Mastering WEBLINX: Elevating Performance through Advanced Prompting

Vishvak Raghavan and Aayush Kapur
McGill University, Computer Science
vishvak.raghavan@mail.mcgill.ca aayush.kapur@mail.mcgill.ca

Abstract

Conversational web navigation poses a formidable challenge as digital agents navigate web browsers, responding to user instructions across multi-turn dialogues to accomplish real-world tasks. The introduction of WEBLINX, a robust benchmark derived from 2300 expert demonstrations, has provided an invaluable platform for training and evaluating agents across diverse scenarios (Lù et al., 2024). Various large language models (LLMs), including text-only, image-to-text, and multimodal variants, have undergone evaluation on the WEBLINX benchmark. Notably, smaller fine-tuned text-only decoders (S-LLaMA (Xia et al., 2023) and LLaMA2 (Touvron et al., 2023)) have demonstrated superior performance compared to zero-shot LLMs and fine-tuned multimodal models, yet room for improvement persists. Recent advancements in prompting have highlighted LLMs' prowess as adept few-shot and zero-shot reasoners. Furthermore, guiding LLMs with longer-term historical context has shown promise in enhancing performance. Our study delves into how advanced prompting methods impact text-only decoder capabilities in conversational web navigation. By evaluating on the WEBLINX benchmark, we demonstrate that integrating the main conversation topic into the prompt offers enhanced contextual guidance, leading to more accurate actions. Furthermore, we observe that smaller LLMs struggle as zero-shot and few-shot reasoners in conversational web navigation. Our findings highlight the need for a nuanced approach to prompt designs, considering both the capabilities and limitations of different model sizes.

1 Introduction

With the rise of conversational assistants revolutionizing human-computer interactions, conversational web navigation has emerged as a pivotal area in natural language processing (NLP). Conversational web navigation involves a digital agent to control a web browser based on user instructions in a multi-turn dialogue (Lù et al., 2024).

Recently, the introduction of WEBLINX (Lù et al., 2024) has marked a significant advancement in this field. WebLINX is a benchmark comprising over 2000 demonstrations of conversational web navigation conducted by human experts across more than 150 real-world websites (Lù et al., 2024). The principle underlying WEBLINX involves encapsulating the HTML state of a webpage within a single, comprehensive prompt that also outlines a range of possible actions. This prompt is then presented to a LLM to determine the subsequent course of action, taking into account the user's input within the given context. The construction and content of the prompt are crucial, given the constraint of a finite token limit, prompting WEBLINX to retain only essential information. To refine the selection of potential actions, a smaller model is employed to identify a list of top candidate HTML elements for consideration—specifically, ten elements. Additionally, the prompt incorporates the last five interactions from the user for added context as well as past actions already performed by the agent. The prompt explicitly instructs the LLM to recommend solely the final action deemed appropriate, without delving into extraneous details.

WEBLINX provides a robust platform that facilitates the training and evaluation of dialogue-enabled navigation agents across diverse scenarios. Thus far, WebLINX has been used to assess 19 zero-shot and fine-tuned models employing various input modalities, such as smaller image-to-text, larger text-only decoders, and multimodal models (capable of accessing both image and text). Text-only decoders, such as LLaMA2 and Sheared-LLaMA, consistently demonstrate superior performance compared to image-only and multimodal models, particularly after fine-tuning. The exploration of diverse prompting strategies presents multiple opportunities for further enhancements.

For conversational agents to effectively manage and respond within extended dialogues, they must accurately recall past contexts and seamlessly integrate this information into their responses (Maharana et al.). This is crucial for generating responses that are consistent with the ongoing narrative. However, LLMs often struggle with processing lengthy conversations and understanding the complex temporal and causal dynamics within dialogues (Maharana et al.). Hatalis et al. (Hatalis et al., 2023) have shown that providing a description of the current situation to LLM agents in the prompts enables the agents to draw on data from long-term memory. In the application of conversational web navigation, capturing the main theme of the ongoing conversation and incorporating it into the prompt ensures the agent is aware of this context when determining subsequent actions.

Further, recent studies indicate that LLMs' reasoning abilities can be boosted by applying strategies that introduce step-by-step reasoning. One effective method, few-shot Chain of Thought (CoT) (Wei et al., 2022) prompting, provides models with examples of reasoning processes instead of simple question-answer pairs. This method breaks down complex tasks into manageable steps, improving performance on tasks requiring multi-step reasoning. Using only eight CoT examples with a PaLM 540B model has set new benchmarks in accuracy for math word problems on the GSM8K benchmark, outperforming regular few-shot learning and even optimized GPT-3 models (Wei et al., 2022). Additionally, Kojima et al. (Kojima et al., 2022) have shown that LLMs can perform well in zero-shot reasoning tasks by simply prefacing questions with *"Let's think step by step"*, encouraging a sequential thought process. This zero-shot-CoT approach, without the need for task-specific prompts, has shown impressive versatility and effectiveness across various reasoning tasks, including significant improvements in MultiArith and GSM8K tasks (Kojima et al., 2022). Both few-shot CoT and zero-shot CoT approaches show considerable potential for enhancing model performance on the WEBLINX benchmark. By integrating these strategies into the prompting process, models can generate more reasoned and stepwise actions, potentially leading to significant improvements in task execution and accuracy.

In this study, we explore various strategies for delivering information to LLMs via prompts, examining how different prompting techniques can influence the models' performance and outcomes. Specifically, we first implement a prompt engineering technique that focuses on capturing the main topic of an ongoing conversation and integrating it into the prompt's context to provide more long-term contextual memory to the model. Our second technique involves employing zero-shot CoT to encourage a detailed, stepwise thought process before responding to each question. Lastly, we plan to use a few-shot CoT approach by adding a selection of randomly chosen examples that showcase both the desired action and the underlying reasoning process directly within the prompt. Through these methods, we aim to assess the effectiveness of these prompting approaches in enhancing LLMs' reasoning capabilities and their ability to generate more accurate and contextually relevant actions.

We evaluated the performance of Sheared-LLaMA-1.3B and Flan-T5-780M, both fine-tuned on the WEBLINX benchmark, using three distinct prompting techniques. Our findings indicate that including the main context/theme of the conversation within prompts, boosts the capabilities of LLMs in conversational web navigation tasks. This strategy improves the models' understanding of user intentions, resulting in more accurate action identification, as reflected in an improvement in the intent of the action models. Despite its potential in handling unseen subcategories, zero-shot CoT did not improve overall model performance. Furthermore, the incorporation of few-shot CoT, which adds examples with detailed explanations for the correct actions, resulted in a decline in overall performance. This decrease is primarily due to the model's struggle to generate text responses that align with user inputs. Our experiments demonstrate the significance of applying enhanced prompting strategies for guiding LLMs and boosting their performance with the same architecture for conversational web navigation.

2 Related Works

2.1 WEBLINX

Before the introduction of WEBLINX (Lù et al., 2024), existing conversational assistants required plugins to enable LLMs to effectively interact with the web. The efforts undertaken for WebLINX aim to bridge this gap in the conversational capabilities of these models for website navigation. The challenge lies in facilitating structured engagement of LLMs with the diverse web environment and enabling them to generalize to novel scenarios. In the WEBLINX paradigm, each action is linked with a Document Object Model (DOM) tree, browser screenshots, and frames from demonstration-level video recordings. The conversational assistant in WEBLINX accommodates both text-only and multimodal inputs depending on the use case. One notable advantage of the WEBLINX paradigm is its facilitation of a natural conversational flow between the user and the agent, fostering incremental instruction buildup through turn-based dialogue navigation.

Due to the abundance of HTML elements on web pages, fitting them all into the context window of LLMs for predicting the next action from the DOM tree poses a challenge. WEBLINX proposed Dense Markup Ranking (DMR) for narrowing down the elements from the HTML input to a few elements suitable for the context window of the LLM (Lù et al., 2024). To manage input sequence length exceeding model limits even after candidate selection, the sequence undergoes strategic truncation to ensure consistency below the desired threshold. Thus, LLMs are furnished with a concise representation of the DOM tree, coupled with action history, detailed instructions, and optionally, screenshots, to predict the subsequent action accurately. In WEBLINX, the prompt provided to LLMs encompasses the HTML representation of the web page, the first and last four user utterances, the top candidate actions for the current turn determined using DMR, and the model's past actions. With this information, the model is asked to select the best action.

