Institute for Visualization and Interactive Systems

University of Stuttgart
Pfaffenwaldring 5a
D–70569 Stuttgart, Germany

Masterarbeit Nr. 3572988

Leveraging Large Language Models for Latent Intention Recognition and Next Action Prediction

Mohamed Ahmed

Course of Study:	Computer Science
Examiner:	Prof. Dr. Andreas Bulling
Supervisor:	Guanhua Zhang, M.Sc.
Commenced:	Febraury 15, 2024
Completed:	August 15, 2024

Abstract

Autonomous agents that operate within graphical user interfaces (GUIs) have a significant potential to improve user experience. To achieve this, such agents must be customized and proactive. Understanding user intentions through their interactions and engagements with GUIs enables these agents to better fulfill user needs. This work introduces a novel LLM-based framework, Mistral-Intention, that accurately recognizes latent user intentions from their interactions. A key innovation is the integration of a sub-goal generation step, using prompt engineering to decompose user tasks into actionable steps, enhancing the model's interpretative capabilities and extendability. Furthermore, the incorporation of a keyword extraction-based loss significantly refines the model's focus on critical information of user actions such as typed values, ensuring comprehensive and relevant intention recognition. We evaluate Mistral-Intention using a range of metrics, including manual metrics and automatic methods based on GPT-4o, against a modified version of the state-of-the-art task automation framework, namely SYNAPSE. Results from extensive testing on the MIND2WEB and MoTIF datasets highlight Mistral-Intention's superior performance in intention recognition across various GUI environments. Furthermore, we implement an LLM-based computer agent capable of predicting the user's next action. We have addressed the challenges faced while developing such agents, such as the limited context window, and understanding the current GUI environment. Our LLM-based agent exhibits an improvement of 15.30% in the element accuracy and 13.20% in operation f1 over the previous state-of-the-art method in MindAct on MIND2WEB. Our work not only pushes the boundaries of computational HCI but also opens new pathways for developing more intuitive and effective user-center interaction solutions.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topic Introduction

The study of intention recognition has become a prominent research area, spanning diverse fields such as robotics [65], autonomous systems [49], and the automotive industry [33]. This multidisciplinary interest is attributed to the key role of accurate intention recognition in enhancing the anticipatory capabilities of intelligent systems. This accuracy is crucial for systems that interact in real time with their environment, making them more responsive and efficient.

In the field of human-computer interaction (HCI), intention recognition is increasingly acknowledged for its potential to revolutionize user experience. It aims to smooth the interaction between humans and machines, significantly reducing user effort [78]. Additionally, it enables the automation of task execution without explicit human instruction, thereby simplifying complex workflows and boosting overall efficiency. These advancements highlight the importance of intention prediction in the evolution of interactive and intelligent systems.

The implementation of autonomous agents that interact with graphical user interfaces (GUIs) to complete tasks has garnered significant interest [22, 26]. They can interpret natural language descriptions and iteratively interact with interfaces. Enhancing these agents' capacity to understand user actions and infer goals can greatly improve their utility, offering personalized and effective assistance. For example, when booking flight tickets online, an ideal agent would not only facilitate the booking but also proactively suggest related tasks like booking a hotel and adding dates to a calendar.

This level of assistance requires a deep understanding of the underlying goals that drive user interactions. By accurately identifying these goals, we can augment the agent's memory, thereby enhancing its personalization capabilities and increasing its utility for future tasks. This thesis work distinctively tackles the challenge of recognizing latent user intention from UI interactions, a subject that has not been thoroughly investigated in the HCI community.

1.2 Motivation

Our work is driven by the enhanced user experience and the advanced intelligence capabilities offered by automation agents. This has inspired us to investigate the recognition of latent intentions using a sequence-to-sequence (Seq2Seq) framework. By interpreting the previous interactions and understanding the underlying behavior of the user, we aim to summarize such behavior to ease downstream tasks such as personalized recommendation systems. Achieving this goal would have significant implications for task automation agents, allowing for more accurate, independent, and reliable next-action prediction.

This task, which involves understanding and summarizing human behavior from previous interactions falls under the umbrella of Seq2Seq tasks. Seq2Seq tasks have been effectively managed by recurrent neural networks, such as Long Short-Term Memory (LSTM) units [25]. The main limitation of the LSTM-based methods is that they suffer from long sequences and they are hard to be trained [48]. Furthermore, they lack generalization on different environments and intentions. The introduction of transformers and the attention mechanism [63], which established the way for the development of Large Language Models (LLMs), has shown superior effectiveness and managing such tasks. Despite the transition from LSTM to transformers, to the best of our knowledge, the integration of LLMs into latent intention recognition within GUI environments remains a relatively under-explored area of research.

A potential application of intention prediction is the development of LLM-based task automation agents. These agents are designed to understand the current GUI environment, the intended task, and the sequence of actions performed, subsequently predicting the next action to fulfill the intended task. However, these agents face significant challenges, including the limited context length of the LLM and the complexity of recognizing the structured environment they operate within.

Motivated by this research gap, we aim to develop an LLM-based framework for translating human interactions into a textual format that captures the underlying goal of such actions This Seq2Seq2 approach aims to overcome the limitations of the other approaches. Successfully achieving this objective would have a significant impact on the development of automation agents, recommendation systems, and the detection of unintended errors, thereby equipping these systems with advanced intelligence capabilities. Furthermore, we present an LLM-based agent that address the previously mentioned challenges.

1.3 Key novelty and Contributions

Our main contributions are as follows:

- Implementation of the first LLM-based pipeline to summarize user behavior from a broad range of intentions in everyday web-based tasks.
- Introduction of a keyword-based loss to prevent our LLM from over-summarization and ignoring important words.
- Adoption of a task automation frameworks, SYNAPSE [80], as a comparative baseline for the intention summarization.
- Employing comprehensive evaluations using both manual and automated protocols validate the effectiveness and reliability of the proposed solutions.
- Developing an LLM-based pipeline for the next action prediction task as an application of the summarized intentions.

2 Related Work

Artificial intelligence (AI) has revolutionized a multitude of fields, ranging from natural language processing to the different challenges of computer vision. The research area has dedicated efforts to develop models that can enhance human capabilities across various applications. Moreover, AI advances lead to automating some tasks and eliminating any human interaction or effort to perform such tasks. Among these, computer agents have been significant advancements, with large language models (LLMs) demonstrating exceptional capabilities.

LLMs have experienced substantial development and are employed in diverse fields from finance [70] and education [64] to human-computer interaction (HCI) [80]. The ability of these networks to understand and generate texts nearly indistinguishable from human-created ones has opened up possibilities to build complex text-based applications.

In parallel, the field of HCI has also benefited from the advancements in AI intense learning (DL). Techniques in task automation agents, recommendation systems, and human behavior modeling have become increasingly important in the field of HCI. These methods are crucial in different aspects of HCI, such as improving user experience (UX) and automating repetitive tasks to increase productivity.

2.1 Interaction Behaviour Modeling

Modeling human behavior on an interaction task has been studied heavily in multiple fields such as social science [30, 72], robotics [46], the medical field [11], and in HCI [45]. Understanding human behavior is fundamentally part of the quest of AI for computationally modeling human intelligence. In HCI, these models can assist the UX designer in determining how reliable and usable an interface is without testing it with real users, which is expensive. Different classical purely analytical methods such as Fitts' law [18] and Hick's law [24] presented robust estimates of human performance on tasks such as motor control and decision-making, they are incapable to capture the whole set of factors that may come to play in day-to-day tasks. Difference works tried to extend these methods, however, it is not a trivial task to accommodate factors that are not easy to measure.

In the next sections, we will discuss previous works modeling interactive behavior using different DL approaches. Moreover, we will dive into using LLM in behavior modeling in different fields such as social science and HCI.

2.1.1 Deep Learning for Interaction Behaviour Modeling

Thanks to the recent advancements in the DL community, researchers have started to implement a DL-based data-driven methodology for modeling interactive behaviors that can overcome the limitations of the analytical methods.

To address the challenge of creating robots capable of dynamic, non-verbal communication through gestures, which are often ambiguous and multimodal, GAN-C [46] explored the use of Generative Adversarial Networks (GANs) for modeling human interactive behavior during dialogues, incorporating factors such as interaction intensity, time evolution, and time resolution into the network structure. Another robotic application is to enables robots to predict human future motion and adapt their actions accordingly for safe and efficient collaboration in a shared workspace, such an application requires the robots to understand the interactive behavior within the surrounding environment [61].

In the HCI community, Li et al. presented a deep recurrent neural network architecture, namely LSTM [25], to model and predict human performance in performing a sequence of UI tasks, specifically, target selection from a vertical list or menu [35]. By utilizing LSTM, they could successfully capture various factors that affect the UI task process. Such factors are not only what humans perceive at the moment but also what they learn from the past. Different groups of researchers have attempted to address other aspects of human behaviors on user interfaces, such as predicting human perception of UI interactivity. TapShoe [60] presented an approach to improve mobile interface usability by predicting the tappability of interface elements by training a deep neural network based on visual, spatial, and semantic features of the elements. The study highlighted the importance of understanding and modeling human behavior in the context of mobile interface design to enhance usability and user experience. Other works focused on user identification in Augmented Reality (AR) and Virtual Reality (VR), Liebers et al. investigated the usability of using hand-tracking data to model human behavior for implicit user identification, instead of using vulnerable methods such as entering username and password [37]. Furthermore, Mouse2Vec [79] has introduced a novel self-supervised method for mouse behavior modeling. it utilizes a transformer-based encoder-decoder architecture to capture the temporal, spatial, and semantic information in mouse trajectories. It was trained on two large-scale datasets, Buffalo [32] and EMAKI [57], and detected mouse events while reconstructing the input data. The learned representations were shown to identify interpretable mouse behavior

clusters and retrieve similar mouse trajectories. Such representations are also beneficial as effective features for downstream tasks, such as user identification [12], next activity prediction [19], and interactive task recognition [17].

2.1.2 Large Language Models for Behaviour Modeling

In recent years, there has been an increase in the development and application of LLMs because of their remarkable abilities in reasoning, planning, and decision-making. These advanced models have found significant use in human behavior modeling, where they are employed to understand and predict human actions and interactions. For example, Social-LLM [30] integrated LLMs with social network data to model user behavior. It combined user-generated content, such as profile description and metadata, with network interactions like retweets and mentions to enhance user detection tasks. The study highlighted the LLM's robustness and efficiency in capturing complex social dynamics, providing a valuable tool for computational social science research.

Other researchers used LLMs to understand and simulate human trust. Specifically, Xie et al. focused on the application of LLMs in Trust Games, a method widely used in behavioral economics to measure trust [72]. The researchers utilized the Belief-Desire-Intention (BDI) framework [5, 53] to model the reasoning processes of LLMs and assessed their trust behaviors compared to humans. Their findings indicated that LLM agents, particularly GPT-4 [1], exhibit trust behaviors that closely align with human behaviors. The results suggest that LLMs can model human trust behaviors, which has significant implications for human-agent interaction.

Moreover, there also exist some works that combine users' embeddings with LLMs to better understand user's behavior. Namely, USER-LLM [45], addresses the challenge of effectively integrating complex and potentially noisy user interaction data into LLMs. USER-LLM generates user embeddings through a Transformer-based encoder using self-supervised pretraining, which captures latent user preferences and their evolution over time. These embeddings are then integrated with LLMs via cross-attention and soft-prompting techniques, allowing the models to dynamically adapt to user contexts. Furthermore, USER-LLM maintains the original LLM's knowledge while providing personalized responses, making it an effective solution for user behavior modeling and personalization in various applications.

While LLMs have been extensively utilized in various human behavior modeling paradigms, we observe a notable gap in their application within the HCI community. To the best of our knowledge, their potential remains underexplored in this domain. In this work, we aim to bridge this gap by addressing the usage of LLMs for modeling interactive behavior in GUI environments.

2.2 Intention Recognition from Behaviour

Identifying human intention from their past interaction with the surrounding environment has become a prominent research area, spanning diverse fields such as robotics [65], autonomous system [49], and the automotive industry [33]. This multidisciplinary interest is attributed to the key role of accurate intention prediction in enhancing the anticipatory capabilities of intelligent systems. This accuracy is crucial for systems that interact in real-time with their environment, making them more responsive and efficient. In HCI, intention prediction is increasingly acknowledged for its potential to revolutionize user experience. It aims to smoothen the interaction between humans and computers, significantly reducing user efforts [78]. Additionally, it enables the automation of task execution without explicit human instruction, thereby simplifying complex workflows and boosting overall efficiency. These advancements highlight the importance of intention prediction in the evolution of interactive and intelligent systems.

2.2.1 Deep Learning for Intention Recognition

Studies in different fields have adapted DL approaches for intention recognition tasks. For example, Yan et al. presented a method for recognizing human intentions in human-robot collaboration scenarios [74]. They developed a deep LSTM to recognize human intentions based on the sequence of human motions captured by a camera, achieving high prediction accuracy, even when using only 40% of the motion sequences. Philips et al. presented a study on predicting the intentions of human drivers at intersections to aid in the development of autonomous vehicles. [50]. Inspired by previous works, they introduced an LSTM model, which outperformed the existing model in predicting whether a driver will turn left, turn right, or continue straight up to 150 meters before reaching an intersection. Other works have decided to include another modality such as eye gaze gestures. For instance, Prajod et al. proposed a novel approach using a convolutional neural network (CNN) to improve human-robot collaboration by recognizing human attention through gaze tracking [52]. The study presented an assembly scenario where a human operator and a collaborative robot (cobot) worked together to assemble a gearbox.

Moreover, the field of HCI participated in the study of intention recognition. Many works utilized the action sequences generated from mouse and keyboard input. For instance, Argwal et al. [2] presented a method based on LSTM that processes inputs from both mouse and keyboard to predict the user intent to buy an item from an e-commerce platform. Other works have included gaze dynamics as their source of information. For example, Bendarik et al. [7] introduced a method for predicting human intentions to distinguish between intentional and non-intentional button presses in the puzzle.

Furthermore, Soleymani et al. [55] implemented a method to automatically recognize user intent during image search sessions using multimodal data. The data was collected from 51 participants, including facial expressions, physiological responses, eye gaze, and implicit user interactions. The results showed that eye gaze and implicit user interactions, such as mouse movements and keystrokes, were the most informative features. Inspired by this finding, Zhang et al. [78] investigated the task of predicting user intents from mouse and keyboard inputs as well as gaze behavior during text formatting activities. They implemented dual-stream classifiers, one for action sequences and one for gaze features. These classifiers were trained using random forest and logistic regression models. Subsequently, the outputs were combined using a late fusion approach to predict user intents. They achieved an accuracy of 96% when using only action sequences for high-level intentions and 64% when using only gaze features. The study concluded that action sequences were more informative for intent prediction in this context than gaze data, while the fusion of both modalities did not significantly improve the results over using action sequences alone.

The majority of intention recognition methods have predominantly utilized the Long Short-Term Memory (LSTM) architecture to analyze sequences of actions, mouse or keyboard motions, or gaze dynamics. LSTMs have gained popularity because of their capacity to capture temporal relationships in sequential data. The development of Transformers has brought about a major change in the AI field, specifically in tasks involving sequences. This is due to the introduction of the self-attention mechanism, which has greatly enhanced the ability to process long-range sequences [63]. This trend has resulted in an increasing interest in investigating Transformers as a more efficient substitute for intention recognition tasks.

2.2.2 Large Language Models for Intention Recognition

Intention recognition is a crucial task aimed at enhancing user experience while interacting with the current environment. Existing works have employed the use of LLMs to capture the user intention based on their behavior. For instance, Bodonhelyi et al. [9] investigated the investigates the effectiveness of large language models (LLMs) like GPT-4 [1] in recognizing user intent and the impact of intent-based prompt reformulation on user satisfaction. The findings revealed that GPT-4 outperformed GTP-3 in recognizing common user intents but was less effective with less frequent ones. Additionally, when user intent was accurately identified, participants were more satisfied with responses to their original prompts than with those to reformulated prompts. Moreover, LLM4ISR [59] was designed to improve intent-aware session recommendations using LLMs. The methodology involved creating an initial prompt to guide LLMs in understanding and predicting user intents within a session. This prompt was iteratively optimized using a

self-reflection process where the LLM evaluated and refined its prompts based on identified errors. This approach not only enhanced the accuracy of session recommendations but also ensured better comprehension and transparency of user intents. In the paradigm of intention recognition in GUI environments, we could not find much work in this area. However, a single recent work [8] has briefly discussed an approach for autonomous agents to determine user intentions based on their interactions with the current GUI. The core challenge addressed is the ability of these agents to deduce a user's intended task from observed actions within a UI, thus enhancing the personalization and proactivity of agent-based interactions. The Android-In-The-Wild and Mind2Web datasets are utilized for experimental validation. Comparative experiments between human performance and state-of-the-art models, including GPT-4 and Gemini-1.5 Pro, demonstrate that while Gemini outperforms GPT-4, both lag behind human capability, indicating substantial room for improvement in model performance. Furthermore, the study implemented an automatic evaluation protocol using a large language model, specifically GPT-4o [1], to evaluate the quality of the intention summaries. This automated protocol assesses the satisfaction of intentions against both the ground truth and the historical actions documented.

