Web Assistant persona.

Prompt	Recall
You are a user surfing a website. You have a specific goal for which you want to find out the top-5 most important elements in the current DOM.	**38.889%**
You are a user who is browsing a website to perform a task. You want to find out the top-5 most important elements in the current DOM for your task.	22.222%

Table 7: Initial Candidate prompts for Generic User persona.

Prompt	Recall
You are a UI designer working to improve the user experience of a website. Given the user task and current DOM, you have to find out the top-5 most important elements.	**50.0%**
You are a UI designer working on Quality Assurance of a website. Given the user task and current DOM, you have to find out the top-5 most important elements.	33.333%
You are a UI designer who wants to make user-friendly interface. Given the user task and current DOM, you have to find out the top-5 most important elements.	22.222%
You are a UI designer who wants to ensure user can find task-specific information easily. So, for a given task and current DOM, you want to find out the top-5 most important elements.	27.778%

Table 8: Initial Candidate prompts for UI Designer persona.

A.3 Creation of Task Description Based on Varying Specificity

We use GPT-4 to create *simplified* and *abstract* task description for mind2web. We use a simple few-shot prompting technique to show how to simplify the task description while preserving the meaning. The first author did a manual investigation over a generated subset of task description to ensure its quality. The specific prompt used for the creation of task description is shown below in Figure 6. In the prompt, A1 and A2 denote *simplified* and *abstract* task description respectively.

Your task is to remove variable names in a task description. Variables can be place, date, product name and so on. Respond with the variables you removed and the simplified task description after removal. Provide two simplifications - 1) where no variable exists, 2) where you may keep some variables that are necessary to preserve the meaning.

Here are two examples:
Task description - Find the nearest Chinese restaurants in Washington DC.
Response -
<Reasoning>Chinese is a specific cuisine. Washington DC is a place. So, I need to remove them.</Reasoning>
<Variables>var1: Chinese, var2: Washington DC</Variables>
<Template>Find the nearest [var1] restaurants in [var2] </Template>
<A1>Find the nearest restaurants</A1>
<A2>Find the nearest restaurants in Washington DC</A2>

Task description - Check for pickup restaurant available in Boston, NY on March 18, 5pm with just one guest.
Response-
<Reasoning>Boston, NY is a place. March 18 is a date. 5pm is a time. one is a number. So, I need to remove them.</Reasoning>
<Variables>var1: Boston, NY, var2: March 18, var3: 5pm, var4: one</Variables>
<Template>Check for pickup restaurant available in [var1] on [var2], [var3] with just [var4] guest. </Template>
<A1>Check for pickup restaurant</A1>
<A2>Check for pickup restaurant in Boston, NY at 5pm</A2>

Always think step-by-step and include <Reasoning>, <Variables>, <Template>, <A1>, <A2> tags.

Figure 6: Prompt for GPT-4 model to construct abstract and simplified task description.

A.4 Filtering Module from Mind2Web

The filtering module in Mind2Web paper is based on DeBERTa [39] model. We directely used the pretrained module provided in their paper in our analysis. We start by analyzing their accuracy as shown as Figure 7. For majority of the test samples, the ground truth element is present within the Top-10 elements returned by the model. Hence, we truncate the results from DeBERTa at Top-10. To understand the impact of a larger window, we also show results from Top-50 as well.

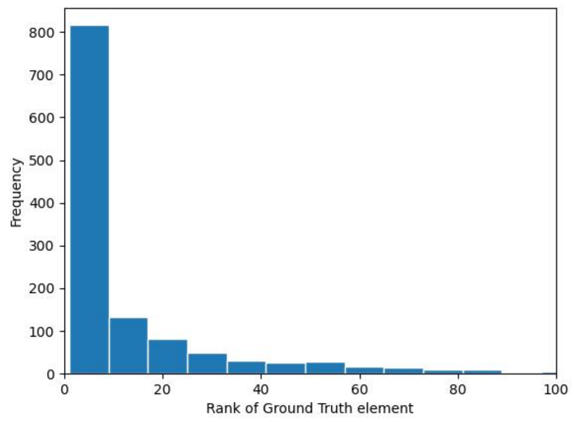


Figure 7: Distribution of Rank of ground truth elements in the prediction of Ranking Module. For majority of the data samples, the ground truth is present in within Top-10.

A.5 Distribution of the HTML Element Count and Its Impact on Performance

For this analysis, we categorized the results into five clusters based on the number of HTML elements in each sample. Then, we calculated the sample count within each cluster along with the corresponding element accuracy

In Figure 8, we demonstrate this analysis as a pie-chart, where each pie represents one of these five clusters. Within each pie, we show the number of samples that fall within the corresponding cluster; and the accuracy is demonstrated right outside the pie (shown as "acc."). The legend in the figure shows the range of the number of HTML element for each cluster.

It is worth noting that as the number of HTML elements increases, the available samples diminish. Furthermore, it appears that the LLM's performance falters when dealing with substantially longer HTML content, specifically within the range of $(3248.6, 3962.0]$.

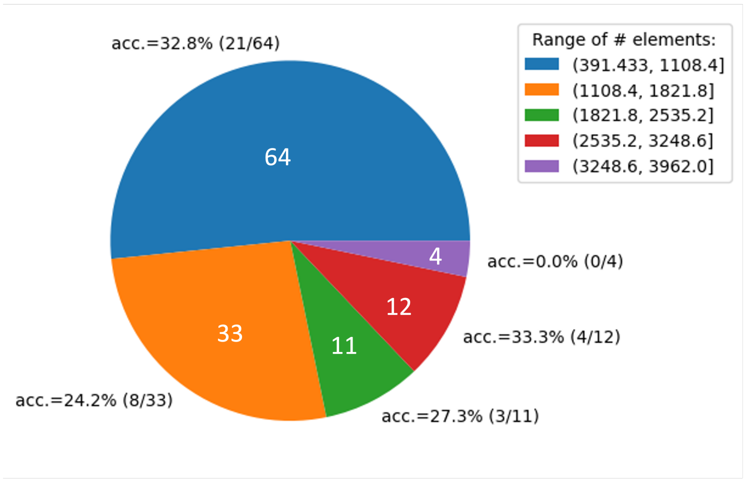


Figure 8: Distribution of Accuracy with respect to Number of HTML elements