The WEBLINX paper revealed that smaller finetuned models, such as Sheared-LLaMA, outperform much larger models on the WebLINX benchmark. Conversely, zero-shot models like GPT-4 (Zheng et al., 2024) exhibit poor performance due to their lack of situational awareness. Moreover, to develop agents capable of functioning effectively in real-world scenarios, it is crucial to construct models capable of generalizing to unseen situations. This underscores the importance of exploring various approaches to improve prompting to guide the LLM for predicting the next action. Our primary focus will be on refining various facets of the prompting mechanism within the action prediction pipeline, with special attention given to text representations. Text representations are prioritized due to their lower computational demands compared to multimodal data.

2.2 Long term memory prompting

For agents to effectively handle and respond within extended dialogues, it is essential for them to accurately recall previous contexts and integrate this information into subsequent responses (Maharana et al.). Conversational agents must effectively draw on relevant past context to produce responses that align with the continuing narrative (Maharana et al.). However, LLMs face difficulties in processing extended conversations and grasping the long-range temporal and causal dynamics within dialogues (Maharana et al.; Xu et al., 2022; Jang et al., 2023)

Incorporating memory systems with conversational agents enhances the coherence and contextuality of interactions between the agent and the user (Hatalis et al., 2023). A long-term memory component helps maintain focus on the overarching goal while the agent navigates through intermediate steps. Hatalis et al. (Hatalis et al., 2023) suggested that LLM agents could use the description of the current situation and their prompt as a form of working memory, while drawing on data from a long-term storage to enrich the prompt, thereby simulating the recall of information from long-term memory. One effective method for implementing this involves using dialogue embeddings stored in an external vector database that facilitates fast maximum inner-product searches (Hatalis et al., 2023). The dialogues can then be retrieved by approximate nearest neighbours (ANN) algorithms.

This methodology is exemplified in retrieval augmented generation (RAG) techniques (Gao et al.). Basic RAGs operate by segmenting raw data into manageable chunks and performing retrieval as previously outlined. The data retrieved through this process is then incorporated into prompts to facilitate output generation, effectively integrating long-term memory with LLMs for producing responses. This approach provides a framework for integrating long-term memory with an LLM to generate outputs.

In our application, the goal is to capture the main theme of the ongoing conversation and incorporate it into the prompt. This ensures the agent is aware of this context when determining subsequent actions. As the conversation evolves, new utterances should be added to the main storage, allowing for a comprehensive history to be maintained, which aids in selecting the theme for future prompts.

2.3 Reasoning ability of LLMs

It has been widely observed that pretrained LLMs often struggle with tasks requiring complex reasoning, particularly those involving multi-step processes (Brown et al., 2020; Rae et al., 2021; Smith et al., 2022). While traditional fine-tuning methods have been successful in enhancing task-specific performance (Rajani et al., 2019; Cobbe et al., 2021; Zelikman et al., 2022; Nye et al., 2021), they come with limitations such as the need for extensive labeled data, potential over-fitting (Gururangan et al., 2018; Niven and Kao, 2019), and reliance on narrow task distributions (Mccoy et al.). In response to these challenges, approaches such as few-shot and zero-shot prompting have emerged as promising techniques (Reynolds and McDonell, 2021; Brown et al., 2020; Radford et al., 2018). Few-shot prompting involves conditioning the model on a small number of examples during inference, while zero-shot prompting relies solely on natural language instructions to guide the model (Brown et al., 2020; Radford et al., 2018). Both methods significantly reduce the dependency on task-specific data and offer potential for robust performance across various tasks (Reynolds and McDonell, 2021; Brown et al., 2020; Radford et al., 2018). Nevertheless, few-shot learning exhibits limited efficacy on tasks necessitating reasoning skills, and its performance typically does not notably enhance with larger model scales (Liu et al., 2022; Rae et al., 2021).

Recent research has shown that reasoning capabilities of LLMs can be improved by incorporating step-by-step reasoning strategies. Few-shot Chain of thought prompting (CoT) is one such approach where the model is provided with step-by-step reasoning examples rather than standard question-answer pairs (Wei et al., 2022) (Wang et al., 2022). This facilitates the decomposition of complex reasoning tasks into simpler steps, thereby enhancing the model's performance on multi-step reasoning tasks. Notably, CoT prompting has demonstrated significant performance improvements over regular few-shot learning on arithmetic, commonsense, symbolic, and other logical reasoning tasks, especially when combined with large-scale language models (Rae et al., 2021; Wei et al., 2022; Wang et al., 2022).

While the success of few-shot CoT is attributed to LLMs' ability for few-shot learning, Kojima et al. demonstrate that LLMs can also exhibit decent zero-shot reasoning capabilities (Kojima et al., 2022). By introducing a straightforward prompt, *"Let's think step by step"*, before answering each question, the zero-shot CoT approach facilitates step-by-step thinking and successfully generates plausible reasoning paths even in scenarios where standard zero-shot methods fail. Notably, zero-shot CoT proves to be versatile and task-agnostic, accommodating a wide range of reasoning tasks without the need for prompt customization per task. For instance, it achieves substantial score gains on tasks such as MultiArith and GSM8K, indicating its efficacy in enhancing zero-shot reasoning capabilities (Kojima et al., 2022). While zero-shot CoT generally falls short compared to few-shot CoT in various scenarios, few-shot CoT's effectiveness heavily relies on the specific prompt designs for each task (Kojima et al., 2022). Few-shot CoT's performance suffers when the question types in the prompt examples do not align with those of the tasks (Kojima et al., 2022). Conversely, the adaptability of a singular prompt across a range of reasoning tasks in zero-shot CoT could prove more advantageous in broader logical reasoning applications.

CoT shows promising potential for prompting in action models. EmbodiedGPT (Mu et al., 2023), an end-to-end multi-modal foundation model for embodied AI, has recently incorporated CoT during pretraining. Operating in a multi-turn question answering format like WEBLINX, EmbodiedGPT demonstrates higher success rates compared to other baselines across diverse demonstrations (Mu et al., 2023), showcasing the effectiveness of CoT in facilitating multi-turn dialogue within action models.

3 Modeling

3.1 Models

In our experiments, we exclusively used two text-only decoder models since our focus was on prompts for text-only systems. The first model employed was Sheared-LLaMa-1.3B (Xia et al., 2023), a text-only chat model. The second was Flan-T5-780M (Chung et al., 2022), a text-only model designed for processing instructions. This selection allowed us to assess how different types of models—chat-based versus instruction-based—respond to our prompting strategies. We used the Sheared-LLaMa-1.3B and Flan-T5 models, which had been previously fine-tuned on the WEBLINX dataset by the original authors of the WEBLINX framework.

3.2 Baseline

To effectively evaluate the impact of our prompting approaches on the aforementioned models, we established a baseline using the original prompting setup detailed in the WEBLINX paper (see Figures 1 and 5 in Appendix C and Figures 9 and 13 in Appendix D for example prompts). This allowed us to conduct a direct comparison between the standard prompts outlined in the study and the prompting techniques we implemented. By maintaining this baseline, we could measure any performance variations and accurately assess the efficacy of our prompting strategies against the predefined standards set by the initial results. To set up the baseline, we re-executed the results for the Sheared-LLaMa-1.3B and Flan-T5 models using the existing code and setup provided in the original study.

3.3 Incorporating evolving main topic of conversation into prompt

Our first approach to refining prompt engineering involved enhancing the contextual depth provided to the models. Typically, the models were prompted with the initial user utterance and the last four user utterances of the conversation. This setup, however, might omit crucial elements of the ongoing dialogue, potentially missing the underlying motivations or themes driving the conversation.

To address this limitation, we introduced a technique designed to encapsulate the main topic of the conversation within the prompt. This method aims to give the action model a semblance of “long-term memory”, allowing it to better grasp the broader context and flow of the conversation. We identified the main topic by extracting key phrases and keywords from the entire conversation history. This was done using a domain independent keyword extraction algorithm, Rapid Automatic Keyword Extraction (RAKE) (Sharma). Using the top 20 keywords, we pinpointed the most relevant utterance that best encapsulates the core subject or issue being discussed. This selected utterance is then labeled as the "main theme/topic" and is systematically included in the prompt. This process is dynamic and occurs at every turn, allowing the “main theme” of the prompt to evolve in tandem with the conversation. As the dialogue progresses and diversifies, the main theme is continually updated to reflect the most current and relevant themes.

We hypothesized that by integrating a dynamically updated main topic derived from the entire conversation, the model would not only maintain a more cohesive understanding of the dialogue but also show improved performance in responding appropriately and contextually. This enhancement was anticipated to empower the model to act more effectively, leveraging a comprehensive view of the dialogue’s direction and purpose at each step.