We noticed that the usage of LLMs in intention recognition is an under-researched area, especially in GUI environments such as websites and Android devices. Moreover, existing works struggled to generalize on different sets of environments or intentions. To fill this gap, in this work, we are the first to employ LLMs for intention recognition in GUI environments using different interactive datasets such as Mind2Web [15] and MoTIF [10].

2.3 Next Action Prediction from Behaviour

A considerable number of works have studied the task of next-action prediction, as it is the main basis for task automation agents. In this part, we focus on the implementation of the next action prediction in Graphical User Interface (GUI) environments that can navigate the current environment autonomously. Typically, these agents are categorized into two groups: those that do not utilize LLMs (non-LLM-based) and those that do (LLM-based). This distinction is crucial as it influences the generalizability of the agents.

2.3.1 Non-LLM-Based Automation Agents

For the non-LLM-based agents, we witnessed different methods based on different techniques. For instance, Liu et al. [39] have employed a deep reinforcement learning

(RL) approach to implement an agent for web-based tasks, such as booking flights or managing emails. For the training and evaluation of their method, they utilized the World of Bits (WoB) [54] and Miniwob++[39]. Moreover, Shi et al. [54] combined RL with behavioral cloning (BC) to design agents to complete web-based tasks by performing low-level keyboards and mouse actions on the internet. Furthermore, CC-Net [28] have also combined RL and BC and achieved human-level performance on Miniwob++. Despite their remarkable result, they required an extensive dataset of 2.4 million demonstrations, equivalent to $6,300$ hours of human effort. Zhang et al. [78] implemented a multimodal approach for next-action prediction in text formatting, focusing on two distinct aspects of the behavioral timeline. Their research aimed at predicting future actions, from a closed set of predefined actions, by considering data from user interactions, including both mouse and keyboard activities, as well as gaze behavior.

While these methods have achieved remarkable outcomes, they share a notable limitation by building their agents on a simplified, simulated environment. This approach has led to poor performance on fully developed GUI environments and a lack of generalization across three different levels—task, website, and domain [15].

2.3.2 LLM-based Automation Agents

Recent advances in LLMs have significantly transformed the field of automation agent, due to their multi-decision making and reasoning capabilities. Given that many GUIs utilize text-based elements, such as HTML, there has been a surge in developing LLM-based automation agents tailored for various GUI contexts, including HTML environments, Android, and iOS. These agents leverage multiple LLM techniques, including prompt engineering and fine-tuning, to enhance task automation capabilities across these platforms. A primary limitation of these agents is the limited context length of LLMs, which restricts their ability to encompass the entire environment within a single context.

For HTML-based agents, MindAct [15] employs a two-stage model for training LLMs to operate in complex web environments. This methodology addresses the challenge of processing raw HTML from real-world websites, which can be too extensive for LLMs due to the large number of elements involved. First, the candidate generation steps involve a fine-tuned small LM that ranks the elements present on a web page. Based on the task description, snapshot of the web page at the given step, and the history of performed actions, this stage selects a pool of top candidate elements. The process uses a ranking task approach, where each DOM element is paired with the task query and fed into an encoder-only LM to generate a matching score. The top elements are selected based on the highest scores, significantly reducing the volume of data to be processed in the next stage. Subsequently, utilizing the candidate selected in the previous step,

task description, and history of actions to predict the next action. They formulated the next action prediction task in a multi-choice question-answering format. This allows the LLM to choose from a list of options rather than generating a complete target element. The LLM also generates the operation necessary for the selected element. After the successful implementation and testing, researchers noticed that MindAct struggled with complex long-term tasks. Another example for HTML-based agents, SYNAPSE [80] addressed this limitation by employing the multi-step decision-making of LLMs. SYNAPSE depended heavily on the in-context learning (ICL) of the LLMs by encoding the trajectories, actions, exemplars, and task descriptions in one input prompt fed to the LLMs. SYNAPSE consisted of three key components, each responsible for tackling different issues. First, **state abstraction** filters out irrelevant information from computer states (HTML), allowing it to focus on task-relevant data within the limited context of the LLMs. Second, **Trajectory-as-Exemplar Prompting (TaE)** uses complete trajectories of state and action Paris as prompt for LLMs, which improves the method's ability to make decisions over multiple steps. Lastly, **Exemplar Memory** stores successful trajectories (Mind2Web training data) and uses similarity search to retrieve relevant ones for new tasks, aiding generalization across different tasks and environments. Although SYNAPSE achieved remarkable results with the simplified MiniWob++, it struggled with the state length of real-world data such as Mind2Web.

Previous works also implemented LLM-based automation agents for the Andriod environment. For instance, DroidBot-GPT [69] presented an agent that leverages GPT for automating interactions with Android mobile applications based on natural language descriptions. The method involves transforming the current GUI state information and available actions into textual prompts. By employing zero-shot learning, the LLM is designed to accurately predict the subsequent appropriate action. This approach is fully unsupervised and utilizes the existing knowledge embedded in LLMs, trained on diverse software application manuals, enabling them to make informed decisions about UI interactions without specific training for each app. The evaluation conducted on a custom dataset comprising 33 tasks across 17 Android apps demonstrates a task completion rate of 39.39% and an average partial completion progress of 66.76%. Furthermore, AutoDroid [67] builds upon the foundational approach of DroidBot-GPT by enhancing the prompts with app-specific information through memory injection, thereby augmenting the LLM's commonsense knowledge. The method was evaluated using a benchmark of 158 tasks across 13 apps, achieving a task completion rate of 71.30% and an accuracy of 90.90%.These results significantly surpass those of baselines powered by GPT-4.

To overcome these limitations, we will combine the best of these works, by incorporating the candidate-generation part from Mind2Web and the trajectories from SYNAPSE to enhance the multi-step decisions of our methodology. We will also test the effectiveness of reformulating the task as a multi-choice question instead of a generative task.

3 Datasets

This chapter is dedicated to detailing the datasets utilized in our study. We primarily focus on **Mind2Web** [15]. Unlike another existing dataset that was constructed in a simulated environment [54, 76], Mind2Web has been collected from various sets of websites spanning different domains and crowdsourcing tasks to cover as mush as possible functionalities offered by these websites. Such an approach has filled the gap between the simplified simulated environment and the real-world environments. Therefore, agents trained on Mind2Web can perform better in real-world applications.

In addition to Mind2Web, we also incorporate another dataset called **MOTIF** [10] to further adapt, validate, and test our model generalizability on other GUI environments. MoTIF is a mobile-based dataset for interactive vision language navigation with unknown command feasibility. In the upcoming sections, we will discuss details about each dataset and the preprocessing applied to adapt it for our work.

3.1 Mind2Web

The main objective of the dataset is to implement an LLM-based agent that can follow textual instructions to complete a task on a target website. Each data point contains 3 parts. **Task Description** represents a high-level description of the task in natural language (English). **Action Sequence**, which is a sequence of actions required to complete a task on multiple web pages on a website. Each action is a tuple including *(Target Element, Operation)* where the target element is a DOM interactive element in the current GUI environment and the operation is a pair of *(action, value)*. Mind2Web supports three main operation: *click*, *select*, and *type*. For *select* and *type*, it provides an extra argument which is the value associated with the action. **Web-page Snapshot** contains both the raw HTML code of the web page and the DOM tree.

3.1.1 Data collection and Composition

To collect such a complex dataset and to verify the integrity and reliability of such data, they had a strict collection process that consisted of four stages:

Website Selection:
The process begins by picking up 5 top-level domains: Shopping, Travel, Service, Entertainment, and Information. These domains were broken down into 31 sub-domains. After picking some sample websites from each domain, these websites were ranked using *similiarweb.com* based on popularity in the US.

Task Proposal:
Annotators are given a target website, a brief description, and sample tasks to inspire them to propose diverse, open-ended, and realistic tasks that require multiple interactions and focus on high-level goals. Additionally, they prompted ChatGPT [71] to generate 50 seed tasks for the website, from which 10 are randomly picked and shown to the annotators as inspiration, not for direct usage, so boosting diversity and creativity. Rejections of annotator suggestions similar to the seed tasks before being used in demonstrations, the authors further screen the suggested activities for quality and diversity.

Task Demonstration:
They have developed a Playwright-based [51] tool to demonstrate task execution within a web browser. Workers use this tool to show how to perform their proposed tasks accurately. Each interaction is divided into two parts: selecting an element on the web page and then choosing an operation to execute on it. Workers confirm their selections at each step. Once the task is completed, they have the chance to review and modify the task description.

Task Verification:
All task demonstrations are verified by the authors to ensure accuracy and consistency. First, task descriptions are aligned with the annotated actions, with modifications made if necessary. Second, recorded actions are checked for correctness, and any extraneous steps are removed. Finally, the tasks' starting and ending points are standardized, such as excluding actions like closing popup windows or stopping at the search results page for certain tasks.

3.1.2 Statistical Analysis:

Mind2Web is now considered the benchmark for implementing generalized agents capable of following language instructions to perform complex tasks on real websites. As illustrated in Table 3.1, Mind2Web consists of $2,350$ tasks spanning 137 websites from 31 domains. These tasks are divided into two main non-overlapping sets: a training dataset with $1,009$ tasks from 73 websites and a test dataset comprising the remaining

tasks. In this section, we compare Mind2Web with previous datasets designed for similar purposes, highlighting its significant advantages.

Firstly, Mind2Web utilizes a diverse array of real-life websites, rather than simplified ones, from various domains. This diversity ensures that the dataset reflects a wide range of real-world scenarios. Secondly, Mind2Web includes high-level tasks that require a deeper understanding and more complex interactions. Lastly, the dataset presents complex environments where each page contains an average of $1,000$ DOM elements. These advantages make the task of building a generalized agent more challenging and realistic, thereby pushing the boundaries of current capabilities.

Due to these comprehensive and complex features, Mind2Web significantly surpasses other datasets in terms of providing a robust and challenging benchmark for evaluating the performance of generalized agents.

	# Dom.	# Env.	Env. Type	Avg. # Elements	# Tasks	Task Info.	Avg. # Actions
MiniWoB++ [39]	–	100	Simplified mobile websites	28	100	Low-level	3.6
WebShop [76]	1	1	Simplified shopping websites	38	12,000 products	High-level	11.3
RUSS [73]	–	22	Real-world websites	801	80	High & low	5.4
PixelHelp [36]	4	4	Mobile apps	–	187	High & low	–
META-GUI [58]	6	11	Mobile apps	79	1,125 dialogues	High-level	4.3
MoTIF [10]	15	125	Mobile apps	188	756	High & Low	4.4
MIND2WEB	5 / 31	137	Real-world websites	1,135	2,350	High-level	7.3

Table 3.1: Comparison between dataset and other benchmarks

As noticed in Table 3.1, the number of DOM elements in MIND2WEb is significantly greater than any other datasets designed for the same task. Moreover, this huge number of DOM elements will be a bottleneck in our methodology using LLMs because of the limited context length. To overcome this issue, they applied simple heuristics to clean the raw HTML files, keeping only visible elements and elements with a semantic meaning similar to the current task. Subsequently, the average number of DOM elements decreases from $1,135$ to 580 while keeping an overall recall of 94.7% for the target element in the training data.

3.1.3 Evaluation of Agent Generalizability

As shown in Figure 3.1, the diversity of MIND2WEB offers a unique opportunity to evaluate an agent's generalizability across different levels. To achieve this, they divided the remaining 1341 into three subsets, as follows:

- **Cross-Domain:** includes 912 tasks from 73 websites in two held-out top-level domains (Information and Service), challenging the model to generalize to entirely new domains without prior exposure.

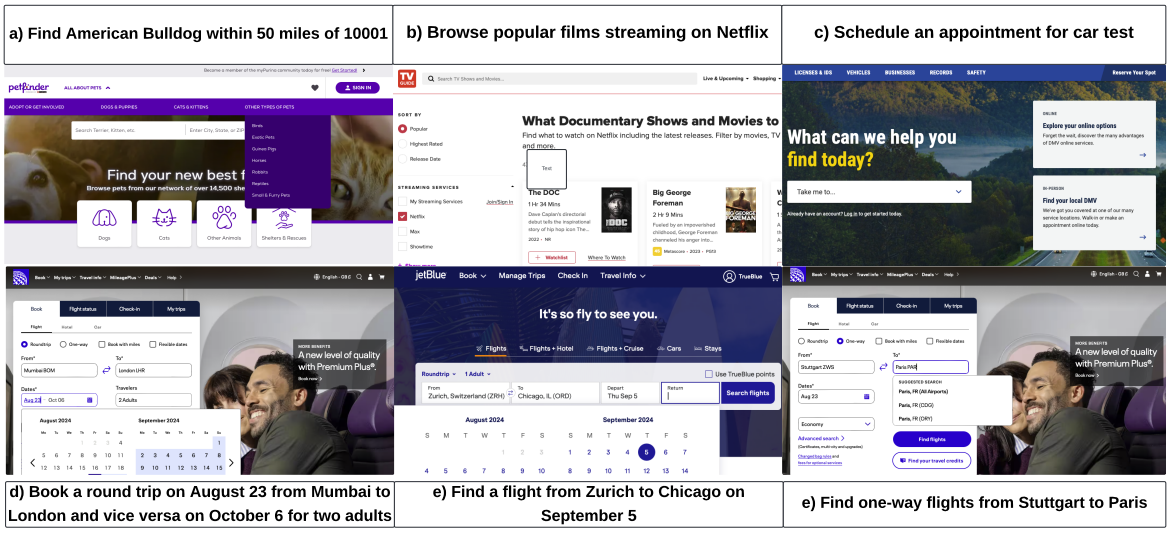

Figure 3.1: Examples for MIND2WEB test sets

- **Cross-Website:** comprising 177 tasks from 10 websites in each remaining top-level domain, testing the model's ability to adapt to new websites within familiar domains and task contexts
- **Cross-Task:** which consists of 252 tasks from 69 websites, randomly splitting 20% of the remaining data regardless of domains and websites, assessing the model's generalization to new tasks within known environments.

3.1.4 Limitation

As we demonstrated in the previous sections, MIND2WEB serves as a benchmark for training LLM-based computer agents. However, despite its advantages over the previous works, it still suffers from one significant limitation. It doesn't account for causal dependencies when performing tasks. For example, giving a high-level description of the task is "Book a flight from Stuttgart to Paris for 2 adults", one user can select the number of adults first and then set the origin and destination, while another might choose the origin and destination first then set the number of adults. Both approaches are valid and achieve the same result, yet MIND2WEB does not recognize this flexibility in task execution. This limitation may result in static instead of dynamic agents that can adapt to different GUI environments. Additionally, MIND2WEB does not record the default values of the website in the action history, as these do not require explicit user interaction.

4 User Intention Recognition

This chapter focuses on the utilization of Large Language Models (LLMs) for summarizing human intention recognition within graphical user interface (GUI) environments, such as websites and mobile applications. This summarization has various applications in Human-Computer Interaction (HCI), including unintentional error detection [4], next action prediction [7], task automation [27, 68, 69], and recommendation systems [40]. To achieve these applications, users must provide task intentions or descriptions as part of the input.

The primary goal is to demonstrate the application of LLMs in understanding and summarizing the underlying intentions behind a sequence of actions. Accomplishing this will simplify the implementation of the previously mentioned applications, reducing the need for detailed task descriptions. This not only streamlines the process but also minimizes human interaction, aiding in the completion of repetitive tasks.

In this chapter, we detail our pipeline to adapt Mistral-7B [29] into our **Mistral-Intention** and the enhancements we have made to improve its performance metrics. We will then present the results of these modifications and conclude with an ablation study to highlight the impact of each change.

4.1 Task Definition

Given an observed UI trajectory where a user interacts with the interface to complete a specific task, our objective is to infer the user's original intent from this sequence of actions, as shown in Figure 4.1. This intent identification scenario is essentially the reverse of the UI Automation task. To tackle this, we repurposed the input and output structures of UI Automation, swapping their roles. This approach allows us to utilize UI automation datasets for intent identification.

Our input comprises a UI trajectory, which is a series of steps executed by the user. Each step includes a snapshot of the UI at that moment and the corresponding action taken by the user. From this sequence, we aim to generate a natural language description that accurately reflects the user's intended task.