An example prompts that incorporates the main topic/theme of the conversation can be found in Figures 2 and 6 in Appendix C as well as Figures 10 and 14 in Appendix D.

3.4 Zero-shot Chain of thought prompting

To refine our model’s capability for reasoned decision-making, we integrated a zero-shot CoT prompting strategy (Kojima et al., 2022). This method involves enhancing the prompt with the phrase *“Let’s think step by step”*, aimed at fostering a logical, step-by-step approach to problem-solving before the model attempts to answer a query (Kojima et al., 2022). In traditional settings, zero-shot CoT utilizes a dual-prompt system (Kojima et al., 2022). The initial prompt includes *“Let’s think step by step”* to initiate a comprehensive thought process, generating a detailed reasoning path. Subsequently, a second prompt is introduced, which incorporates the reasoned response from the first prompt. This second prompt is crafted to refine the output, ensuring that the response is in the desired format that directly answers the question or completes the task.

Given the constraints of our task, which accommodates a conversation between instructor and navigator in a time efficient manner, we adapted the dual-prompt approach. We modified the initial prompt to invoke step-by-step reasoning and get the output action in one prompting action. The revised prompt reads: *“Let’s think step by step to come up with reasoning and use that reasoning for this task”*. This streamlined approach allows us to extract reasoned, actionable responses in just one step, enhancing efficiency and coherence. Examples of zero-shot CoT prompts tailored for the WEBLINX benchmark can be found in Figures 3 and 7 in Appendix C as well as Figures 11 and 15 in Appendix D.

3.5 Few-shot Chain of thought prompting

To enhance the effectiveness of our prompts, we aimed to implement few-shot CoT by incorporating an example of a successfully executed demonstration, complete with a detailed explanation, directly into the prompt. We hypothesized that by observing a successful example, the agent would be better equipped to generalize to new, unseen situations. Specifically, if the example was framed clearly enough to capture the essential details of another demonstration, it could aid the agent in making decisions for the current utterance it is processing.

To implement this, we proposed to review all utterances in chat-based demonstrations from the validation set. We preprocessed these utterances by removing stop words and unnecessary words or phrases like “okay”, “sure”, and “alright”. From these dialogues, our goal was to identify the main action of each demonstration. This was accomplished by converting each dialogue into its corresponding embedding using the *MiniLM* model (Wang et al., 2020). We then calculated the cosine similarity between each dialogue and all other dialogues in the dataset, selecting the top-3 most similar dialogues as the primary representatives of the conversation for that particular demonstration.

Using these representative dialogues and the processed list of all dialogues from the demonstration, we extracted the relevant question-answer pairs. From these pairs, we manually crafted examples for each demonstration that succinctly encapsulated the problem the agent was addressing, along with an explanation for the correct action. We carefully abstracted away HTML elements and state-related information from these prompt examples, as they would not be included in the concise, two-line examples we aimed to incorporate in our prompts. The purpose of creating these examples was to assist the agent in its decision-making process when determining the appropriate response based on the current prompt. We add a random example, from the set of all manually curated explanations to the prompt.

An example of the addition of few-shot CoT prompt can be found in Figures 4 and 8 in Appendix C as well as Figures 12 and 16 in Appendix D. It is important to note that the example omits HTML elements and aims to assist the agent in determining the necessary steps to choose the correct course of action. This approach was adopted to ensure that even if the demonstration originates from a different category than the current prompt to which it is being added, the generalized information should aid the model in making the correct decision.

4 Datasets

In this study, we assess our model using the WEBLINX benchmark dataset as outlined in the original paper. The WEBLINX benchmark, designed for conversational web navigation, features 2,337 recorded demonstrations of interactions between an instructor and a navigator who work together to complete tasks on 155 different websites. Each demonstration averages 43 turns. Since our research focuses on altering the prompts used for action prediction, it is crucial to evaluate our model on the same dataset that was used to establish the benchmark. This approach ensures that we can accurately compare our model’s performance against the established results from previous studies that utilized the same benchmark.

Each demonstration captures real-time interactions, with the navigator recording their control of the web browser. Every demonstration, D, consists of a sequence of n states paired with n corresponding actions. At each turn, the state encompasses the website’s representation, and the action aligns with one of the intents listed in Table 9. The demonstrations are categorized into eight primary categories, which are further divided into as many as 50 subcategories. Each website is categorized under one main category and one subcategory. Given that a demonstration may involve multiple websites, it can be associated with one or more subcategories. Since we are only interested in text-only models, we ignored screenshots and video demonstrations given to image-only and multimodal models.

We performed evaluation on the TEST_{IID} split to assess in-domain generalization, and four out-of-domain splits (TEST_{WEB}, TEST_{CAT}, TEST_{VIS}, TEST_{GEO}) to test on various scenarios. Table 7 describes the demos in each split in more detail. Table 8 presents the number of demos for each split and the mean number of turns.

5 Evaluation

The evaluation metrics employed to assess the models performance on the WebLINX benchmark adhere to the metrics originally proposed by the authors of WebLINX. This decision ensures consistency and uniformity in the evaluation schema across different models and benchmarks. Specifically these metrics are turn-level evaluation metrics to measure the similarity between the predicted action and the reference action. The metrics used in the WebLINX benchmark are describe below.

5.1 Intent Matching (IM)

This metric does not evaluate a model's capability to predict the correct arguments but rather assesses the model's ability to accurately identify the intended action. When comparing a prediction, denoted as a' , with a reference action, denoted as a, the intent match is denoted as IM(a', a). If the intents are identical, IM(a', a) = 1, otherwise; IM(a', a) = 0.

5.2 Element Similarity using IoU

When comparing a prediction a' with reference a, the Intersection over Union (IoU) metric calculates the overlap between the predicted and reference elements. IoU is then scaled by the Intent Match (IM) score. This formulation prioritizes elements with substantial visual overlap, penalizes predictions significantly smaller or larger than reference elements, and assigns a score of 0 if the elements do not overlap.

5.3 Text Similarity using F1

WEBLINX measures the lexical similarity of text arguments using chrF, which calculates an F1 score based on n-gram overlap between text inputs a' and a. For URL inputs, this F1 score is applied to segments to compute the URLF. The resulting chrF/URLF is then scaled by the IM score.

5.4 Turn-level score and overall score

To facilitate better model comparisons, intent groups are divided into two categories by WEBLINX: the element group (EG), consisting of actions like *click*, *textinput*, and *submit*, evaluated using IoU; and the text group (TG), encompassing actions like *load*, *say*, and *textinput*, evaluated using F1 score. During each turn, if the action falls within EG, the score is determined by IoU. Conversely, if the action belongs to TG, the score is calculated using the F1 score. In cases where a turn involves actions from both TG and EG (i.e. *textinput*), the turn score is the product of IoU and F1 scores. The overall model score is then computed using the micro-average of turn-level scores.

6 Results

In this section, we report the results of our experiments defined in Section 3. We aggregate the results for the two models, Sheared-LlaMa-1.3B and Flan-T5-780M in Table 1. See Appendix A for additional results. We were unable to fully replicate the baseline results from the original WEBLINX paper for the Sheared-LlaMa-1.3B and Flan-T5 models (see Appendix E.3 for further explanation). However, the results we obtained in our experiments still serve as a useful proof of concept, demonstrating the potential impact of our prompting techniques on model performance in conversational web navigation.

6.1 Incorporating evolving main context of conversation to prompt

In Table 1, we observe that integrating the main theme of the conversation enhances the performance of both Sheared-LlaMa and Flan-T5 across various test splits. Tables 2, 3, 4 and 5 reveal that this benefit persists in every out-of-distribution split, except for TEST$_{\text{CAT}}$ in the case of Sheared-LlaMa and both TEST$_{\text{CAT}}$ and TEST$_{\text{VIS}}$ (albeit marginally) for Flan-T5. This suggests that LLMs still find unseen subcategories more challenging than familiar websites within the same categories, aligning with findings from the original WEBLINX paper. Despite these gains, including the main conversation topic does not resolve difficulties with unseen subcategories, although it improves performance across other splits. Most notably, the overall IM score is significantly higher than the baseline, while improvements in EG IoU and TG F1 scores, though present, are not as pronounced. This indicates that incorporating the main topic of the conversation into the prompt equips the model with a richer context of the user's objectives, enabling more accurate identification of the intended actions.

6.2 Zero-shot CoT

Table 1 demonstrates that zero-shot CoT does not enhance the performance of models in conversational web navigation tasks across both in-distribution and out-of-distribution splits for the Sheared-LlaMa and Flan-T5 models. This method not only failed to reach the performance levels ob-

Table 1: Aggregated results for the two models, Sheared-LlaMa-1.3B and Flan-T5-780M. We report results of intent match (using IM), element group (IoU), text group (F1), and the overall score (using micro-average on turn-level scores). All results are on TEST_{OOD} except the last column which is on TEST_{IID}.