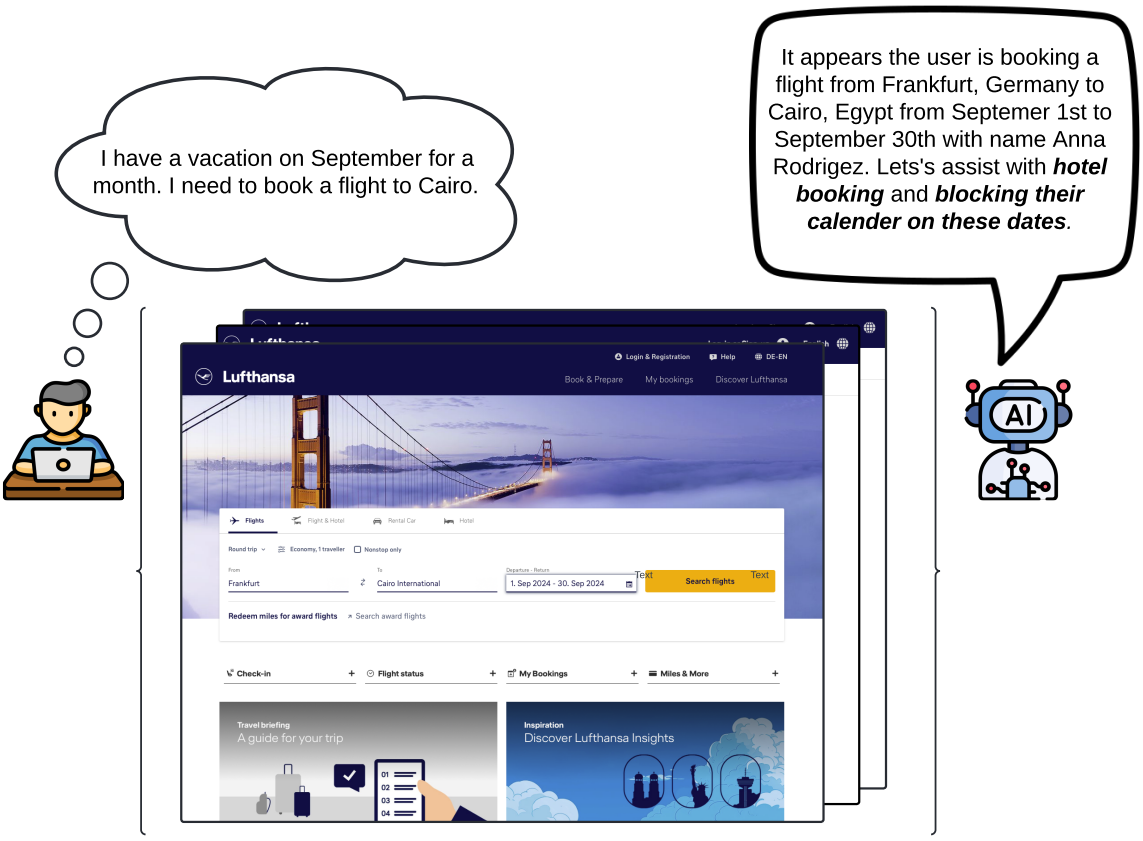

Figure 4.1: Task definition visualization

We focus on the fundamental aspect of the intent recognition task, assuming that the observed UI trajectory successfully represents the user's intent with no error. By addressing this core scenario, we aim to establish a reliable method for interpreting user latent intentions from their UI interactions.

4.2 Methodology

As illustrated in Figure 4.2, Mistral-Intention decomposes of the following steps:

1. **Sub-Goals Generation:** Utilize prompt engineering to generate a description for each step.
2. **Keyword Extraction-based Loss:** Concentrate on the significance of each action by extracting relevant keywords.

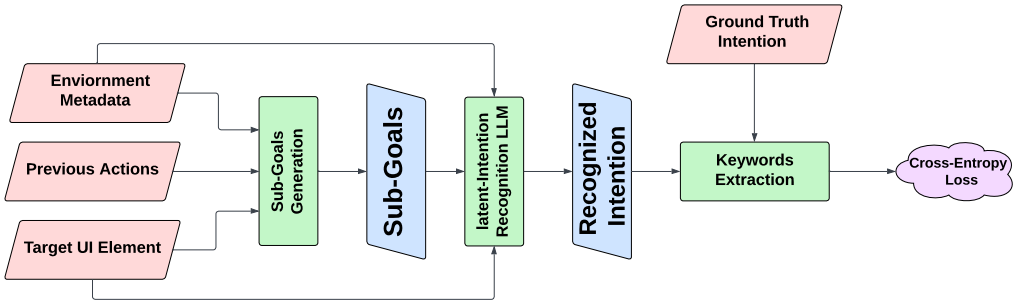

Figure 4.2: Mistral-Intention Training Pipeline

3. **Overall Goal Recognition:** Develop a comprehensive summary of the user's intention.

4.2.1 Sub-Goals Generation

The primary objective of this step is to break down the overall task goal into several smaller sub-goals that must be accomplished sequentially to achieve the main goal. Successfully implementing this step will enhance the extendability of our pipeline and increase its robustness for handling longer action sequences. This means that when a new action is triggered by the user, we do not need to summarize all the actions again. Instead, we only need to summarize the new actions alongside the previously summarized ones.

Sub-Goal Creation through Prompt Engineering

Due to the nature of our datasets, Mind2Web and MoTIF, sub-goals are not inherently part of the dataset. Previous works [66] in the fields of text summarization and classification have addressed this issue by utilizing in-context learning (ICL) capabilities, specially few-shot learning, of any large-scale autoregressive language models to augment their datasets with new samples.

In one-shot learning for text generation, the input consists of a single example text that provides a reference style, structure, and content. We expect our model to utilize this example to generate new text that matches the characteristics of the input. Inspired

by [14, 77] and due to being excellent few-shot learners, we opt for GPT-3 as our data augmentation model.

The prompt used for sub-goals generation consists of 3 key components, as shown in Figure 4.3. Firstly, The basic instruction where we represent the task and the expected output. Secondly, the example input is composed of environment metadata, history of actions and target UI, and textual experience of actions. Lastly, the expected output structure is to be generated by our model.

This design efficiently decomposes the overall goal into sub-goals, considering all essential inputs required for the sub-goals to collectively achieve the main objective.

4.2.2 Intention Recognition Step

In this step, our objective is for the LLM to recognize and summarize the latent intention behind each performed action, providing reasoning for why the user executed each action. Specifically, we aim to guide our LLM to produce a comprehensive and detailed summary of the user's intentions, incorporating all important information.

To achieve this, the LLM must not only identify the immediate goals of individual actions but also understand the broader context and objectives driving the user's behavior. This involves interpreting the sequence of actions, the associated user interface elements, and the GUI environment metadata to form a coherent narrative.

Intention Recognition as Next Token Prediction

Leveraging the auto-regressive nature of the LLM, we have formulated the recognition step as a next-token prediction task. We fine-tuned an LLM to summarize the user intention and generate a final detailed summary in one sentence. Our input text has five main parts, as illustrated in Figure 4.4.

The basic instruction involves explaining the task to the LLM and specifying what needs to be accomplished and the inputs needed. The environment metadata provides the LLM with context about the website or app where these actions occur. The history of actions and UI elements includes the actions themselves, along with their corresponding values and UI elements. The final part is the generated sub-goals generated in Section 4.2.1.

Keywords-Extraction Based Loss

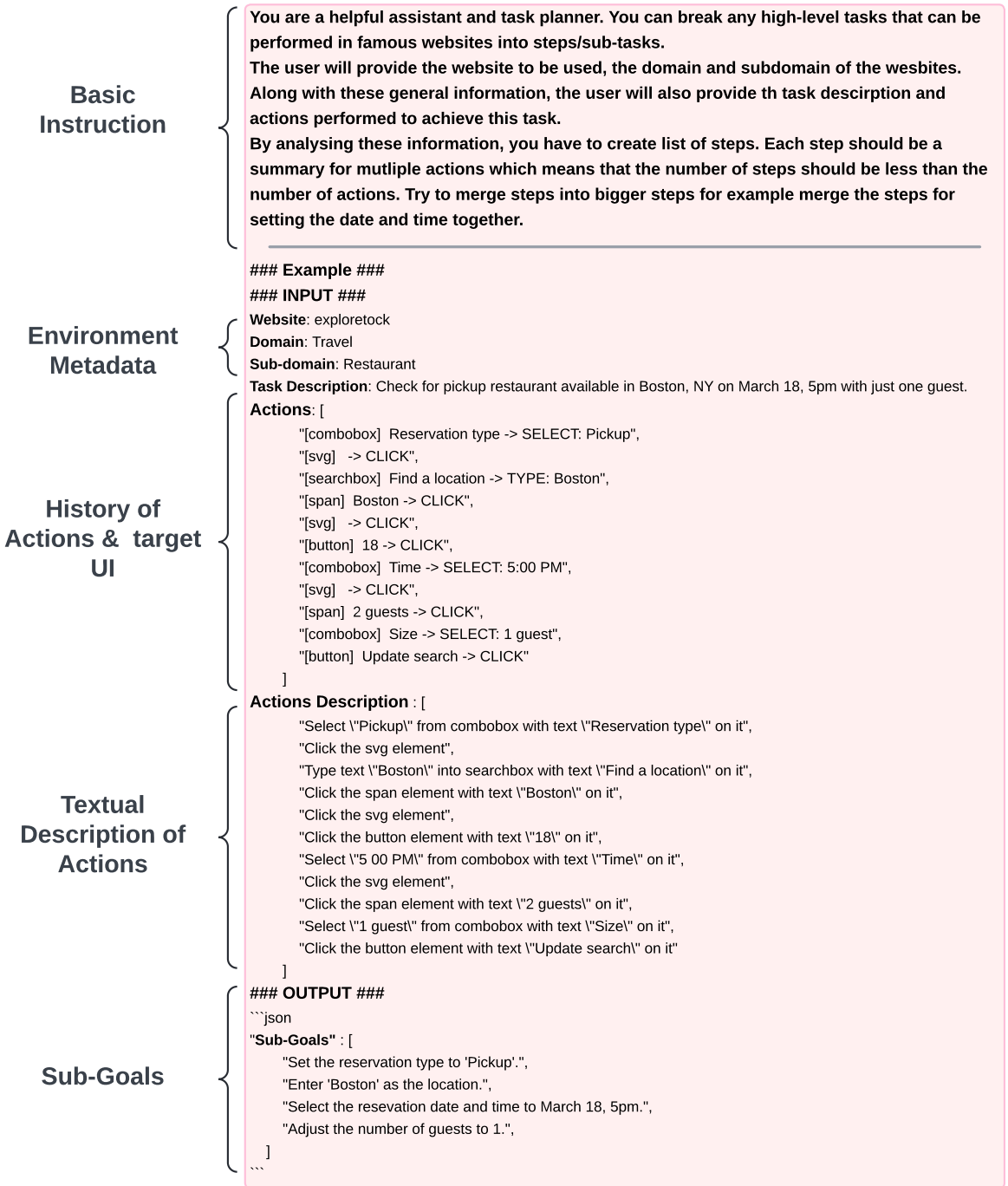

Figure 4.3: Sub-Goals Generation Prompt

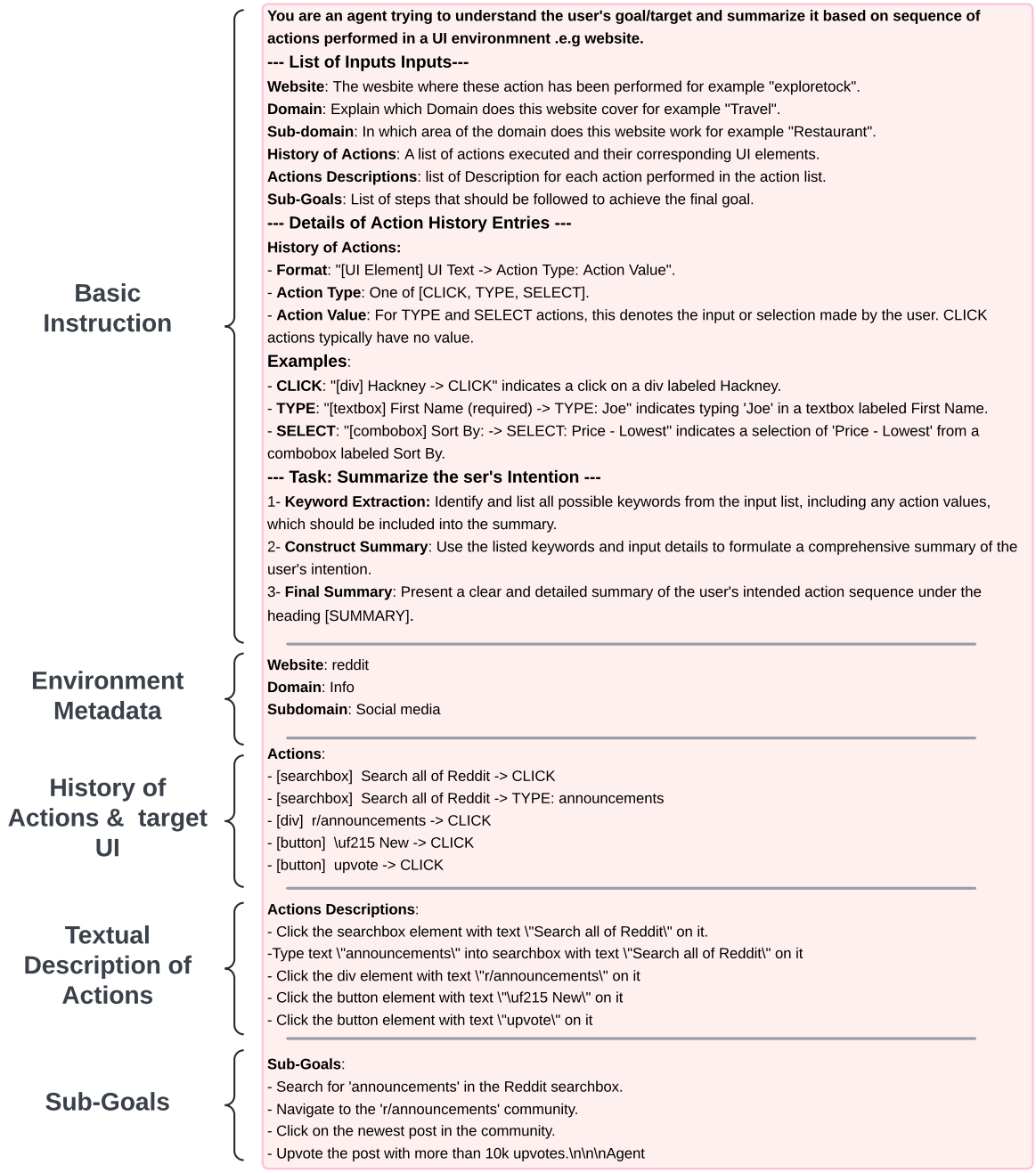

Figure 4.4: Our Recognition Model Prompt

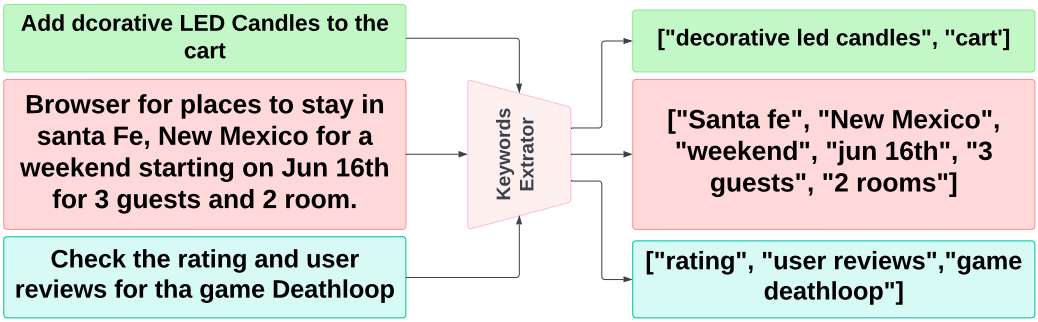

Figure 4.5: Output of KeyBERT after applying it to a sample of our dataset.

Our goal recognition task shares similarities with other text generation tasks, such as text summarization [31] and question answering [34], where multiple valid outputs can exist. This ambiguity arises because of the synonyms and richness of each language.

After the initial fine-tuning results, we noticed that our method tends to over-summarization, leading to incomplete summaries that are unsuitable for downstream tasks. For instance, the goal "Log in using email: test@test.com and password: 12345" is overly summarized into "log in" causing the loss of essential details. To solve this issue, we aim to direct our LLM to give more attention to the critical words within both the action history and the sub-goals created during the planning step. For example, in a typing action, important keywords would include the specific input being typed. Furthermore, we want to ensure that our model does not overly focus on the verb used, such as distinguishing between "find" and "search", nor on the exact order of words in a sentence, as long as the generated sentence is grammatically correct because multiple sentences can express the same meaning.

To achieve this biased attention, we employed a new loss function called keyword extraction-based loss. It forces the LLM to give more attention to the critical details such as typed values in text fields or selected inputs from drop-down lists. The key complement of this loss is the integration of keyBert [20]. KeyBert is a pre-trained keywords extraction model that benefits from BERT's embedding [16] that captures the contextual meaning of a word in the given text, as shown in Figure 4.5.

Overall loss Calculation

After successfully generating the tokens for our final summary, the loss calculation involves two steps. First, we utilize KeyBERT to extract the keywords from both the gen-

erated token set and the ground truth token set. Secondly, we compute the cross-entropy loss between these tokens to determine the overall final loss. By incorporating this loss into the training regime, we ensure that the summarized intention is semantically aligned with the ground truth goal. This approach enhances the accuracy and relevance of the generated summaries.

4.3 Implementation Details

In this section, we will highlight some details in the training of our pipeline including the models used to derive the metrics of our framework.

4.3.1 Our Large Language Model

Given the substantial computational power required for fine-tuning large LLMs, we opted to use a relatively smaller model with fewer than 7 billion parameters. Our recognition model is based on the Mistral-7B architecture, which strikes a balance between performance and efficiency.

Despite its relatively small size, Mistral-7B has overcome other open-source models with the same or higher number of parameters such as LLaMa-2-13B [62] in many tasks such as reasoning and comprehension. This choice allows us to leverage advanced language model capabilities while maintaining manageable resource requirements. By using Mistral-7B, we ensure that our system remains accessible and efficient without compromising the quality of the generated summaries.

This model's efficiency and effectiveness make it well-suited for our application, providing robust performance in recognizing and summarizing user actions within GUI environments. It has two additional advantages over other open-source LLMs with the same number of parameters: it uses Grouped-Query Attention (GQA) for faster inference, which is essential for our use case. Secondly, it employs Sliding Window Attention (SWA) to handle longer sequences at a smaller computational cost.

4.3.2 Mistral-Intention Finetuning Details

We leverage the ModelCenter framework [44] to fully fine-tune the Mistral-7B model using four Tesla V100-32GB GPUs. Key hyper-parameters for this fine-tuning process are detailed in table 4.1. Additionally, we dynamically adjust the loss scale in response to changes in training loss to prevent underflow.

Model	Max Length	Epochs	Batch Size	LR	Time (h)	LR Scheduler	Optimizer
Mistral-Intention	2048	20	16	1e-06	13	Cosine	AdamOffload

Table 4.1: The hyper-parameters applied during the training of Mistral-Intention.

4.3.3 Development of a Comparative Baseline

The study of intention recognition in real-life GUI environments, which involves a vast number of intentions, is an under-studied area. As a result, we were unable to locate a suitable baseline method for comparative analysis. As mentioned before in 4.1, our task is the reverse of task-automation. Subsequently, We have adapted SYNAPSE [80] to establish our baseline since it is the state-of-the-art method for task automation using LLM. As mentioned before in Chapter 2, SYNAPSE consists of three key components: The first three strategies are state abstraction (removing task-irrelevant information from raw states; raw HTML), trajectory-as-exemplar prompting (prompting the LLM with full trajectories of the abstracted states and actions to improve multi-step decision-making), and exemplar memory (storing exemplar embeddings and retrieving them via similarity search for generalization to new tasks).

Modification on State Abstraction

The original implementation of the state abstraction component requires the original intention as part of the input, providing the model with insight into the expected output. To overcome this, we have modified the state abstraction process to filter out irrelevant information and UI elements based on previous actions and the generated sub-goals.

Modification on Exemplar Memory

For this part, we only modified the semantic search to retrieve exemplars for the current task. Unlike the initial setup, where retrieval was based on task intention, our approach computes retrieval using the action history and sub-goals.

Modification on Trajectory-as-Exemplar

This component is responsible for augmenting the input to the LLM with the full trajectory of actions. It also provides exemplar trajectories retrieved from the exemplar memory using semantic search based on the metadata of the current task. To make it

compatible with our task, we adjusted the trajectories to include only the abstracted state and the previous actions.

After the implementation of these modifications, we used the same fine-tuning procedure that we used to train Mistral-Intention Section 4.3.2.

4.4 Evaluation Protocols

To evaluate our results, we employed different metrics, including both n-gram-based and embedding-based metrics. we also conducted an evaluation protocol that included both human assessments and automatic evaluations using language models. The details of these evaluation methods will be explained in the subsequent sections.

4.4.1 N-Gram Based Metrics

Our goal recognition task closely parallels other text generation tasks, such as text summarization and question answering, as demonstrated in previous research [31, 34]. As a result, we have adopted several of their established evaluation metrics, including:

Recall-Oriented Understudy for Gisting Evaluation (ROGUE)

ROUGE [38] is a set of metrics used for evaluating summarization quality by comparing AI-generated summaries to human-created ones. It focuses on recall, measuring the overlapped n-grams between the summaries. The final goal is to ensure the generated summary captures as much relevant contextual information from the reference summary as possible.

ROUGE-N is a specific variant that evaluates the overlap of n-grams, which are contiguous sequences of n items from a given text. For example, ROUGE-1 measures the overlap of single words (unigrams), while ROUGE-2 measures the overlap of two-word sequences (bigrams).

ROUGE-W (Weighted Longest Common Subsequence) is another variant that adds different weights to the elements of the common subsequence, giving more attention to longer subsequences. This approach helps effectively capture the structure of the summaries.

However, the core ROUGE metrics are primarily recall-oriented, different variants like *ROUGE-L* (Longest Common Subsequence) incorporate aspects of both precision and recall.

Metric for Evaluation of Translation with Explicit Ordering (METEOR)

METEOR [6] is an automatic metric for machine translation evaluation based on a concept of unigram matching between machine-produced translations and human-produced references. Unigrams are matched by surface forms, stemmed forms, and meanings.

METEOR computes a score using the following components:

- **Unigram Precision**: The ratio of unigrams in the machine translation that are also in the reference translation.
- **Unigram Recall**: The ratio of unigrams in the reference translation that are also in the machine translation.
- **Fragmentation Penalty**: A measure of how well-ordered the matched unigrams are in the machine translation relative to the reference. This penalty is designed to capture the degree of disorganization in the translation.

The final METEOR score is a weighted combination of unigram precision, unigram recall, and the fragmentation penalty. The formula is designed to balance the importance of matching content (precision and recall) with the quality of word order (fragmentation).

Bilingual Evaluation Understudy (BLEU)

BLEU [47] is an automatic metric used to evaluate the quality of machine-translated text. It compares the machine-generated translations to human-produced reference translations by calculating the precision of n-grams (contiguous sequences of n words) present in both. Key components of BLEU include:

- **N-gram Precision**: Measures the proportion of n-grams in the candidate translation that appear in any of the reference translations. BLEU typically combines unigram, bigram, trigram, and higher-order n-gram precisions.
- **Brevity Penalty**: Adjusts the score to account for translations that are too short, ensuring that shorter translations that may miss important content are penalized. The final BLEU score is a weighted geometric mean of the precision scores, combined with the brevity penalty.

It is widely used due to its simplicity and effectiveness concerning human judgments of translation quality, making it a standard metric in the field of machine translation evaluation.

4.4.2 Embedding-Based Metrics

While the previously mentioned n-gram-based metrics are widely used in various text-based tasks, they have significant limitations. Firstly, they do not fully capture all aspects of the generated text's quality, such as fluency and context. Secondly, they did not take into consideration the actual semantic meaning of the generated text. Additionally, they assume that all the words in a sentence have the same importance degree, failing to account for synonyms and the varying significance of different words [23].

Moreover, these metrics are biased to penalize flexible generations that result in the same meaning using different words leading to inaccurate assessments of the generated text. Therefore, as demonstrated by [49], these metrics should be complemented with other evaluation methods that consider the overall semantic content, coherence, and appropriateness of the text within its broader context.

Cosine Similarity

To overcome n-gram metrics limitations, we decided to employ embedding-based metrics, specifically cosine similarity, that capture the semantic meaning of the text. Cosine similarity allows us to evaluate how similar the generated text is to the original text's semantics.

Calculating the cosine similarity between two sentences requires converting them into vectors (embeddings) within the same vector space. These vectors represent the semantic and syntactic information of each sentence.

4.4.3 Human Evaluation Protocol

A recent study [56] has shown that cosine-similarity is not a reliable metric and it is heavily dependent on the method and regularization technique which leads to meaningless similarity scores. Therefore, we can not depend only on it for capturing the semantic similarity between the generated intention and the human-created one.

To avoid this confusion, we conducted a comparative user study where the evaluator assessed the recognized intention from both our fine-tuned baseline and Mistral-Intention, as well as the ground truth intention. They were asked to rate the similarity on a scale from 1, which represents low similarity, to 5, highest similarity, determining how each of the recognized and ground truth intentions satisfies each other.

4.4.4 Automatic Language Models Evaluation Protocol

As mentioned before, there is a common behavior ambiguity in intention recognition tasks and other text generation tasks, such as summarization. This inherent ambiguity arises because a single trajectory can fulfill multiple intents. To further address this challenge during evaluation, we have conducted an automatic evaluation protocol.

Inspired by Berkovitcj et al. [8], we adapted the **satisfaction relation between tasks** metric as a matching criterion between the summarized intention and the corresponding ground truth task description.

Satisfaction Relation Between Tasks

In a GUI environment with two task descriptions A and B, A is said to satisfy B if every possible user trajectory that accomplishes A also accomplishes B. This implies that completing A leads to the completion of B, thereby categorizing B as a more general task than A within that environment. Based on these criteria, a predicted task description matches the ground truth description if they mutually satisfy one another, and it is considered a partial match if only one satisfies the other. Therefore, tasks that match can essentially be viewed as paraphrases expressing the same intent within the UI context.

4.5 Main Results

The enhancements integrated into our pipeline have shown a notable increase in intention recognition when compared to the baseline created by the adaptation of SYNAPSE on Mind2Web and MoTIF. This performance boost suggests our model's effectiveness in recognizing latent intention, thereby potentially being utilized in HCI tasks such as unintended error detection. Our model inference time In this section, we will further discuss both the quantitative and qualitative results of our methodology with a comparison to the baseline.

4.5.1 Quantitative Results

To further understand results, we represent the quantitative ones of Mistral-Intention on four different protocols as explained before in Section 4.4. Each of these protocols provides further insights into our methodology.

Automatic Numerical Evaluation Metrics

Mind2Web:

Our model achieved cosine similarity scores of 0.83 on cross-domain, 0.82 on cross-task, and 0.84 on cross-website evaluations, with an average inference time of 2 seconds. The relation between inference time and number of input tokens is visualized in Appendix B. As Table 4.2 shows, Mistral-Intention demonstrated a significant improvement over the fine-tuned SYNAPSE, with an absolute increase in cosine similarity scores of 0.64 on the cross-domain, 0.62 on the test task, and 0.62 on the test website. Our analysis indicates that Mistral-Intention exhibits better performance on the Cross-Website test dataset compared to other datasets. This suggests that Mistral-Intention has enhanced generalization capabilities, particularly in adapting to new websites. Such a finding underscores its robustness and applicability in diverse web environments, marking it as a particularly effective tool in scenarios involving novel website content.

Metric	Cross Domain		Cross Task		Cross Website	
	Our	**SYNAPSE**	**Our**	**SYNAPSE**	**Our**	**SYNAPSE**
SacreBLEU	**23.72**	2.09	22.75	2.20	21.02	2.06
ROUGE-1	0.5960	0.1224	0.5932	0.1265	**0.6094**	0.1283
ROUGE-2	0.3551	0.05303	0.3539	0.0545	**0.3554**	0.0525
ROUGE-L	**0.5279**	0.1049	0.5154	0.1070	0.5182	0.1050
ROUGE-W	**0.5281**	0.1050	0.5154	0.1074	0.5181	0.1055
METEOR	0.5712	0.0708	0.5548	0.0737	**0.5773**	0.0781
Cosine Sim.	0.8369	0.1999	0.8257	0.2034	**0.8497**	0.2201

Table 4.2: Comparison Performance Metrics: Mistral-Intention vs. Fine-Tuned SYNAPSE Across Different Mind2Web Test Sets

MoTIF:

For further evaluation and to test the generalizability of Mistral-Intention, we tested Mistral-Intention on MoTIF. As shown in Table 4.3, Mistral-Intention performs better on all metrics when testing on MoTIF than Mind2Web due to the number of performed actions in each dataset. In Mind2web, the average number of actions is 7.30, however, in MoTIF is 4.40. For more analytical details about each dataset, check Chapter 3.

Metrics	ScareBLUE	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-W	METEOR	Cosine-Similarity
MoTIF	60.12	0.8570	0.8418	0.8570	0.8570	0.8455	0.8787

Table 4.3: Performance Metrics of Mistral-Intention on MoTIF

Human Evaluation Protocol

We split our participants into two non-overlapping groups, Group A consisted of 30 participants, and Group B had 36 participants. Then, we randomly sampled 30 instances from Mind2Web, selecting 10 instances from each test dataset. Each group was presented with 15 non-overlapping instances. For the correctness of our evaluation, we only consider participants with at least a business-fluency English level.

AS in Table 4.4 demonstrates, Mistral-Intention outperforms the adapted SYNAPSE across the 3 different test datasets. With an increase of 1.42% in Cross-Domain $p < 0.001$), 1.51% in Cross-Task $p < 0.001$), and 1.56 in Cross-Task ($p < 0.001$). Furthermore, A Wilcoxon signed-rank test confirmed the statistical significance of our method (p < 0.001). Additionally, our human evaluation corroborates the findings presented in **??**, reinforcing that Mistral-Intention demonstrates superior generalization capabilities for new websites. This alignment between human assessments and automated metrics further validates Mistral-Intention's effectiveness across diverse web environments.

Metric	**Cross Domain**		**Cross Task**		**Cross Website**	
	Our	**SYNAPSE**	**Our**	**SYNAPSE**	**Our**	**SYNAPSE**
Mean Score	3.95	2.53	3.91	2.40	**4.00**	2.44

Table 4.4: Human evaluation protocol mean score overall test dataset

Automatic Language Models Evaluation Protocol

Building on recent advances in LLMs and inspired by Berkovitch et al. [8], we employed GPT-4o [1] as the automatic evaluator of the satisfaction scores of our predicted intentions against the ground truth task descriptions. The specific prompt used is detailed in Appendix A. Since the satisfaction task is a classification problem where the LLM assigns a binary label to each recognized intention, we used the F1 score as its primary evaluation metric. This approach provides a balanced assessment of both precision and recall in the model's performance. The outcomes of our analysis support those of the protocols previously discussed. Notably, our method achieved an F1 score of 0.79, marking a 0.04 improvement over Berkovitch et al. However, it is important to note

that our approach relies solely on textual data, in contrast to the image and textual modalities used by [8].

Although these quantitative results are indicative of the method's success, the ultimate test would be would be a qualitative assessment. The following section will present quantitative examples from the test dataset, illustrating the enhancement achieved by our model.

4.5.2 Qualitative Results

To complement our quantitative findings, we examined individual tasks from our test datasets to evaluate the results of our methodology, particularly focusing on identifying the underlying goal from the input list.

In Figure 4.6, you can find the same list of inputs as illustrated before in Figure 4.4. The two parts at the bottom represent the ground truth goal as provided by Mind2Web and the recognized goal by Mistral-Intention, respectively. This example confirms that our model can successfully recognize the latent intention/goal of the performed action. Moreover, We can notice how Mistral-Intention extracted all the critical words from the input list such as the word *announcements*, which is the typed value in the second action. We can also conclude that Mistral-Intention is capable of recognizing the underlying meaning of each performed action and producing a detailed and descriptive summary without ignoring the essential information.

4.6 Ablation Study

In our research, we conducted an ablation study to evaluate the impact of each alteration applied to the standard Mistral-7B. We began by fine-tuning Mistral-7B using the Mind2Web dataset. Following this, we incrementally integrated the Sub-goals generating step and the keywords extraction loss to arrive at our final enhanced methodology. The performance of the model was assessed to understand the contribution of each modification to the overall effectiveness of the model.