Model	**Overall IM**	**EG IoU**	**TG F1**	**Overall Score Test_{OOD}**	**Overall Score Test_{IID}**
Sheared-Llama-1.3B					
Baseline	0.3003	0.0123	0.2693	0.2212	0.2612
Main theme	**0.3157**	**0.0157**	0.2672	**0.2254**	**0.2732**
Zero-shot-CoT	0.2991	0.0095	**0.2721**	0.2170	0.2654
Few-shot-CoT	0.2985	0.0103	0.2563	0.1958	0.2450
Flan-T5-780M					
Baseline	0.2045	0.0086	0.1346	0.1372	0.2165
Main theme	**0.2090**	**0.0102**	**0.1366**	**0.1379**	**0.2242**
Zero-shot-CoT	0.2026	0.0077	0.1364	0.1357	0.2124
Few-shot-CoT	0.2022	0.0073	0.1189	0.1229	0.1937

served when the main topic of the conversation was included in the prompt, but it also failed to surpass the baseline performance, and in some instances, it even performed slightly worse. Despite this overall under performance, a closer examination reveals that zero-shot CoT did lead to a marginal performance increase in the TEST_{CAT} split for both models (Table 2). This suggests that while the technique may not generally enhance model performance, it could have a specific benefit in dealing with unseen subcategories.

6.3 Few-shot CoT

Table 1 shows that the introduction of few-shot CoT, which involves adding examples with manually generated explanations for the correct actions, led to decreased performance in both Sheared-LlaMa and Flan-T5 models across all out-of-distribution splits and in-distribution splits. This reduction was evident even in the TEST_{CAT} split (Table 2), which previously showed improved results with zero-shot CoT, suggesting that few-shot CoT negates some benefits achieved by zero-shot techniques. Notably, while the IM score and EG IoU for both models were generally lower with few-shot CoT, they still hovered around baseline levels. However, the TG F1 scores exhibited a more pronounced decline relative to the baseline. This could suggest that few-shot CoT, by focusing on providing detailed example-based reasoning, may disrupt the models' ability to match and generate text responses (*say*) that align with the users *textinput*, impacting their performance in processing conversational directives and actions.

7 Discussion

7.1 Experimental findings

Our experiments demonstrate that integrating the main context of the conversation into the prompt enhances the performance of LLMs, specifically Sheared-LlaMa and Flan-T5, in conversational web navigation tasks. This approach provides the models with a richer understanding of the user's objectives, allowing for more accurate identification of intended actions, as seen by improved overall IM scores. Conversely, while zero-shot CoT shows some promise in generalizing to unseen subcategories, it does not improve overall model performance. The lack of improved performance could stem from our utilization of finetuned models, thereby constraining the impact of zero-shot CoT. Furthermore, incorporating few-shot CoT, which involves adding examples with manually generated explanations for the correct actions, results in reduced performance. This decline is particularly evident in the TG F1 score, suggesting that few-shot CoT's focus on detailed, example-based reasoning might interfere with the models' ability to correctly match and generate text responses (*say*) that align with user inputs (*textinput*), thereby impairing their ability to process conversational directives effectively.

One potential reason for the under performance with few-shot CoT could be its sensitivity to the design of task prompts. Few-shot CoT effectiveness can decline if the types of questions used in the prompt examples do not match the task questions (Kojima et al., 2022). Specifically, if the example prompts drawn from the validation set are markedly different from those in other testing splits, this disparity can hinder the model's capacity to generalize effectively from these examples. Additionally, providing a summary of the entire demo as part of few-shot CoT might be too broad and not sufficiently task-specific. Rather than summarizing a section of a demonstration, providing examples that include more specific user requests, intended actions, and corresponding reasoning, may enhance the model's ability to learn effectively. Lastly, the suboptimal results observed with zero-shot CoT and few-shot CoT might also be attributed to the use of smaller LLMs in our experiments. Larger LLMs are generally more adept at learning from CoT reasoning (Wei et al., 2023), suggesting that utilizing bigger models could potentially yield improved outcomes.

7.2 Limitations

The prompt length must stay within the context window size of the model used to generate actions, limiting our ability to include extensive history of the conversation or detailed examples for each demonstration. Consequently, only the most crucial information from the conversation history can be included in the prompts. Furthermore, examples designed to guide the LLM must omit the HTML elements from the original demonstration, potentially leading to a loss of valuable contextual information. This constraint on prompt length inherently limits the effectiveness of these techniques by restricting how comprehensively useful information can be encapsulated within the token limits.

In our experiments, we did not investigate the impact of varying the placement of *"Let's think step by step"* to come up with reasoning and use that reasoning for this task" or the few-shot CoT examples within the prompts. Different placements might yield different results (Liu et al., 2022).

8 Conclusions

We conducted three experimental enhancements to improve the prompting techniques used with LLMs on the WEBLINX benchmark for conversational web navigation. The first experiment involved identifying the main theme of the conversation. We extracted keywords from dialogues between users and navigators, using these keywords to pinpoint the most relevant user utterance, which was then incorporated as the main theme in the prompt. The second experiment introduced zero-shot CoT by appending the statement, *"Let's think step by step to come up with reasoning and use that reasoning for this task"*, to encourage more reasoned actions from the model. Lastly, in our third experiment, we applied few-shot CoT in an attempt to enhance the model's reasoned decision-making. For this, we manually curated examples for each demonstration in the validation set, that clearly defined the problem being addressed and included explanations for the correct actions, randomly choosing one demonstration to include in the prompt. We evaluated these three prompt changes on the Sheared-LLaMa-1.3B and FlanT5-780M models, fine-tuned on the WEBLINX benchmark.

We found that incorporating the main context of the conversation into the prompts substantially enhances the performance of LLMs in conversational web navigation tasks. This strategy improves the models' understanding of user intentions, resulting in more accurate action identification, as reflected in an improvement in the intent of the action models. Conversely, despite showing promise in addressing unseen subcategories, zero-shot CoT did not enhance overall model performance. Moreover, the integration of few-shot CoT, despite its detailed action explanations, led to a reduction in performance. This decline was largely caused by the model's inability to generate text responses that aligned with user inputs, suggesting that the complex reasoning required by few-shot CoT might interfere with the models' ability to produce accurate text responses, thereby diminishing their effectiveness in managing conversational directives.

Overall, while the integration of targeted prompting strategies shows promise in refining the performance of LLMs in conversational web navigation, the complexity of reasoning in few-shot CoT presents challenges that merit further investigation and optimization, particularly with larger LLMs. Our findings highlight the need for a nuanced approach to the design of prompts, considering both the capabilities and limitations of different model sizes. The incorporation of a rich conversational context proves beneficial, but the challenges faced by smaller LLMs indicate that the complexity and demands of advanced prompting methods require careful calibration to truly leverage their potential in enhancing model performance.

9 Future work

Our study has identified several potential directions for advancing the use of LLMs in conversational web navigation, specifically through innovative prompting techniques and the integration of advanced model architectures. The performance of few-shot CoT prompting was below expectations. Employing a pipeline of diverse models, and interspersing larger, more capable models like GPT-4 to generate reasoning and intermediate examples, could enrich the quality of outputs before arriving at the final decision. Further, expanding the use of few-shot CoT by incorporating multiple examples in a single prompt could be considered. However, limitations related to the context window size and the relevance of the example category must be addressed. It may be beneficial to explore more targeted example selection or to compress prompts more effectively to accommodate a greater variety of examples without exceeding token limits.

Incorporating the main context of the conversation in the prompt led to improvements in the model performance. We calculated the main context of the conversation by finding keywords throughout the conversation and using these keywords to select the most relevant user utterance. Incorporating historical context by calculating embeddings for past interactions using an encoder-based model and storing these in a vector database offers another frontier. A MIPS (Maximum Inner Product Search) supporting vector database could then be used to identify the closest thematic matches. This approach could be further enhanced by combining it with CoT reasoning techniques to provide a more nuanced understanding of past interactions and their implications for current decision-making processes.

Although this study focused on text-only models, some of our methods could inspire enhanced prompting techniques for image-only and multimodal models. For instance, just as we manually crafted summaries for demonstrations that included the correct action and its explanation for few-shot CoT prompting, one could similarly provide written summaries (i.e. figure captions) of the content displayed in screenshots for image-only and multimodal models. This approach could offer models with better contextual understanding and potentially elevate the generally weaker performance observed in image-only and multimodal models compared to text-only decoders.