4.6.1 Impact of Finetuning

LLMs are well-known for their in-context learning capabilities through prompt engineering [43]. To assess the impact of our fine-tuning process, We tested the vanilla Mistral-7B leveraging the zero-shot learning technique. As anticipated, the performance

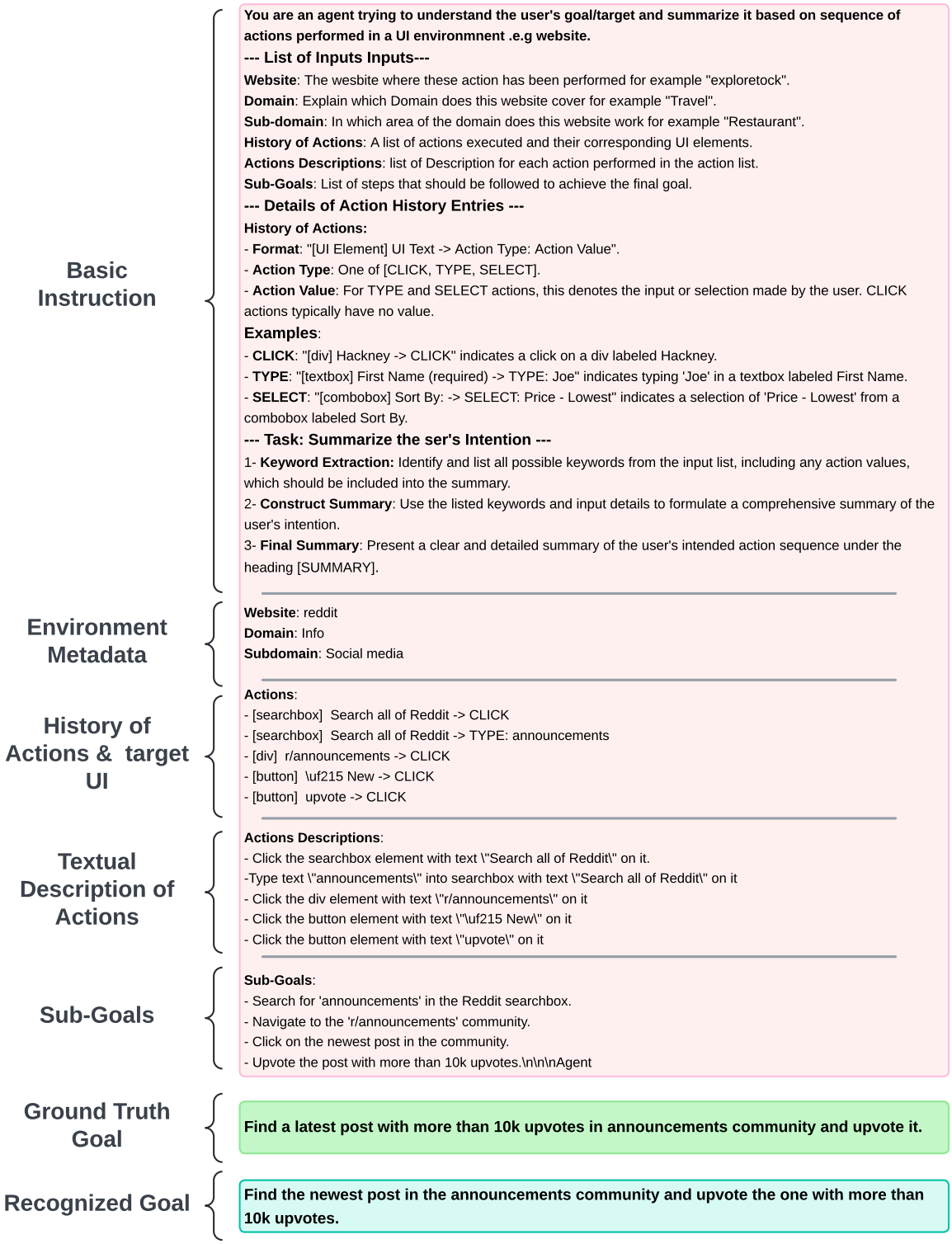

Figure 4.6: Mistral-Intention Training Pipeline

of the vanilla model was relatively low on all the evaluation metrics used, as indicated in Table 4.5.

Metric	Cross Domain		Cross Task		Cross Website	
	Our	Mistral-7B	Our	Mistral-7B	Our	Mistral-7B
SacreBLEU	**23.72**	1.06	22.75	1.05	21.02	0.9973
ROUGE-1	0.5960	0.0835	0.5932	0.0885	**0.6094**	0.0606
ROUGE-2	0.3551	0.0403	0.3539	0.0410	**0.3554**	0.0423
ROUGE-L	**0.5279**	0.0727	0.5154	0.0751	0.5182	0.0774
ROUGE-W	**0.5281**	0.0793	0.5154	0.0841	0.5181	0.0864
METEOR	0.5712	0.1787	0.5548	0.1862	**0.5773**	0.1927
Cosine Sim.	0.8369	0.4823	0.8257	0.4702	**0.8497**	0.4906

Table 4.5: Comparison Performance Metrics: Our Mistral-Intention vs. Mistral-7B across different Mind2Web test sets

Figure 4.7 provides a visual comparison, illustrating that the vanilla Mistral-7B model was less effective at recognizing, understanding, and summarizing the latent intention of the user compared to the Mistral-Intention. It also reinforces the quantitative data, underscoring the need for further refinements to achieve a better semantic understanding of the performed actions and their associated values.

As illustrated in Figure 4.7, Mistral-Intention can capture the semantics and summarize the user intention in a detailed summary. In the too example of Figure 4.7, our model grasped that the latent intention was to save the first article on the homepage, regardless of which article appears first, rather than saving a specific article titled "Q&A With Daniel Willingham".

4.6.2 Impact of Keywords Extraction Loss

Our next step in refining our method involved the integration of the keywords extraction loss. This loss's role is crucial; it ensures the recognized intentions by our model are not only similar to the ground truth intention (as indicated by the cosine-similarity score) but also contain the most essential information from the history of actions. the addition of this loss guided the LLM to pay more attention to keywords, such as action values, and less attention to normal words, such as verbs. This goal was successfully met as evidenced by an increase in the cosine similarity score of almost 13% and improved performance across all metrics used for the evaluation of our method, as detailed in Table 4.6.

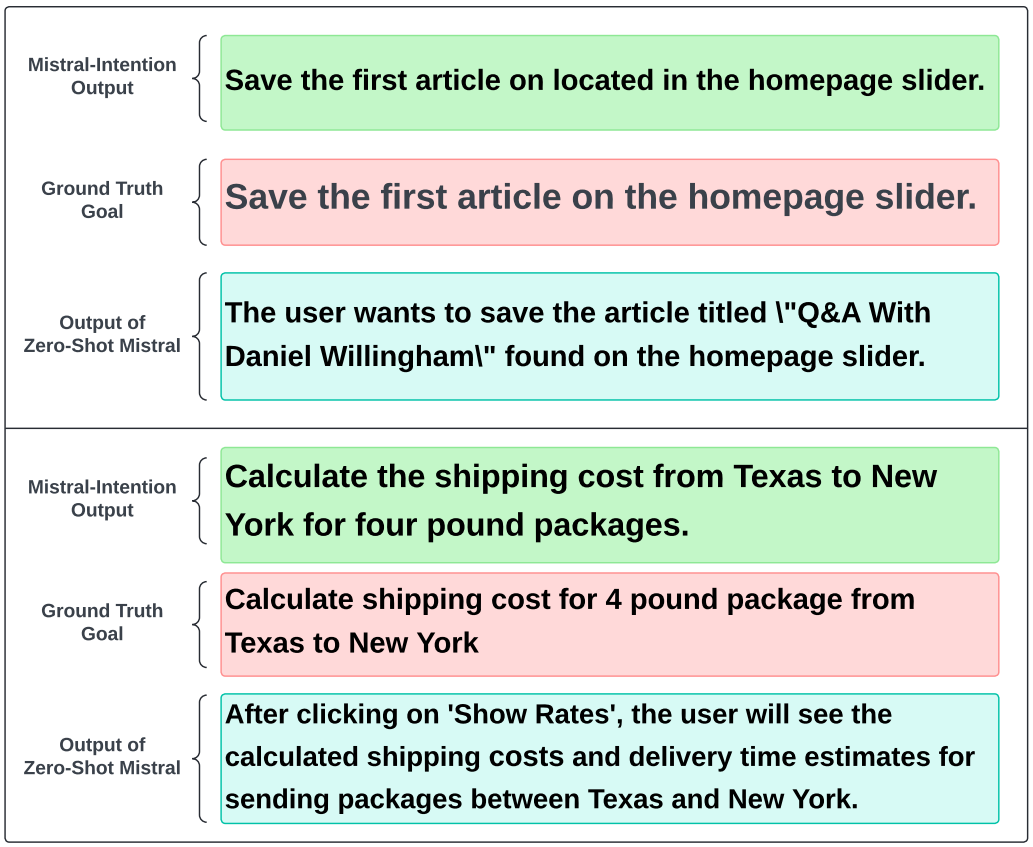

Figure 4.7: Examples of intention recognition before and after the fine-tuning, alongside the ground truth intention for comparison.

Figure 4.8 demonstrates the functionality of the keywords extraction loss. It shows a comparison between the intentions recognized with and without it. In each sub-image, the top rectangle displays the output after incorporating this loss into Mistral-Intention. Here, you can notice that all the critical information from the ground truth presents in our recognized intention, such as the place, price range, and number of rooms required. On the other hand, the bottom rectangle of each sub-image represents the output without including this loss, it is obvious that critical details, such as the place, are missing. This comparison underscores the keywords extraction loss's effectiveness in forcing the model to focus more on the important details throughout the performed trajectory.

Through the words extraction loss's inclusion, our model not only semantically understands and summarizes the latent intention (as reflected by the cosine similarity score) but also shows marked improvement in including the important information in the summary, a crucial aspect for potential usage of our method in downstream tasks such as personalized recommendation and task automation.

Metric	Cross Domain		Cross Task		Cross Website	
	w/ KW-E	w/o KW-E	w/ KW-E	w/o KW-E	w/ KW-E	w/o KW-E
SacreBLEU	**23.72**	18.52	22.75	17.83	21.02	15.08
ROUGE-1	0.5960	0.5048	0.5932	0.5216	**0.6094**	0.5147
ROUGE-2	0.3551	0.2962	0.3539	0.2714	**0.3554**	0.3066
ROUGE-L	**0.5279**	0.4461	0.5154	0.4530	0.5182	0.4318
ROUGE-W	**0.5281**	0.4779	0.5154	0.4792	0.5181	0.4630
METEOR	0.5712	0.5124	0.5548	0.4786	**0.5773**	0.4991
Cosine Sim.	0.8369	0.7436	0.8257	0.7358	**0.8497**	0.7523

Table 4.6: Comparison Performance Metrics before and after adding the keywords extraction loss

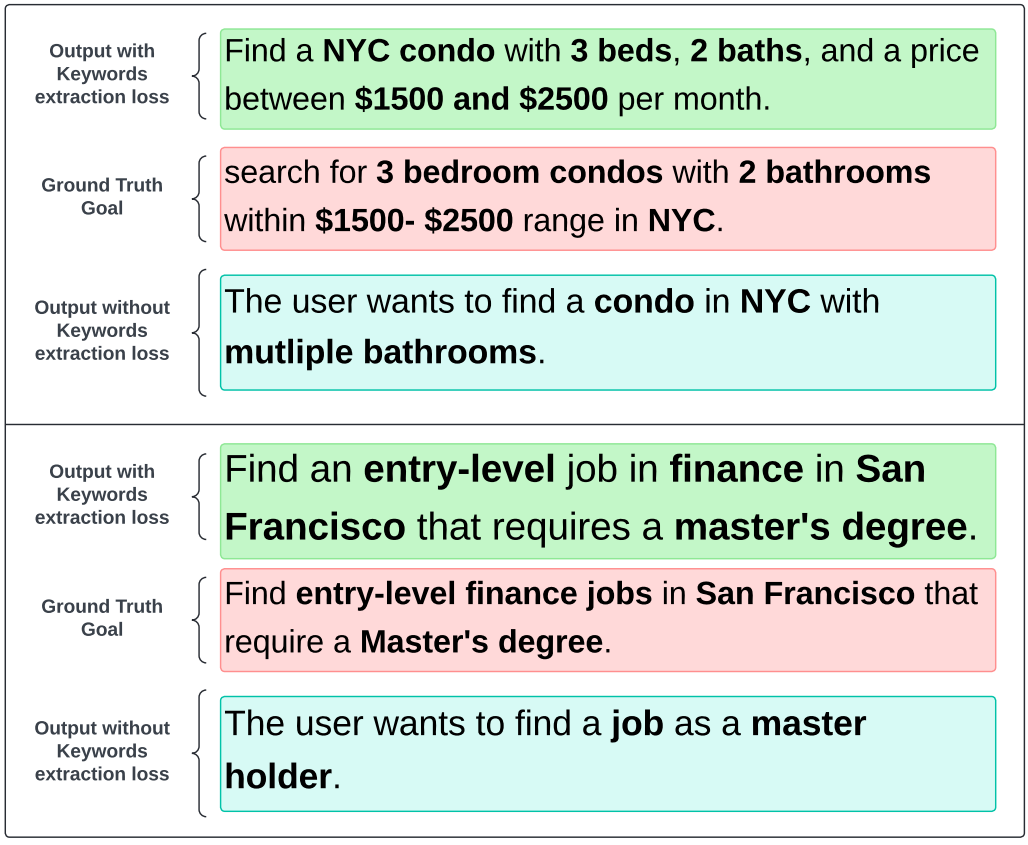

Figure 4.8: Examples of intention recognition before and after the integration of keywords extraction loss, alongside the ground truth intention for comparison.

5 Next Action Prediction

This chapter focuses on building an LLM-based computer agent for next-action prediction in GUI environments, such as websites and mobile applications. The implementation of such a model holds the premise of establishing a daily assistant agent that can help with repetitive tasks. Therefore, we can increase human productivity, enhance user experience, and enable more responsive systems.

Although previous LLM-based computer agents have achieved promising results, they struggle to solve simplified tasks within an abstracted GUI environment, due to the following reasons. Firstly, the constrained context length of LLMs, coupled with the complexity of GUI environments, significantly hinders the application of few-shot learning for such tasks. Additionally, the raw HTML files of real-world websites are often too rich to fit within a single prompt [15]. Secondly, GUI environments contain a huge number of task-irrelevant information and UI elements that can distract LLMs and cause hallucinations [42]. Subsequently, these agents struggle to solve long-horizon tasks. Lastly, ICL requires task-specific exemplars to be embedded in the prompt to direct the LLM output. Opposite to other tasks that can be successfully achieved using ICL, such as language translation, it is not a trivial task to find exemplars to be embedded into the prompt.

Given an environment of a Graphical User Interface (GUI), a task description, and a history of action, our objective is to build an LLM-based computer agent that can understand the current environment state and anticipate the user's next action to be performed. This prediction involves predicting the action itself, with its associated value if applicable, and the target UI element.

5.1 Methodology

As illustrated in Figure 5.1, our LLM-based agent decomposes of two key components:

1. **GUI Cleaning:** Remove task-irrelevant information from the current GUI state.
2. **LLM-based Computer Agent:** Fine-tuned LLM for next action prediction.

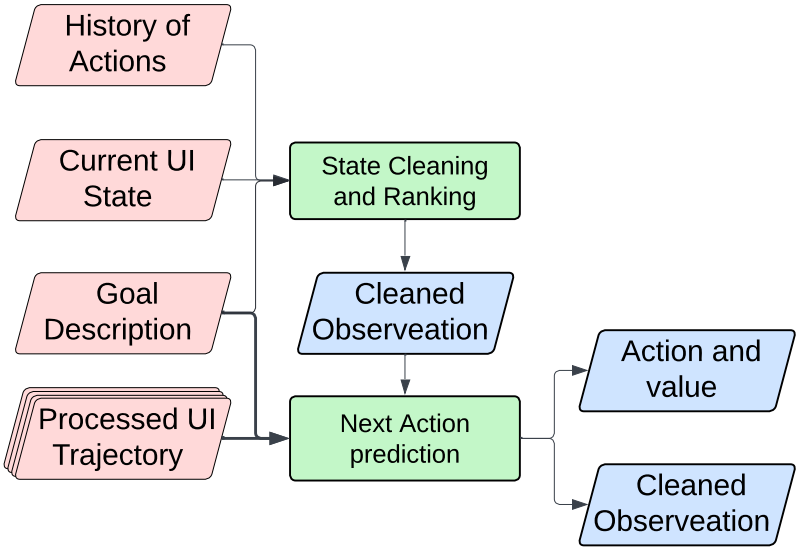

Figure 5.1: Next Action Prediction Pipeline

5.1.1 GUI Environment Cleaning with Small LMs

The primary objective of this step is to streamline the current GUI state by eliminating all task-irrelevant information and UI elements. This approach addresses the first limitation of LLM-based computer agents, which is the limited context length.

Previous works have proposed different methods to tackle this issue, SYNAPSE [80] has designed a state-abstraction component. It takes advantage of the few-shot learning ability of LLMs to include only the task-relevant information from raw states and output a clean state, namely state observation, for each action along the trajectory. Although this method has worked and yielded promising results, it does not guarantee that the observation can fit in the context length of the LLM-based computer agent. They also applied their method on a relatively simpler dataset, namely MiniWob++ [39]. To build on the work of SYNAPSE, our experiments apply a similar learnable state-abstraction approach but adapt it to more complex datasets and scenarios, demonstrating the effectiveness of our method in managing context length while maintaining task performance in environments that pose greater challenges than those addressed in MiniWob++.