These areas represent promising avenues for future research that could address some of the current limitations and enhance the capabilities of LLMs in processing conversational directives and actions more effectively. Each of these initiatives has the potential to push the boundaries of what conversational AI can achieve in practical applications.

10 Contribution

Both Vishvak and Aayush implemented the prompt changes, with Aayush taking a more prominent role in leading the implementation. Vishvak ran each model using the different prompt changes. Both authors contributed equally to the writing of the paper and the initial research idea and problem formulation.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. 2023. Memory matters: The need to improve long-term memory in llm-agents. *Proceedings of the AAAI Symposium Series*, 2:277–280.

Jihyoung Jang, Minseong Boo, and Hyounghun Kim. 2023. Conversation chronicles: Towards diverse temporal and relational dynamics in multi-session conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13584–13606, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv:2205.11916 [cs]*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2022. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv (Cornell University)*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents.

R Mccoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. 2023. Embodiedgpt: Vision-language pre-training via embodied chain of thought.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah B Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard J Powell, George , Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John W Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, M Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Michel Arthur, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhongying Gong, Daniel Toyama, Cyprien, Yujia Li, Tayfun Terzi, Vladimir Mikulik, I. Babuschkin, Aidan Clark, Diego , Aurelia Guy, Chris Jones, James T Bradbury, Matthew S Johnson, Blake A Hechtman, Laura Weidinger, Iason Gabriel, William M Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis insights from training gopher.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*.

Vishwas B. Sharma. rake-nltk: Python implementation of the rapid automatic keyword extraction algorithm using nltk.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zheng, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using deepspeed and megatron to train megatron-turing NLG 530b, A large-scale generative language model. *CoRR*, abs/2201.11990.

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. 2023. Llama 2: Open foundation and fine-tuned chat models.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv:2203.11171 [cs]*.

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. *arXiv:2201.11903 [cs]*.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. 2023. Larger language models do in-context learning differently. *arXiv:2303.03846 [cs]*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv (Cornell University)*.

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Boyuan Zheng, B Gou, Jihyung Kil, Hongjian Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv (Cornell University)*.

Appendix

A Additional Results Tables

To complement Section 6, we include the grouped results for each split: in-domain (TEST_{IID}), TEST_{CAT}, TEST_{WEB}, TEST_{GEO}, TEST_{VIS}.

Table 2: Element Group (EG), Text Group (TG) and overall results for TEST_{CAT} on Sheared-Llama-1.3B and Flan-T5-780M.

Model	**Overall Micro Avg**	**Overall IM**	**EG IoU**	**TG F1**
Sheared-Llama-1.3B				
Baseline	0.2401	0.2966	0.0112	0.2914
Main theme	0.2370	**0.3111**	**0.0161**	0.2906
Zero-shot-CoT	**0.2405**	0.2987	0.0116	**0.2915**
Few-shot-CoT	0.2045	0.3005	0.0106	0.2682
Flan-T5-780M				
Baseline	0.1151	0.1876	0.0011	0.1314
Main theme	0.1113	**0.1960**	**0.0061**	0.1306
Zero-shot-CoT	**0.1154**	0.1879	0.0010	**0.1323**
Few-shot-CoT	0.0895	0.1884	0.0012	0.1018

Table 3: Element Group (EG), Text Group (TG) and overall results for TEST_{WEB} on Sheared-Llama-1.3B and Flan-T5-780M.

Model	**Overall Micro Avg**	**Overall IM**	**EG IoU**	**TG F1**
Sheared-Llama-1.3B				
Baseline	0.2353	0.3071	**0.0142**	**0.2997**
Main theme	**0.2421**	**0.3174**	0.0120	0.2984
Zero-shot-CoT	0.2338	0.3105	0.0084	0.3004
Few-shot-CoT	0.1998	0.3066	0.0116	0.2735
Flan-T5-780M				
Baseline	0.1553	0.2337	0.0122	0.1236
Main theme	**0.1631**	**0.2440**	**0.0130**	**0.1284**
Zero-shot-CoT	0.1575	0.2358	0.0119	0.1275
Few-shot-CoT	0.1474	0.2313	0.0108	0.0924

Table 4: Element Group (EG), Text Group (TG) and overall results for TEST_{GEO} on Sheared-Llama-1.3B and Flan-T5-780M.

Model	**Overall Micro Avg**	**Overall IM**	**EG IoU**	**TG F1**
Sheared-Llama-1.3B				
Baseline	0.2309	0.2684	0.0036	0.2817
Main theme	**0.2361**	**0.2861**	**0.0041**	0.2774
Zero-shot-CoT	0.2203	0.2604	0.0003	**0.2928**
Few-shot-CoT	0.2076	0.2633	0.0027	0.2783
Flan-T5-780M				
Baseline	0.1309	0.1676	0.0024	0.1797
Main theme	**0.1341**	**0.1681**	**0.0041**	**0.1832**
Zero-shot-CoT	0.1276	0.1604	0.0008	0.1812
Few-shot-CoT	0.1142	0.1642	0.0005	0.1756

Table 5: Element Group (EG), Text Group (TG) and overall results for TEST_{VIS} on Sheared-Llama-1.3B and Flan-T5-780M.

Model	**Overall Micro Avg**	**Overall IM**	**EG IoU**	**TG F1**
Sheared-Llama-1.3B				
Baseline	0.1786	0.3291	0.0202	0.2045
Main theme	**0.1864**	**0.3481**	**0.0307**	0.2022
Zero-shot-CoT	0.1732	0.3269	0.0178	0.2036
Few-shot-CoT	0.1714	0.3239	0.0164	**0.2052**
Flan-T5-780M				
Baseline	**0.1437**	**0.2291**	**0.0189**	0.1037
Main theme	0.1431	0.2280	0.0176	0.1042
Zero-shot-CoT	0.1422	0.2262	0.0172	**0.1046**
Few-shot-CoT	0.1404	0.2247	0.0167	0.1045

Table 6: Element Group (EG), Text Group (TG) and overall results for TEST$_{\text{IID}}$ on Sheared-Llama-1.3B and Flan-T5-780M.

Model	**Overall Micro Avg**	**Overall IM**	**EG IoU**	**TG F1**
Sheared-Llama-1.3B				
Baseline	0.2612	0.3137	0.0	0.3143
Main theme	**0.2732**	**0.3185**	**0.0427**	0.3047
Zero-shot-CoT	0.2654	0.3180	0.0	**0.3192**
Few-shot-CoT	0.2450	0.3147	0.0081	0.2921
Flan-T5-780M				
Baseline	0.2165	0.2709	0.0134	0.1756
Main theme	**0.2242**	**0.2868**	**0.0142**	**0.1767**
Zero-shot-CoT	0.2124	0.2647	0.0126	0.1748
Few-shot-CoT	0.1937	0.2635	0.0115	0.1596

B Dataset Supplementary Statistics

In Section 4, we introduced the WEBLINX dataset. Here, we offer additional statistics for readers seeking a more comprehensive understanding of the dataset. Table 7 provides a description of demos in each split. Table 8 presents a breakdown of the number of demonstrations for each split and the average number of turns. Most demonstrations consist of 40-50 turns, except for the TEST$_{\text{VIS}}$ split, which is notably lower due to the absence of follow-up questions or remarks related to on-screen events (Lù et al., 2024). Table 9 describes the complete action space evaluated in WEBLINX.

Table 7: Demonstration (Demo) splits for evaluation (Lù et al., 2024).

Split	**Description**
TEST$_{\text{IID}}$	In-domain demos to test in-domain generalization
TEST$_{\text{OOD}}$	Aggregation of splits for OOD evaluation
TEST$_{\text{WEB}}$	Unseen websites from the same subcategories
TEST$_{\text{CAT}}$	New subcategories within the same categories
TEST$_{\text{GEO}}$	Geographic locations not in TRAIN
TEST$_{\text{VIS}}$	Instructor does not see the screen

Table 8: Turn level stats by splits (Lù et al., 2024).

Split	# Demos	μ turns	σ turns	Active	Total
VALID	100	40.76	14.51	1717	4076
TEST$_{\text{IID}}$	100	43.18	16.08	1846	4318
TEST$_{\text{CAT}}$	223	45.30	25.43	4979	10102
TEST$_{\text{WEB}}$	211	40.47	18.17	4184	8540
TEST$_{\text{VIS}}$	444	36.05	20.09	7725	16006
TEST$_{\text{GEO}}$	290	48.05	18.66	6141	13934

Table 9: Complete list of WEBLINX action space (Lù et al., 2024).

Action	Description
say(speaker, text)	talking to instructor or navigator
click(element)	click on an element
click(x, y)	click on the coordinates mapping to an element
hover(element)	hover over an element
hover(x, y)	hover over the coordinates mapping to an element
textinput(element, value)	type text into the element
change(element, text)	change the value of the element to another option
load(url)	load the URL of a new webpage
submit(element)	submit the form
scroll(x, y)	scroll to the coordinates
copy(element, text)	copy the text from the element
paste(element, text)	paste the text into the element
tabCreate()	create a new tab
tabRemove(tabId)	remove the tab
tabSwitch(tabIdFrom, tabIdTo)	switch between tabs

C Input Prompt Template

We provide the templates for chat-based models like Sheared-LLaMA-1.3B and and for the instruction-based models like Flan-T5-780M. We show prompt templates for each of the three prompt experiments we conducted.

C.1 Template Prompt for Chat-based Models (Sheared-LLaMA)

```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{{PAST ACTIONS}}
{END REPEAT}
Please select the best action using the correct format, do not provide any other information or explanation.
```

Figure 1: Chat-based models baseline prompt template

```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Theme: {{MAIN THEME OF CONVERSATION}}
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{{PAST ACTIONS}}
{END REPEAT}
Please select the best action using the correct format, do not provide any other information or explanation.
```

Figure 2: Chat-based models prompt template with main theme of conversation included

```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Let's think step by step to come up with a reasoning and use that reasoning for this task.
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{{PAST ACTIONS}}
{END REPEAT}
Please select the best action using the correct format, do not provide any other information or explanation.
```

Figure 3: Chat-based models prompt template for zero-shot CoT

```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Here is an example: {{FEW-SHOT EXAMPLE}}
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{{PAST ACTIONS}}
{END REPEAT}
Please select the best action using the correct format, do not provide any other information or explanation.
```

Figure 4: Chat-based models prompt template with few-shot CoT example

C.2 Template Prompt for Instruction-based Models (Flan-T5)

```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{END REPEAT}

{REPEAT W-1 TIMES}
User: {{PAST ACTION BY USER}}
Assistant: {{PAST ACTION BY ASSISTANT}}
{END REPEAT}

USER: {{LAST ACTION BY USER}} Please select the best action using the correct format, do not provide any other information or explanation
Assistant:
```

Figure 5: Instruction-based models baseline prompt template

```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Theme: {{MAIN THEME OF CONVERSATION}}
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{END REPEAT}

{REPEAT W-1 TIMES}
User: {{PAST ACTION BY USER}}
Assistant: {{PAST ACTION BY ASSISTANT}}
{END REPEAT}

USER: {{LAST ACTION BY USER}} Please select the best action using the correct format, do not provide any other information or explanation
Assistant:
```

Figure 6: Instruction-based models prompt template with main theme of conversation included

```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Let's think step by step to come up with a reasoning and use that reasoning for this task.
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{END REPEAT}

{REPEAT W-1 TIMES}
User: {{PAST ACTION BY USER}}
Assistant: {{PAST ACTION BY ASSISTANT}}
{END REPEAT}

USER: {{LAST ACTION BY USER}} Please select the best action using the correct format, do not provide any other information or explanation
Assistant:
```

Figure 7: Instruction-based models prompt template for zero-shot CoT

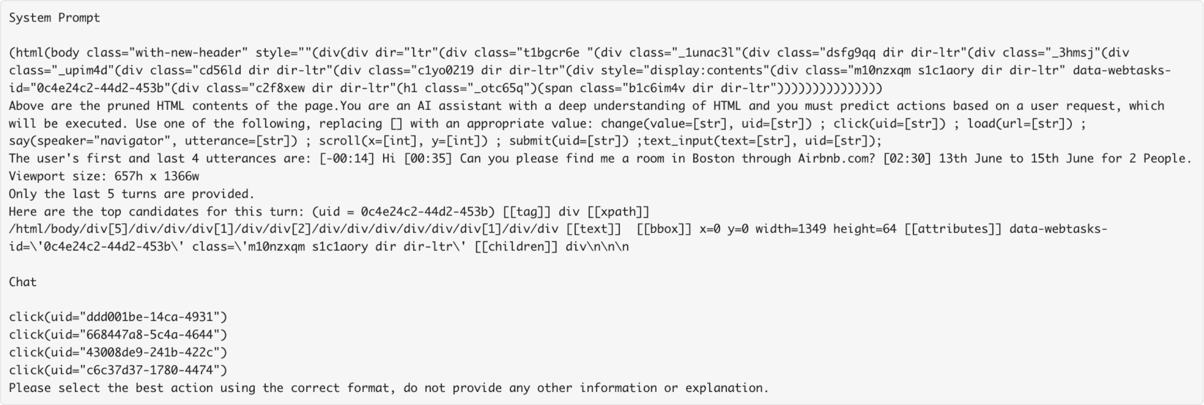
```
{{HTML REPRESENTATIONS}}

Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions
    → based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value:
    → change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int],
    → y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]) ;
The user's first and last 4 utterances are: {{PAST UTTERANCES}};
Viewport size: {{HEIGHT}}h x {{WIDTH}}w ;
Only the last {{W}} turns are provided.
Here is an example: {{FEW-SHOT EXAMPLE}}
Here are the top candidates for this turn: {REPEAT 10 TIMES}
(uid=...) [[tag]] ... [[xpath]] ... [[bbox]] x=X y=Y width=W height=H [[attributes]] a=val1 ... [[children]] {{TAG}}
{END REPEAT}

{REPEAT W-1 TIMES}
User: {{PAST ACTION BY USER}}
Assistant: {{PAST ACTION BY ASSISTANT}}
{END REPEAT}

USER: {{LAST ACTION BY USER}} Please select the best action using the correct format, do not provide any other information or explanation
Assistant:
```

Figure 8: Instruction-based models prompt template with few-shot CoT example

D Input Prompt Samples

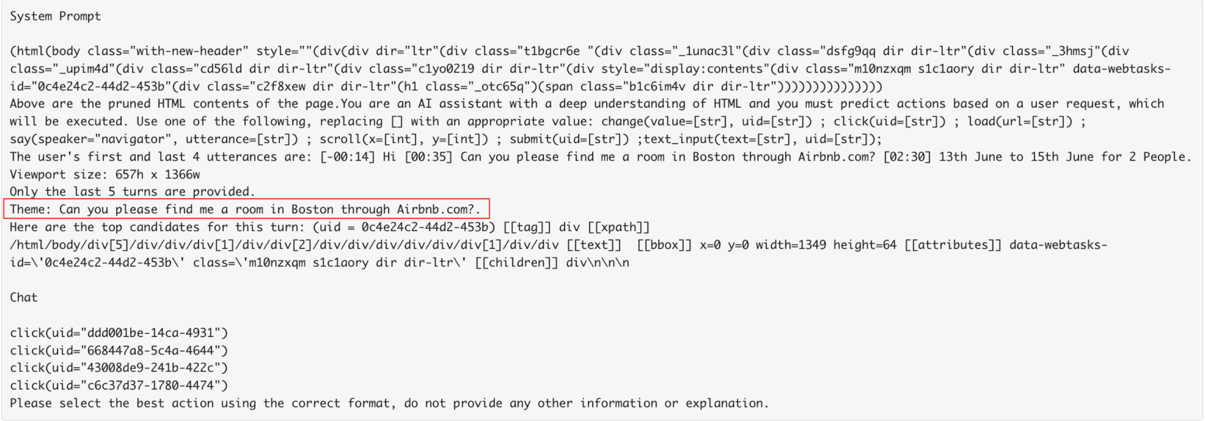
We provide prompt samples for chat-based models like Sheared-LLaMA-1.3B and and for the instruction-based models like Flan-T5-780M. We show sample prompts for each of the three prompt experiments we conducted.