Inspired by the work from MindAct [15], we approached this problem as a ranking task. Given the task description, the filtered GUI state at step t, and the history of previous

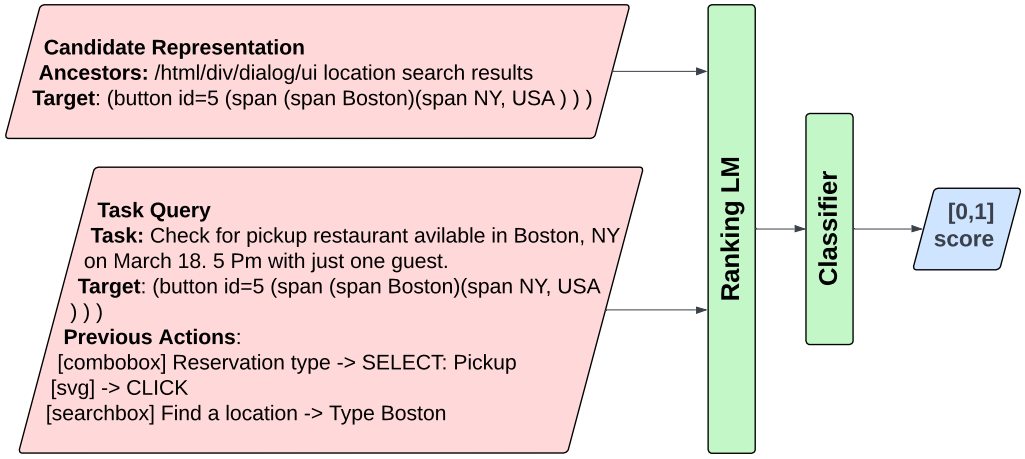

Figure 5.2: GUI State Cleaning and Ranking Pipeline

actions till step $t-1$, we select top-k DOM elements from the current preprocessed state, check Chapter 3 for dataset preprocessing. The task query is constructed by concatenating the task description and the history of previous actions. For each element in the preprocessed state, we recursively build textual representation from the element's tag, its textual content, salient attribute values, and the textual representation of both its direct parent and its child elements. As illustrated in Figure 5.2, we obtain the final classification score for each DOM candidate by feeding the task query and each candidate to an encoder-only LM through the cross-encoder architecture [42]. A cross-encoder architecture processes the task query and the candidate jointly, enabling the model to deeply analyze the interaction between them. Unlike other architectures that encode inputs separately, the cross-encoder directly compares the query and candidate in a shared representation space, resulting in a matching score between the DOM element and the task query. Afterward, The matching score is passed through a sigmoid activation function and optimized with a binary cross-entropy loss.

5.1.2 Next Action Prediction Using LLM

After successfully implementing the GUI cleaning component, we leverage it to streamline the current GUI state, keeping only task-relevant information. This refined state is then used as input for our LLM to predict the next action. Given the task description and

the refined UI trajectories from step $t = 0$ till step $t - 1$, where each trajectory consists of the action performed at step t' where $t' < t$ and the cleaned observation at step $t' - 1$, we formulated the next action prediction as a question answering task.

Next Action prediction As Open-Ended Question Answering

In our approach, we have formulated the next action prediction as an open-ended question-answering task. We fine-tuned Mistral-7B to familiarize it with our use case and give it a better understanding of both the input prompt and the structure of different GUI structures.

As illustrated in Figure 5.3, our prompt consists of 4 key components. First, the basic instruction explains the context of the task to the LLM and specifies what needs to be accomplished. Then, the trajectories from step $t = 0$ to step $t - 1$, the history of performed actions. Lastly, the task query that should be followed to reach our final output.

Next Action Prediction As Multi-Choice Question Answering

Previous works [13, 21] have discovered that fine-tuning LLMs for discrimination tasks is more effective than the generation task when it comes to grounding tasks such as picking the right DOM element for the next action prediction. This approach forces the model to focus more on element selection rather than on generating a meaningful answer, which leads to better precision in tasks that require accurate selection and identification. Inspired by these studies, We reformulated the DOM elements selection into multi-choice question answering instead of open-ended question answering. Therefore, we trained our model to pick the target element from a list of options instead of generating the full answer.

To achieve this enhancement, We have re-engineered our input to include a list of options where the LLM is fine-tuned to pick one of these options and then generate the most suitable action, and value pair for the current time step. Figure 5.4 illustrates the inclusion of the option list in our prompt. In our study, We picked the top-k DOM elements from the current GUI observation, where $k = 25$.

For generalization purposes and to test our model understanding and grounding capabilities, we have included *none of the above options*. This addition verifies that the model doesn't generate answers randomly or feel compelled to choose an option because it is required to do so. By including this option, we can assess whether the model comprehends the context and makes informed decisions, rather than by default select an available option.

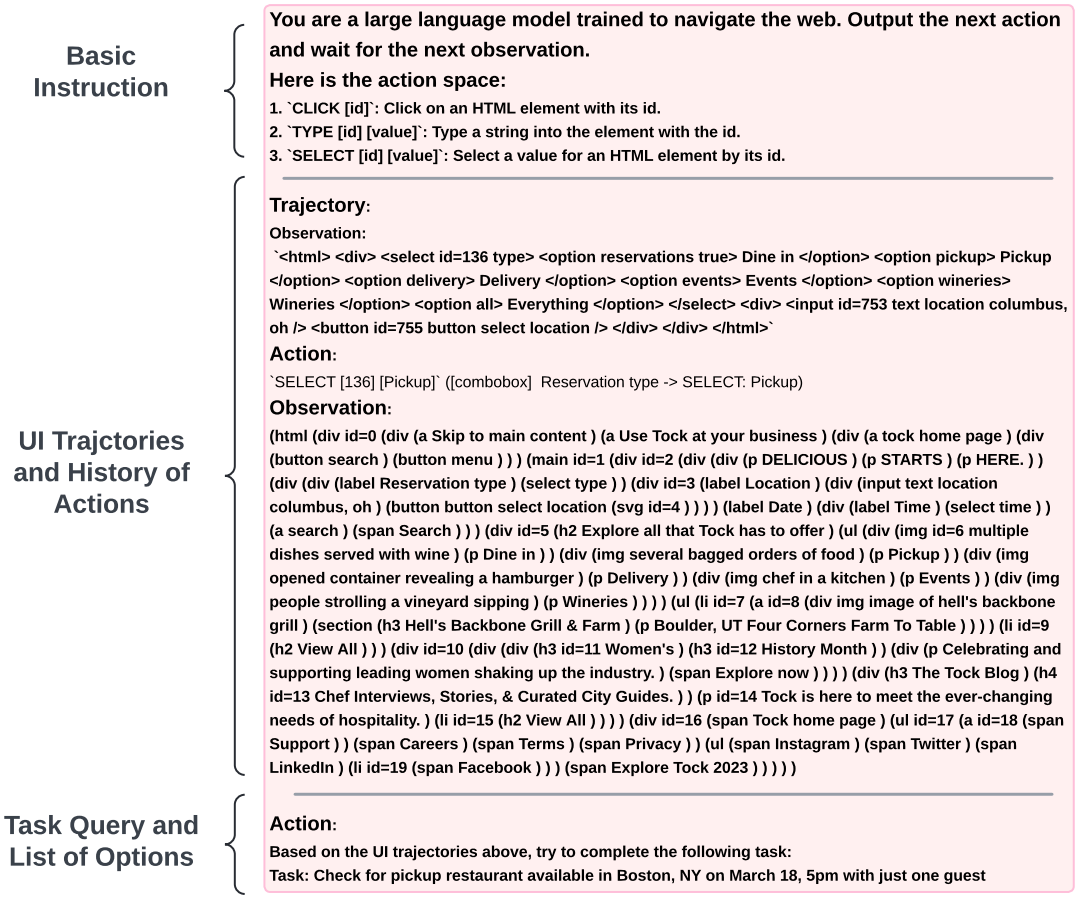

Figure 5.3: Prompt for next action prediction as open-ended question answering

As mentioned before by Matino et al. [42], rich prompts can distract the LLM from the original task and potentially lead it to hallucinate. To mitigate this issue, we implement an algorithm, presented in 5.1. Initially, we randomly select the first k' options and add the **None Of The Above** choice to this list. We then iteratively prompt our fine-tuned LLM, constructing a new set of selected options with each iteration. Finally, we make a last LLM call using this refined set to obtain the final answer. This process is repeated until all options have been evaluated. In the final step, the LLM is tasked with computing the final answer from the aggregated result set. This iterative method ensures that the LLM can handle complex decision-making processes by breaking down the task into manageable chunks, thus improving the accuracy and reliability of the predictions [15].

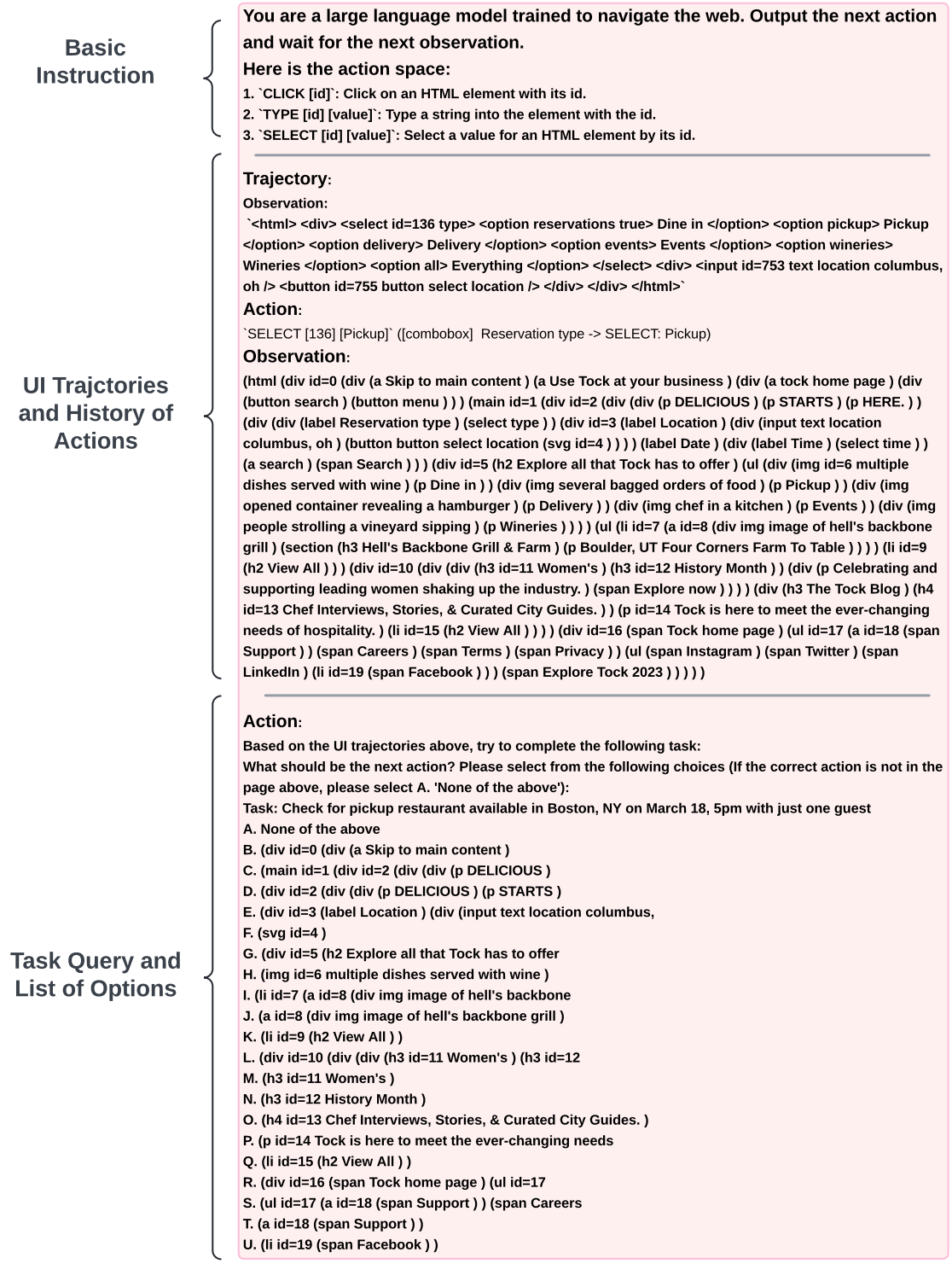

Figure 5.4: Prompt for next action prediction as multi-choices question answering

Algorithmus 5.1 Iterative LLM-agent Inference Algorithm

Input: set of options O of length k, LLM-Based agent G, hyper-parameter k'
Output: Final answer F
$R \leftarrow \emptyset$
while O is not empty **do**
    $Temp \leftarrow O[: k' + 1]$
    $Temp \leftarrow Temp \cup \{$"A. None of the Above"$\}$
    $O \leftarrow O[k' + 1 :]$
    $O' \leftarrow G(Temp)$
    $R \leftarrow R \cup \{O'\}$
end while
$F \leftarrow G(R)$
return F

5.1.3 Implementation Details

Similar to Chapter 4, we opted to use Mistral-7B architecture. Despite its relatively small size, only 7 billion parameters, Mstral-7B boasts a substantial context length of $8k$ tokens, which is double the context length of LLaMa-2-7B [62] and also exceeds other open-source LLM with the same number of parameters, such as Falcon-7B [3] and Baichuan 2-7B [75]. Moreover, it uses SWA which allows it to extend its context length up to $32k$ tokens, which is critical to tackle further the previously mentioned issue of the limited context length. Additionally, our iteratively inference algorithm explained in Algorithm 5.1, can slow down the inference process due to multiple LLM calls, however, the utilization of GQA helps speed up the inference process.

For the fine-tuning details and hyper-parameters, please refer to Section 4.3.2.

5.2 Main Results

In this section, we present the outcomes of the fine-tuning process applied to Mistral-7B. Our focus is on the evaluation of the model's performance on the Mind2Web dataset, assessing the impact of our adaptations. We apply our method on 3 non-overlapping datasets, namely cross-domain, cross-website, and cross-task, to examine its performance and generalizability.

Moreover, we perform both quantitative and qualitative analyses to provide a detailed assessment of our model. The quantitative analysis focuses on numerical metrics,

while the qualitative analysis examines the model's predictions in various scenarios to understand its behavior better.

Furthermore, we include the results of these metrics when formulating the task as open-ended questions versus multiple-choice question answers. This comparison highlights the effectiveness of our approach in different contexts and underscores the robustness of our model.

5.2.1 Evaluation Metrics

In this section, we present the evaluation metrics used to assess the prediction accuracy of our model. As illustrated in Figure 5.1, our final output comprises two components: the next action and the target DOM element. For comparative purposes, we adapted two evaluation metrics that are commonly utilized in both Mind2Act and SYNAPSE:

- **Element Accuracy (EA):** This metric compares the selected DOM element with all acceptable elements, ensuring that the correct element is identified within the GUI.
- **Operation F1 Score:** This metric calculates the token-level F1 score for the predicted actions, providing a measure of precision and recall for the sequence of actions.

Adapting these metrics ensures a comprehensive evaluation of both components of our output. Additionally, this approach allows us to benchmark our model's performance against established methods in the field.

5.2.2 Quantitative Results

Result of Next Action Prediction as Open-Ended Question

We can see in Table 5.1, our adapted method has significantly better results compared to MindAct-GPT and SYNAPSE-GPT. While GPT-3 has 175 billion parameters and GPT-4 has 1.74 trillion parameters, our method based on Mistral-7B has achieved better results in all metrics across the 3 different test datasets. with an increase of 10.70% in Cross-Task, 13.40% in Cross-Website, and 42.90% in Cross-Domain compared to MindAct-GPT3.5. Additionally, compared to SYNAPSE-GPT, our method showed an increase of 18.30% in Cross-Task, 3.60% in Cross-Website, and 34.90% in Cross-Domain. As a side finding, we can conclude that using larger LLM doesn't necessarily leads to better performance, as discovered also by [41].

	Cross-Task		Cross-Website		Cross-Domain	
	Ele. Acc	**Op. F1**	**Ele. Acc**	**Op. F1**	**Ele. Acc**	**Op. F1**
MindAct						
w/ GPT-3.5	20.30	56.60	19.30	48.80	21.60	52.80
w/ GPT-4*	41.60	60.60	**35.80**	51.10	37.10	46.50
Synapse						
w/ GPT-3.5	34.00	-	29.10	-	29.60	-
Ours						
w/ Mistral-7B	**52.30**	**74.30**	32.70	**60.60**	**64.50**	**59.80**

Table 5.1: Comparing our method to MindAct and SYNAPSE

The results obtained under the inclusion of the full trajectories and the cleaning and ranking model show an improvement across all clinical metrics. This enhancement aligns with our earlier hypotheses, providing support for the effectiveness of our model. It shows our model's capability in increasing the accuracy of the next action perdition task. These findings highlight the success of our approach in boosting the performance of existing task automation agents.

Result of Next Action Prediction as Multi-Choice Question

In this section, we will present the results of formulating our DOM element selection into multi-choice question answering instead of open-ended question answering. As shown in Table 5.2, we can notice an improvement in the element accuracy of 4.10% in the Cross-Task, 5.20% in the Cross-Website, and 1.60% in Cross-Domain compared to our method as a generation task. As anticipated, this conversion helped improve the model to select the target DOM element which by nature helped improve the action accuracy.

We achieved improvements in the model performance metrics, which confirms that our method successfully contributed to enhancing the next action prediction. Essentially, the application of our method resulted in a more accurate identification of target DOM elements, underscoring the effectiveness of formulating it to a discrimination task instead of a generative task.

In the next chapter, we will go into the qualitative results showing the effect of our adapted framework on Mind2Web. This is important since in the end, the most important metric is predicting the target element along with the action to be performed.

	Cross-Task		Cross-Website		Cross-Domain	
	Ele. Acc	**Op. F1**	**Ele. Acc**	**Op. F1**	**Ele. Acc**	**Op. F1**
MindAct						
w/ GPT-3.5	20.30	56.60	19.30	48.80	21.60	52.80
w/ GPT-4*	41.60	60.60	35.80	51.10	37.10	46.50
Synapse						
w/ GPT-3.5	34.00	-	29.10	-	29.60	-
Ours						
Generation	52.30	74.30	32.70	60.60	64.50	59.80
Discrimination	**56.40**	**74.60**	**37.90**	**63.30**	**66.10**	**60.00**

Table 5.2: Results of open-ended question vs multi-choice question

5.2.3 Qualitative Results

As we can see in Figure 5.5 and Figure 5.6, our method can successfully comprehend our task and predict the next action and the target DOM element. While reformulating the task enhanced the performance of selecting the target element, we can spot the drawback which is the longer prompt. Although we incorporate the cleaning and ranking component, we still can reach the maximum context with the increase in the history of actions. Reaching the maximum context refers to the model's inability to process additional input due to the length limitations of the context window. Subsequently, this can lead to a poor performance as important information may be truncated to force our prompt to fit within the context window boundaries.

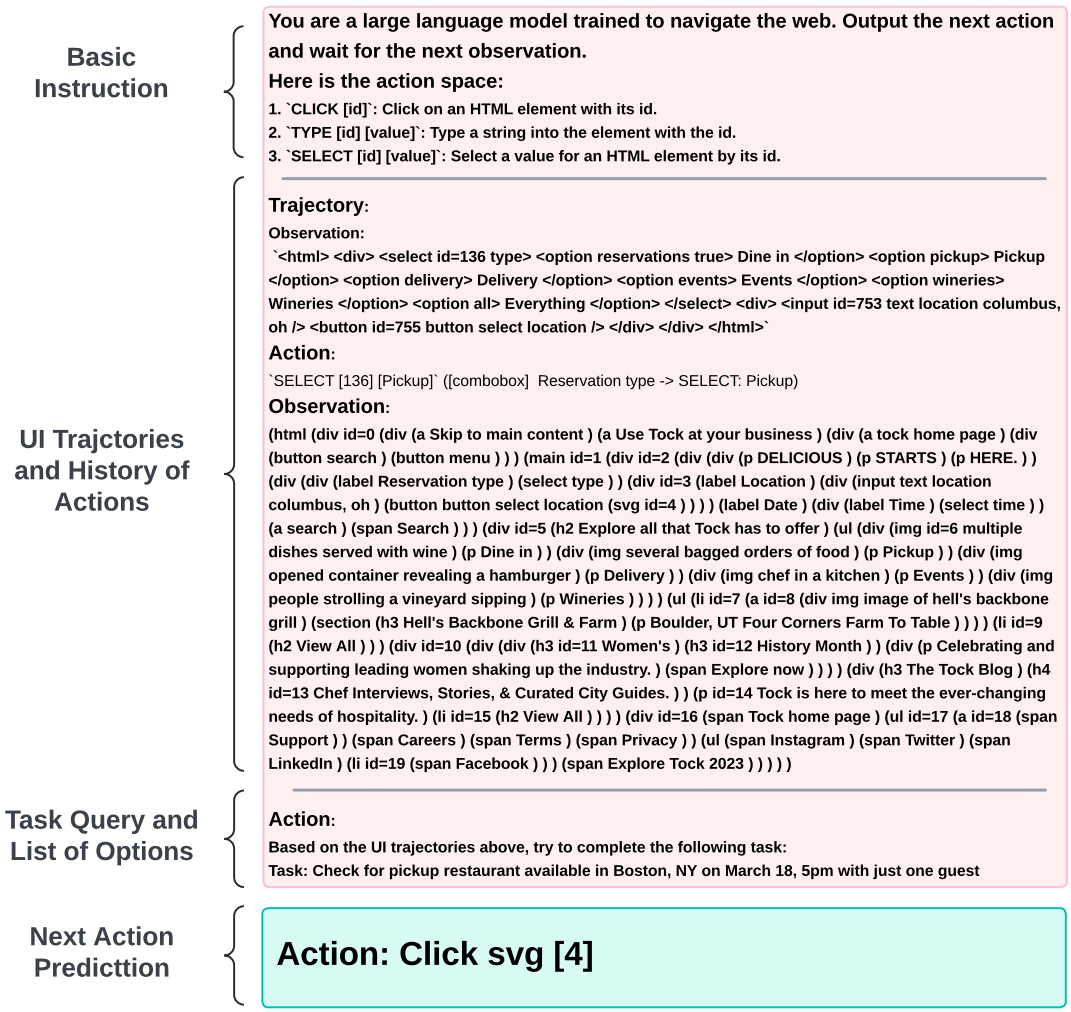

Figure 5.5: next action prediction as a generative task

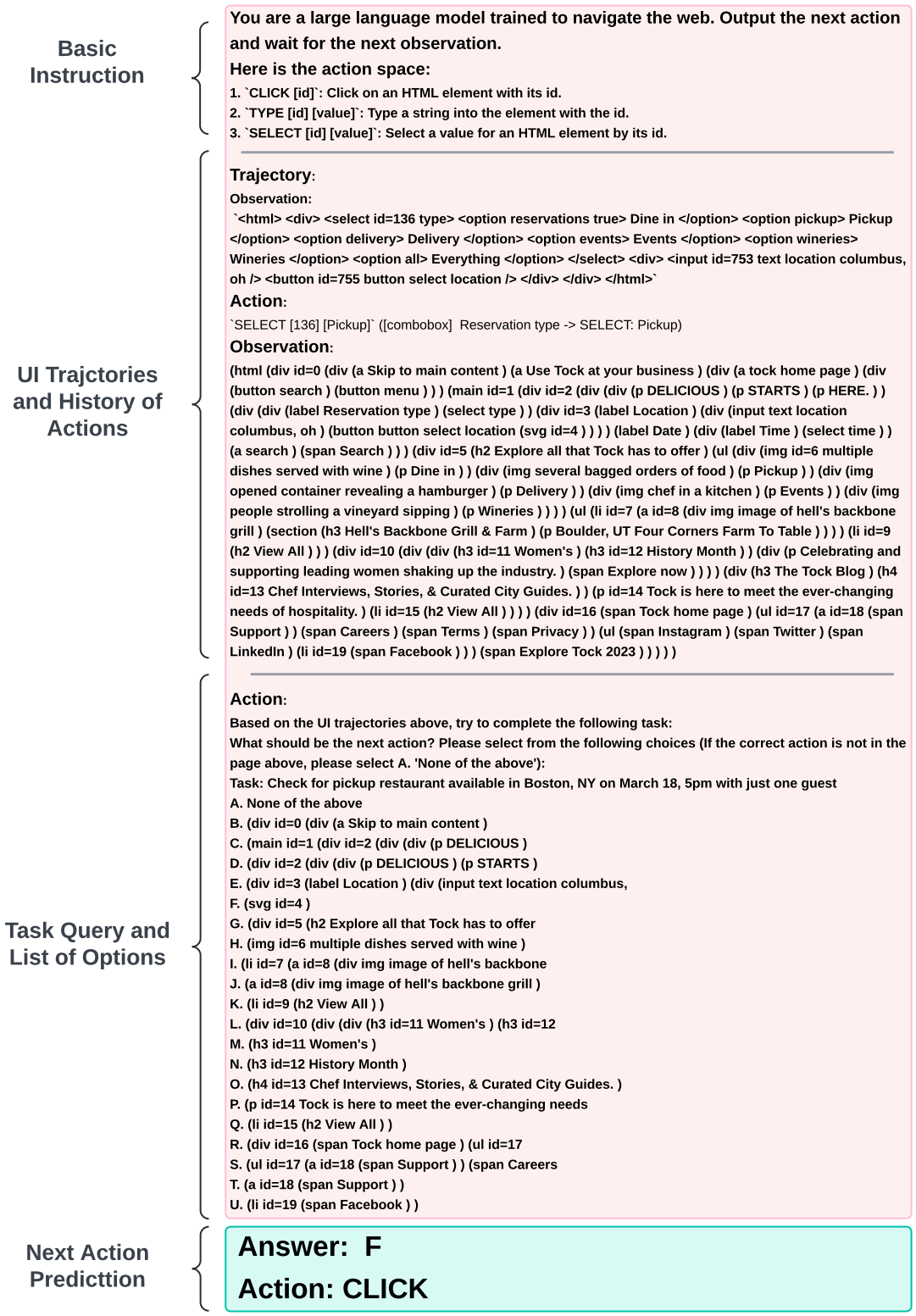

Figure 5.6: next action prediction as a discriminative task

6 Conclusion

In this work, we are the first to explore the challenging task of human behavior summarization from interaction in different real-world GUI environments, such as websites and Andriod applications. We demonstrated the potential of LLMs to enhance human-computer interaction through the accurate recognition of latent user intentions and the prediction of subsequent actions within graphical user interfaces. This study not only addresses the challenges inherent in modeling complex user behaviors but also significantly improves the performance metrics over existing approaches.

By developing the novel **Mistral-Intention**, we managed to understand and summarize different user interactions, thus providing a robust method for understanding and summarizing the latent intention from interactions. The results from this research show a remarkable improvement in the accuracy and efficiency of intention recognition and next action prediction, underscoring the effectiveness of integrating LLMs with HCI systems.

A Key factor in enhancing our model's performance was the integration of a keywords-based loss within the training loop. This modification forced the model to focus more intensely on essential information within the sequence of actions performed. As a result, the model could extract more relevant information from the input prompts, substantially improving the semantic accuracy of the recognized intentions. Extensive experiments demonstrated the effectiveness of Mistral-Intention, with summarized latent intention across different datasets from different domains.

Furthermore, We introduced an LLM-based automation agent capable of predicting the next action within an GUI environment as a demonstration for the potential application of Mistral-Intention. Our agent addressed the challenges faced by other LLM-based automation agents which is the limited context length and the unrecognized complex structure of the current environment.

Initially, the integration of a ranking and classification model refine the raw state by filtering out irrelevant information. This step significantly reduced the number of input tokens, allowing the current step's information to be accommodated within a single prompt, thereby facilitating faster responses. Additionally, incorporating user trajectories up to the current step into the input prompts enhanced the model's understanding of

the existing environment. This enrichment significantly boosted its multi-step decision-making capabilities.

While our study marks a significant step forward, achieving summarized intention comparable to human level understanding remains a future goal. A primary limitation of our study is the model's confusion in scenarios with default GUI settings, like auto-selecting a user's current city on an airline's website without explicit human input. A proposed solution for such limitation is the integration of multi-modal data such as image and text. Another limitation of Mind2Web is the lack of causal dependency between actions, which leads to significant number false negatives in predicting next action. A potential solution to this problem could be the integration of a graph-based dependency model among the actions. Moreover, verifying that our model's summarized intentions, when used as inputs for a task automation agent can reliably replicate the intended sequence of actions. Our work lays the foundation for further advancements in both behavior summatization from interactions and LLM-based agents, especially in HCI domains.

Bibliography

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. “Gpt-4 technical report.” In: *arXiv preprint arXiv:2303.08774* (2023) (cit. on pp. 17, 19, 20, 41).

[2] R. Agrawal, A. Habeeb, C.-H. Hsueh. “Learning user intent from action sequences on interactive systems.” In: *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018 (cit. on p. 18).

[3] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, É. Goffinet, D. Hesslow, J. Launay, Q. Malartic, et al. “The falcon series of open language models.” In: *arXiv preprint arXiv:2311.16867* (2023) (cit. on p. 53).

[4] A. Almehmadi. “Micro-Behavioral Accidental Click Detection System for Preventing Slip-Based Human Error.” In: *Sensors* 21.24 (2021), p. 8209 (cit. on p. 27).

[5] J. Andreas. “Language models as agent models.” In: *arXiv preprint arXiv:2212.01681* (2022) (cit. on p. 17).

[6] S. Banerjee, A. Lavie. “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments.” In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 2005, pp. 65–72 (cit. on p. 37).

[7] R. Bednarik, H. Vrzakova, M. Hradis. “What do you want to do next: a novel approach for intent prediction in gaze-based interaction.” In: *Proceedings of the symposium on eye tracking research and applications*. 2012, pp. 83–90 (cit. on pp. 18, 27).

[8] O. Berkovitch, S. Caduri, N. Kahlon, A. Efros, A. Caciularu, I. Dagan. “Identifying User Goals from UI Trajectories.” In: *arXiv preprint arXiv:2406.14314* (2024) (cit. on pp. 20, 39, 41, 42).

[9] A. Bodonhelyi, E. Bozkir, S. Yang, E. Kasneci, G. Kasneci. “User intent recognition and satisfaction with large language models: A user study with chatgpt.” In: *arXiv preprint arXiv:2402.02136* (2024) (cit. on p. 19).

[10] A. Burns, D. Arsan, S. Agrawal, R. Kumar, K. Saenko, B. A. Plummer. "A Dataset for Interactive Vision Language Navigation with Unknown Command Feasibility." In: *European Conference on Computer Vision (ECCV)*. 2022 (cit. on pp. 20, 23, 25).

[11] Y. Y. Chiu, A. Sharma, I. W. Lin, T. Althoff. "A computational framework for behavioral assessment of llm therapists." In: *arXiv preprint arXiv:2401.00820* (2024) (cit. on p. 15).

[12] D. Chudá, P. Krátky. "Usage of computer mouse characteristics for identification in web browsing." In: *Proceedings of the 15th International Conference on Computer Systems and Technologies*. 2014, pp. 218–225 (cit. on p. 17).

[13] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. "Scaling instruction-finetuned language models." In: *Journal of Machine Learning Research* 25.70 (2024), pp. 1–53 (cit. on p. 50).

[14] H. Dai, Z. Liu, W. Liao, X. Huang, Y. Cao, Z. Wu, L. Zhao, S. Xu, W. Liu, N. Liu, et al. "Auggpt: Leveraging chatgpt for text data augmentation." In: *arXiv preprint arXiv:2302.13007* (2023) (cit. on p. 30).

[15] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, Y. Su. *Mind2Web: Towards a Generalist Agent for the Web*. 2023. arXiv: 2306.06070 [cs.CL] (cit. on pp. 20, 21, 23, 47, 48, 51).

[16] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on p. 33).

[17] A. Elbahi, M. A. Mahjoub, M. N. Omri. "Hidden markov model for inferring user task using mouse movement." In: *Fourth International Conference on Information and Communication Technology and Accessibility (ICTA)*. IEEE. 2013, pp. 1–7 (cit. on p. 17).

[18] P. M. Fitts. "The information capacity of the human motor system in controlling the amplitude of movement." In: *Journal of experimental psychology* 47.6 (1954), p. 381 (cit. on p. 15).

[19] E. Y. Fu, T. C. Kwok, E. Y. Wu, H. V. Leong, G. Ngai, S. C. Chan. "Your mouse reveals your next activity: towards predicting user intention from mouse interaction." In: *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. IEEE. 2017, pp. 869–874 (cit. on p. 17).

[20] M. Grootendorst. *KeyBERT: Minimal keyword extraction with BERT*. Version v0.3.0. 2020. DOI: 10.5281/zenodo.4461265. URL: https://doi.org/10.5281/zenodo.4461265 (cit. on p. 33).

[21] Y. Gu, X. Deng, Y. Su. "Don't Generate, Discriminate: A Proposal for Grounding Language Models to Real-World Environments." In: *arXiv preprint arXiv:2212.09736* (2022) (cit. on p. 50).