D.1 Sample Prompt Input for Chat-based Models (Sheared-LLaMA)



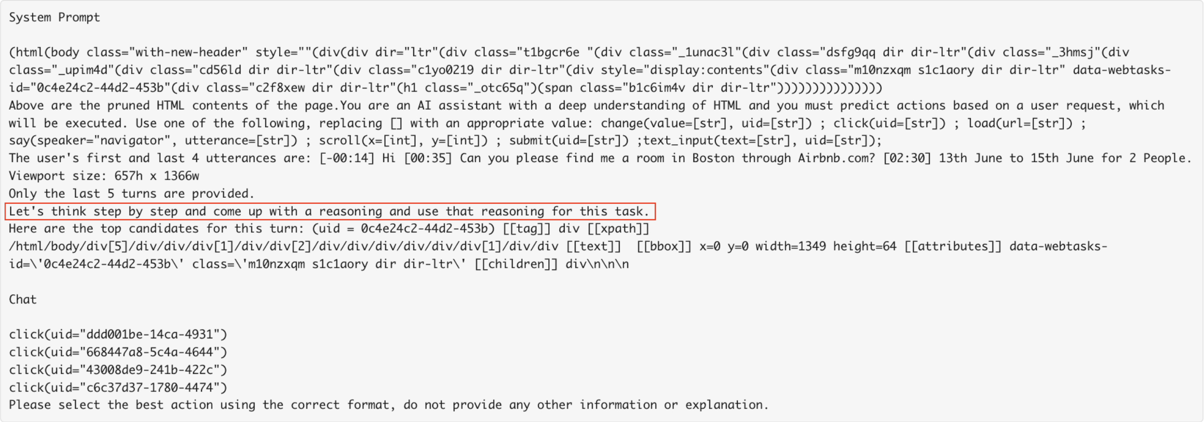
```
System Prompt

(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div
class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-
id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr")))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which
will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ;
say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-
id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Chat

click(uid="ddd001be-14ca-4931")
click(uid="668447a8-5c4a-4644")
click(uid="43008de9-241b-422c")
click(uid="c6c37d37-1780-4474")
Please select the best action using the correct format, do not provide any other information or explanation.
```

Figure 9: Chat-based models baseline sample prompt



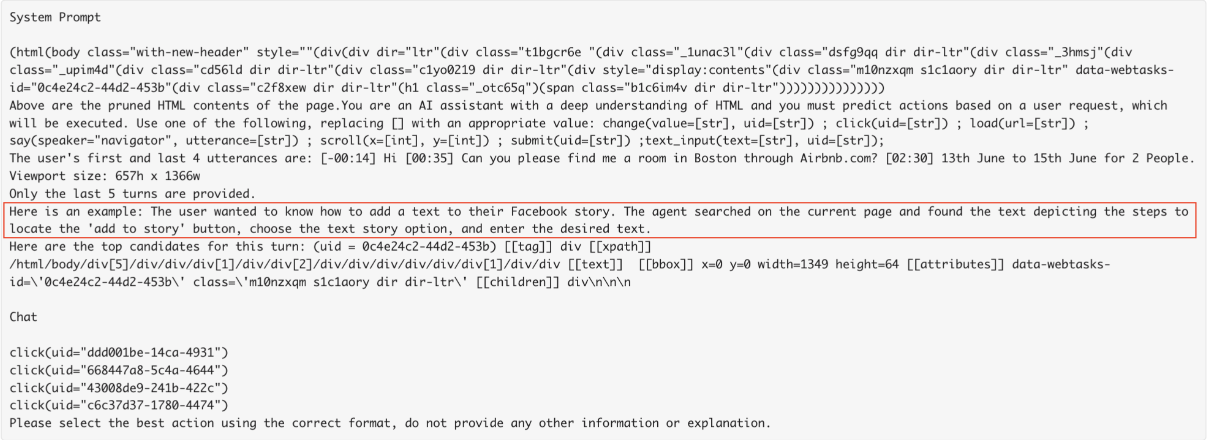
```
System Prompt

(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div
class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-
id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr")))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which
will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ;
say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Theme: Can you please find me a room in Boston through Airbnb.com?.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-
id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Chat

click(uid="ddd001be-14ca-4931")
click(uid="668447a8-5c4a-4644")
click(uid="43008de9-241b-422c")
click(uid="c6c37d37-1780-4474")
Please select the best action using the correct format, do not provide any other information or explanation.
```

Figure 10: Chat-based models sample prompt with main theme of conversation included



```
System Prompt

(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr")))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Let's think step by step and come up with a reasoning and use that reasoning for this task.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Chat

click(uid="ddd001be-14ca-4931")
click(uid="668447a8-5c4a-4644")
click(uid="43008de9-241b-422c")
click(uid="c6c37d37-1780-4474")
Please select the best action using the correct format, do not provide any other information or explanation.
```

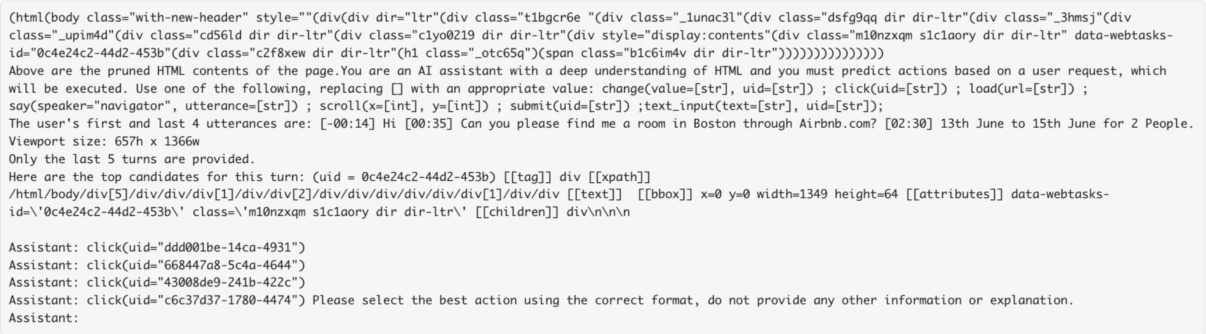
Figure 11: Chat-based models sample prompt for zero-shot CoT



```
System Prompt

(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr")))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ; say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Here is an example: The user wanted to know how to add a text to their Facebook story. The agent searched on the current page and found the text depicting the steps to locate the 'add to story' button, choose the text story option, and enter the desired text.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Chat

click(uid="ddd001be-14ca-4931")
click(uid="668447a8-5c4a-4644")
click(uid="43008de9-241b-422c")
click(uid="c6c37d37-1780-4474")
Please select the best action using the correct format, do not provide any other information or explanation.
```

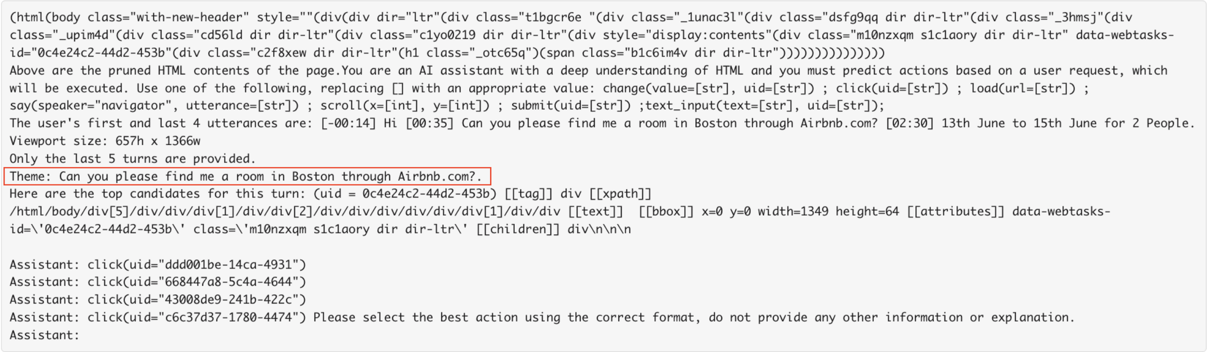
Figure 12: Chat-based models sample prompt with few-shot CoT example

D.2 Sample Prompt Input for Instruction-based Models (Flan-T5)



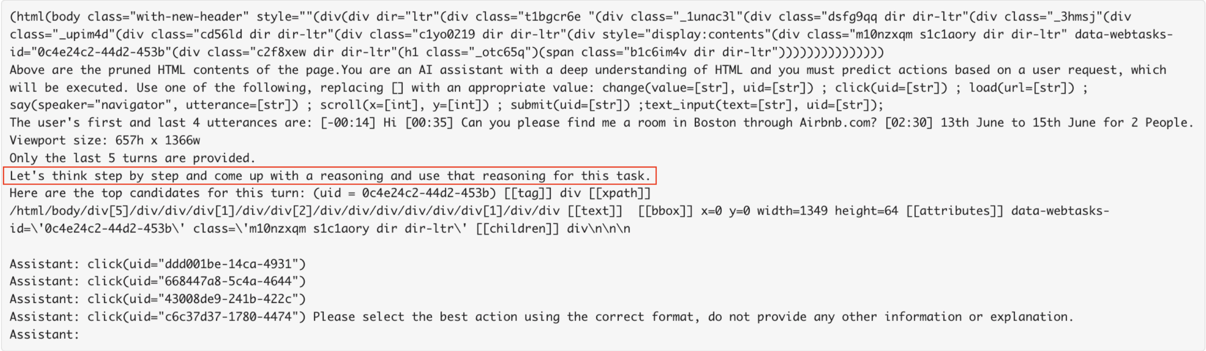
```
(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div
class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-
id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr")))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which
will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ;
say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-
id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Assistant: click(uid="ddd001be-14ca-4931")
Assistant: click(uid="668447a8-5c4a-4644")
Assistant: click(uid="43008de9-241b-422c")
Assistant: click(uid="c6c37d37-1780-4474") Please select the best action using the correct format, do not provide any other information or explanation.
Assistant:
```

Figure 13: Instruction-based models baseline sample prompt



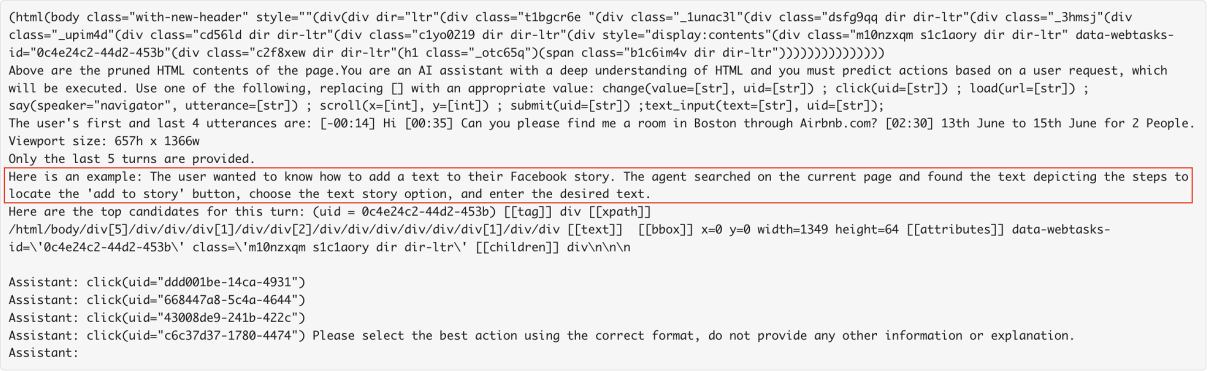
```
(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div
class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-
id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr")))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which
will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ;
say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Theme: Can you please find me a room in Boston through Airbnb.com?.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-
id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Assistant: click(uid="ddd001be-14ca-4931")
Assistant: click(uid="668447a8-5c4a-4644")
Assistant: click(uid="43008de9-241b-422c")
Assistant: click(uid="c6c37d37-1780-4474") Please select the best action using the correct format, do not provide any other information or explanation.
Assistant:
```

Figure 14: Instruction-based models sample prompt with main theme of conversation included



```
(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div
class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-
id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr")))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which
will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ;
say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Let's think step by step and come up with a reasoning and use that reasoning for this task.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-
id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Assistant: click(uid="ddd001be-14ca-4931")
Assistant: click(uid="668447a8-5c4a-4644")
Assistant: click(uid="43008de9-241b-422c")
Assistant: click(uid="c6c37d37-1780-4474") Please select the best action using the correct format, do not provide any other information or explanation.
Assistant:
```

Figure 15: Instruction-based models sample prompt for zero-shot CoT



```
(html(body class="with-new-header" style=""(div(div dir="ltr"(div class="t1bgcr6e "(div class="_1unac3l"(div class="dsfg9qq dir dir-ltr"(div class="_3hmsj"(div
class="_upim4d"(div class="cd56ld dir dir-ltr"(div class="c1yo0219 dir dir-ltr"(div style="display:contents"(div class="m10nzxqm s1c1aory dir dir-ltr" data-webtasks-
id="0c4e24c2-44d2-453b"(div class="c2f8xew dir dir-ltr"(h1 class="_otc65q")(span class="b1c6im4v dir dir-ltr"))))))))))))))
Above are the pruned HTML contents of the page.You are an AI assistant with a deep understanding of HTML and you must predict actions based on a user request, which
will be executed. Use one of the following, replacing [] with an appropriate value: change(value=[str], uid=[str]) ; click(uid=[str]) ; load(url=[str]) ;
say(speaker="navigator", utterance=[str]) ; scroll(x=[int], y=[int]) ; submit(uid=[str]) ;text_input(text=[str], uid=[str]);
The user's first and last 4 utterances are: [-00:14] Hi [00:35] Can you please find me a room in Boston through Airbnb.com? [02:30] 13th June to 15th June for 2 People.
Viewport size: 657h x 1366w
Only the last 5 turns are provided.
Here is an example: The user wanted to know how to add a text to their Facebook story. The agent searched on the current page and found the text depicting the steps to
locate the 'add to story' button, choose the text story option, and enter the desired text.
Here are the top candidates for this turn: (uid = 0c4e24c2-44d2-453b) [[tag]] div [[xpath]]
/html/body/div[5]/div/div/div[1]/div/div[2]/div/div/div/div/div/div[1]/div/div [[text]]  [[bbox]] x=0 y=0 width=1349 height=64 [[attributes]] data-webtasks-
id=\'0c4e24c2-44d2-453b\' class=\'m10nzxqm s1c1aory dir dir-ltr\' [[children]] div\n\n\n

Assistant: click(uid="ddd001be-14ca-4931")
Assistant: click(uid="668447a8-5c4a-4644")
Assistant: click(uid="43008de9-241b-422c")
Assistant: click(uid="c6c37d37-1780-4474") Please select the best action using the correct format, do not provide any other information or explanation.
Assistant:
```

Figure 16: Instruction-based models sample prompt with few-shot CoT example

E Summary of Other Exploratory Attempts

E.1 Training infra issues and setup

- *Dataset retrieval*: Complete dataset download for WEBLINX was failing midway. It was tried about 10 times with the same issue. The job would stop while the download was in progress. The reason being there was no response from huggingface for the requested resource. We solved it by triaging the item for which this was occurring and manually downloading it. This needed to be done repeatedly across multiple jobs. **Error was:** *raise HfHubHTTPError(str(e), response=response) from huggingface_hub.utils.errors.HfHubHTTPError: 500 Server Error: Internal Server Error for url: url_omitted*

- The training performance was coming out to be too slow for the data-size we had. Managing data within the quota allocated on the home directory in Compute Canada and accessing it based on file permissions was problematic too.

E.2 Embedding issue for multiple calls

We were trying to use the *MiniLM* model from SentenceTransformers to generate embeddings which we could use to pick the evolving main theme among the utterances. However, while trying to make it as part of the pipeline for building prompts, we were running into out of memory issues because the same object for the model was being passed to encode different dialogues across recursive calls. This made us go with a different approach to pick the main theme and we used the embeddings approach in a modular manner across single call separately.

E.3 Explanation why we were not able to replicate baseline paper results

We used regular floating point (FP16) during our evaluation instead of BF16 due to GPU compatibility issues. Both the models we used were of tensor type BF16. FP16 has 5 bits with the possible encoding range between $-65,000$ to $65,000$. Meanwhile BF16 can maintain the 32-bit number range, while having memory footprint similar to FP16. It can represent values up to 3.4 x 10^{38}. This means when we loaded the BF16 models using FP16, we would have lost the values by round off which were $> 65,000$ or $< -65,000$. This means the model would have incurred some loss of information which might explain why we were not able to replicate the exact same values for the baselines.

E.4 Experiments Details

- Hyperparameters: For each of the split we ran the evaluation, following parameters were maintained. The exhaustive list can be found in the github repository under config folder for the respective model. Top-10 candidates from the page were considered by the model to pick the next course of action. The batch size per device was set to 2.

- Software details: Due to dependency issues we encountered, we ended up using specific versions of the following packages to run our experiements. (torch==2.0.1 , datasets==2.14.0, cuda 11.7, huggingface-hub==0.17.0, tokenizers==0.14.1, transformers==4.35.0)

- Following modules were sideloaded on the compute canada cluster because we did not have sudo access on the nodes. (StdEnv/2020 gcc/9.3.0 arrow/13.0.0 rust/1.70.0 python/3.10.2 cudacore/.11.7.0)

- Implementation details: We used regular floating point instead of bf16 as used by the WEBLINX package. It would have led to increased memory usage but we did so because of dependency install issues.

- Our GPUs were not compatible with flash_attention_2. This would have led to reduced efficiency and longer execution times.

- Link to code repository : `https://github.com/kpraays/PrompGent`