[22] I. Gur, H. Furuta, A. Huang, M. Safdari, Y. Matsuo, D. Eck, A. Faust. “A real-world webagent with planning, long context understanding, and program synthesis.” In: *arXiv preprint arXiv:2307.12856* (2023) (cit. on p. 11).

[23] S. Haque, Z. Eberhart, A. Bansal, C. McMillan. “Semantic similarity metrics for evaluating source code summarization.” In: *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 2022, pp. 36–47 (cit. on p. 38).

[24] W. E. Hick. “On the rate of gain of information.” In: *Quarterly Journal of experimental psychology* 4.1 (1952), pp. 11–26 (cit. on p. 15).

[25] S. Hochreiter, J. Schmidhuber. “Long short-term memory.” In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on pp. 12, 16).

[26] W. Hong, W. Wang, Q. Lv, J. Xu, W. Yu, J. Ji, Y. Wang, Z. Wang, Y. Dong, M. Ding, et al. “Cogagent: A visual language model for gui agents.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 14281–14290 (cit. on p. 11).

[27] P. C. Humphreys, D. Raposo, T. Pohlen, G. Thornton, R. Chhaparia, A. Muldal, J. Abramson, P. Georgiev, A. Santoro, T. Lillicrap. “A data-driven approach for learning to control computers.” In: *International Conference on Machine Learning*. PMLR. 2022, pp. 9466–9482 (cit. on p. 27).

[28] P. C. Humphreys, D. Raposo, T. Pohlen, G. Thornton, R. Chhaparia, A. Muldal, J. Abramson, P. Georgiev, A. Santoro, T. Lillicrap. “A data-driven approach for learning to control computers.” In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 9466–9482. URL: https://proceedings.mlr.press/v162/humphreys22a.html (cit. on p. 21).

[29] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. “Mistral 7B.” In: *arXiv preprint arXiv:2310.06825* (2023) (cit. on p. 27).

[30] J. Jiang, E. Ferrara. “Social-LLM: Modeling User Behavior at Scale using Language Models and Social Network Data.” In: *arXiv preprint arXiv:2401.00893* (2023) (cit. on pp. 15, 17).

[31] W. S. El-Kassas, C. R. Salama, A. A. Rafea, H. K. Mohamed. “Automatic text summarization: A comprehensive survey.” In: *Expert systems with applications* 165 (2021), p. 113679 (cit. on pp. 33, 36).

[32] K. S. Killourhy, R. A. Maxion. “Comparing anomaly-detection algorithms for keystroke dynamics.” In: *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE. 2009, pp. 125–134 (cit. on p. 16).

[33] P. Kumar, M. Perrollaz, S. Lefevre, C. Laugier. “Learning-based approach for online lane change intention prediction.” In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2013, pp. 797–802 (cit. on pp. 11, 18).

[34] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. “Natural questions: a benchmark for question answering research.” In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 453–466 (cit. on pp. 33, 36).

[35] Y. Li, S. Bengio, G. Bailly. “Predicting human performance in vertical menu selection using deep learning.” In: *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018, pp. 1–7 (cit. on p. 16).

[36] Y. Li, J. He, X. Zhou, Y. Zhang, J. Baldridge. “Mapping natural language instructions to mobile UI action sequences.” In: *arXiv preprint arXiv:2005.03776* (2020) (cit. on p. 25).

[37] J. Liebers, S. Brockel, U. Gruenefeld, S. Schneegass. “Identifying users by their hand tracking data in augmented and virtual reality.” In: *International Journal of Human–Computer Interaction* 40.2 (2024), pp. 409–424 (cit. on p. 16).

[38] C.-Y. Lin. “Rouge: A package for automatic evaluation of summaries.” In: *Text summarization branches out*. 2004, pp. 74–81 (cit. on p. 36).

[39] E. Z. Liu, K. Guu, P. Pasupat, T. Shi, P. Liang. “Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration.” In: *International Conference on Learning Representations (ICLR)*. 2018. URL: https://arxiv.org/abs/1802.08802 (cit. on pp. 20, 21, 25, 48).

[40] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, T. Zhou. “Recommender systems.” In: *Physics reports* 519.1 (2012), pp. 1–49 (cit. on p. 27).

[41] J. Mahapatra, U. Garain. “Impact of Model Size on Fine-tuned LLM Performance in Data-to-Text Generation: A State-of-the-Art Investigation.” In: *arXiv preprint arXiv:2407.14088* (2024) (cit. on p. 54).

[42] A. Martino, M. Iannelli, C. Truong. “Knowledge injection to counter large language model (LLM) hallucination.” In: *European Semantic Web Conference*. Springer. 2023, pp. 182–185 (cit. on pp. 47, 49, 51).

[43] G. Marvin, N. Hellen, D. Jjingo, J. Nakatumba-Nabende. “Prompt engineering in large language models.” In: *International conference on data intelligence and cognitive informatics*. Springer. 2023, pp. 387–402 (cit. on p. 42).

[44] ModelCenter. *ModelCenter*. Accessed: 2023-07-23. 2023. URL: https://github.com/OpenBMB/ModelCenter (cit. on p. 34).

[45] L. Ning, L. Liu, J. Wu, N. Wu, D. Berlowitz, S. Prakash, B. Green, S. O'Banion, J. Xie. "User-LLM: Efficient LLM Contextualization with User Embeddings." In: *arXiv preprint arXiv:2402.13598* (2024) (cit. on pp. 15, 17).

[46] Y. Nishimura, Y. Nakamura, H. Ishiguro. "Human interaction behavior modeling using generative adversarial networks." In: *Neural Networks* 132 (2020), pp. 521–531 (cit. on pp. 15, 16).

[47] K. Papineni. "BLEU: a method for automatic evaluation of MT." In: *Research Report, Computer Science RC22176 (W0109-022)* (2001) (cit. on p. 37).

[48] R. Pascanu, T. Mikolov, Y. Bengio. "On the difficulty of training recurrent neural networks." In: *International conference on machine learning*. Pmlr. 2013, pp. 1310–1318 (cit. on p. 12).

[49] D. J. Phillips, T. A. Wheeler, M. J. Kochenderfer. "Generalizable intention prediction of human drivers at intersections." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1665–1670. DOI: 10.1109/IVS.2017.7995948 (cit. on pp. 11, 18, 38).

[50] D. J. Phillips, T. A. Wheeler, M. J. Kochenderfer. "Generalizable intention prediction of human drivers at intersections." In: *2017 IEEE intelligent vehicles symposium (IV)*. IEEE. 2017, pp. 1665–1670 (cit. on p. 18).

[51] *Playwright*. https://playwright.dev/. Accessed: 2024-08-04 (cit. on p. 24).

[52] P. Prajod, M. Lavit Nicora, M. Malosio, E. André. "Gaze-based attention recognition for human-robot collaboration." In: *Proceedings of the 16th International Conference on PErvasive Technologies Related to Assistive Environments*. 2023, pp. 140–147 (cit. on p. 18).

[53] A. S. Rao, M. P. Georgeff, et al. "BDI agents: from theory to practice." In: *Icmas*. Vol. 95. 1995, pp. 312–319 (cit. on p. 17).

[54] T. Shi, A. Karpathy, L. Fan, J. Hernandez, P. Liang. "World of Bits: An Open-Domain Platform for Web-Based Agents." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup, Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3135–3144. URL: https://proceedings.mlr.press/v70/shi17a.html (cit. on pp. 21, 23).

[55] M. Soleymani, M. Riegler, P. Halvorsen. "Multimodal analysis of image search intent: Intent recognition in image search from user behavior and visual content." In: *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. 2017, pp. 251–259 (cit. on p. 19).

[56] H. Steck, C. Ekanadham, N. Kallus. "Is cosine-similarity of embeddings really about similarity?" In: *Companion Proceedings of the ACM on Web Conference 2024*. 2024, pp. 887–890 (cit. on p. 38).

[57] G. Sun, J. Lin, Y. Zhang, Y. Wong, M. S. Kankanhalli. "EMAKI: An Efficient Mouse and Keyboard Interaction Logging System." In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2019, pp. 2893–2896 (cit. on p. 16).

[58] L. Sun, X. Chen, L. Chen, T. Dai, Z. Zhu, K. Yu. "Meta-gui: Towards multi-modal conversational agents on mobile gui." In: *arXiv preprint arXiv:2205.11029* (2022) (cit. on p. 25).

[59] Z. Sun, H. Liu, X. Qu, K. Feng, Y. Wang, Y. S. Ong. "Large language models for intent-driven session recommendations." In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 324–334 (cit. on p. 19).

[60] A. Swearngin, Y. Li. "Modeling mobile interface tappability using crowdsourcing and deep learning." In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–11 (cit. on p. 16).

[61] S. Tian, X. Liang, M. Zheng. "An Optimization-Based Human Behavior Modeling and Prediction for Human-Robot Collaborative Disassembly." In: *2023 American Control Conference (ACC)*. 2023, pp. 3356–3361. DOI: 10.23919/ACC55779.2023.10156342 (cit. on p. 16).

[62] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models." In: *arXiv preprint arXiv:2307.09288* (2023) (cit. on pp. 34, 53).

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 12, 19).

[64] S. Wang, T. Xu, H. Li, C. Zhang, J. Liang, J. Tang, P. S. Yu, Q. Wen. "Large language models for education: A survey and outlook." In: *arXiv preprint arXiv:2403.18105* (2024) (cit. on p. 15).

[65] W. Wang, R. Li, Y. Chen, Y. Jia. "Human intention prediction in human-robot collaborative tasks." In: *Companion of the 2018 ACM/IEEE international conference on human-robot interaction*. 2018, pp. 279–280 (cit. on pp. 11, 18).

[66] J. Wei, K. Zou. "Eda: Easy data augmentation techniques for boosting performance on text classification tasks." In: *arXiv preprint arXiv:1901.11196* (2019) (cit. on p. 29).

[67] H. Wen, Y. Li, G. Liu, S. Zhao, T. Yu, T. J.-J. Li, S. Jiang, Y. Liu, Y. Zhang, Y. Liu. "Autodroid: Llm-powered task automation in android." In: *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 2024, pp. 543–557 (cit. on p. 22).

[68] H. Wen, Y. Li, G. Liu, S. Zhao, T. Yu, T. J.-J. Li, S. Jiang, Y. Liu, Y. Zhang, Y. Liu. "Empowering llm to use smartphone for intelligent task automation." In: *arXiv preprint arXiv:2308.15272* (2023) (cit. on p. 27).

[69] H. Wen, H. Wang, J. Liu, Y. Li. "Droidbot-gpt: Gpt-powered ui automation for android." In: *arXiv preprint arXiv:2304.07061* (2023) (cit. on pp. 22, 27).

[70] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, G. Mann. "Bloomberggpt: A large language model for finance." In: *arXiv preprint arXiv:2303.17564* (2023) (cit. on p. 15).

[71] T. Wu, S. He, J. Liu, S. Sun, K. Liu, Q.-L. Han, Y. Tang. "A brief overview of ChatGPT: The history, status quo and potential future development." In: *IEEE/CAA Journal of Automatica Sinica* 10.5 (2023), pp. 1122–1136 (cit. on p. 24).

[72] C. Xie, C. Chen, F. Jia, Z. Ye, K. Shu, A. Bibi, Z. Hu, P. Torr, B. Ghanem, G. Li. "Can Large Language Model Agents Simulate Human Trust Behaviors?" In: *arXiv preprint arXiv:2402.04559* (2024) (cit. on pp. 15, 17).

[73] N. Xu, S. Masling, M. Du, G. Campagna, L. Heck, J. Landay, M. S. Lam. "Grounding open-domain instructions to automate web support tasks." In: *arXiv preprint arXiv:2103.16057* (2021) (cit. on p. 25).

[74] L. Yan, X. Gao, X. Zhang, S. Chang. "Human-robot collaboration by intention recognition using deep lstm neural network." In: *2019 IEEE 8th International Conference on Fluid Power and Mechatronics (FPM)*. IEEE. 2019, pp. 1390–1396 (cit. on p. 18).

[75] A. Yang, B. Xiao, B. Wang, B. Zhang, C. Bian, C. Yin, C. Lv, D. Pan, D. Wang, D. Yan, et al. "Baichuan 2: Open large-scale language models." In: *arXiv preprint arXiv:2309.10305* (2023) (cit. on p. 53).

[76] S. Yao, H. Chen, J. Yang, K. Narasimhan. "Webshop: Towards scalable real-world web interaction with grounded language agents." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 20744–20757 (cit. on pp. 23, 25).

[77] K. M. Yoo, D. Park, J. Kang, S.-W. Lee, W. Park. "Gpt3mix: Leveraging large-scale language models for text augmentation." In: *arXiv preprint arXiv:2104.08826* (2021) (cit. on p. 30).

[78] G. Zhang, S. Hindennach, J. Leusmann, F. Bühler, B. Steuerlein, S. Mayer, M. Bâce, A. Bulling. "Predicting Next Actions and Latent Intents during Text Formatting." In: *Proceedings of the CHI Workshop Computational Approaches for Understanding, Generating, and Adapting User Interfaces (2022-01-01)*. 2022, pp. 1–6 (cit. on pp. 11, 18, 19, 21).

[79] G. Zhang, Z. Hu, M. Bâce, A. Bulling. "Mouse2Vec: Learning Reusable Semantic Representations of Mouse Behaviour." In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–17 (cit. on p. 16).

[80] L. Zheng, R. Wang, X. Wang, B. An. “Synapse: Trajectory-as-Exemplar Prompting with Memory for Computer Control.” In: *The Twelfth International Conference on Learning Representations*. 2023 (cit. on pp. 13, 15, 22, 35, 48).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature

A Instructions and Prompts

You're given two sentences, A and B, representing user goals on a website or app, and a history of actions for Task B.
Here is the action space:
1. `CLICK [id]`: Click on an HTML element with its id.
2. `TYPE [id] [value]`: Type a string into the element with the id.
3. `SELECT [id] [value]`: Select a value for an HTML element by its id

Your Task: Determine if A satisfies B.

Definitions:
- **Satisfaction:** A satisfies B if EVERY important information exists in A also exists in B. This means completing A logically leads to completing B.

Analysis Steps:
1. **Extract Requirements:** List requirements and constraints of A and B in a table, noting .
2. **B constraint adjusment:** Remove any requirment/constraint in B that do not appear in the list of actions.
3. Compare requirments: check if A satisfies each B requirement after adjustment. If any answer is no, A doesn't satisfy B.
2. **Rigorous Comparison:** Is each requirement in B met by A? If A has stricter requirements, it satisfies B. If A is looser, go to step 3.
3. **Counter-Example (Optional):** Find a case where requirement A is met while requirement B is not, considering dynamic content changes like prices and availability. Ignore static content like structure and checkboxes.
4. **Chain of Thought:** Clearly explain your reasoning, referencing the requirements. Explain how each requirement in B is logically fulfilled by A or how a counterexample shows it doesn't.
5. **Decision:** Write your answer as: [SATISFACTION] YES/NO [/SATISFACTION]

Key Considerations:
- **User Goals:** Focus on the user's underlying goals, not just literal words.
- **Hisotry of Actions:** We know that the provided history of Actions satisfies B. Which means perfroming these actions logically leads to completing B.
- Prioritize logical reasoning over assumptions.
- **Dynamic vs. Static Content:** Consider if content changes often (e.g., top movies) or stays the same (e.g., filtering by users rating). If A needs dynamic content and B static, A doesn't satisfy B.
- **Direction:** Only answer if A satisfies B (A -> B), not the reverse.
- **Relative vs. Specific Times:** When comparing relative times, such as "tomorrow," and specific times, such as "April 15th," it is acceptable to disregard the relative times or assume that they are acceptable.
- Time Ranges and Dates Matter: Time ranges (e.g., "tomorrow" vs. "the week of April 15th") are considered distinct. Explicit different dates are considered distinct.
- **General vs. Specific:** A specific task A can satisfy a more general task B if completing A fulfills B's requirements. If A is specific and B is general, A might satisfy B, but it depends on whether all conditions in B are fulfilled by A.
- If A is general and B is specific, A does not satisfy B.
- In the context of this task, we will treat the terms "buy," "add to cart," and "check out" as interchangeable.

Determine whether A satisfies B with the given history of actions. Input:

Figure A.1: Model instructions for evaluating Satisfaction relation between two task descriptions (A and B)

Example 1: Stricter A Satisfies Looser B
- A: Purchase a one-way ticket from NYC to LAX on Delta Airlines.
- B: Book a flight from New York to Los Angeles.
Analysis: A's requirements are a subset of B's. Purchasing a one-way flight from NYC to LAX operated by Delta Airlines will fulfill B requirements: Booking flight, NY to LA. [SATISFACTION] YES [/SATISFACTION]
Example 2: Default Values & Satisfaction
- A: Order a pepperoni pizza
- B: Order a large pepperoni pizza
- Trajectory: Website defaults are set to large size when ordering a pizza.
Analysis: B has stricter restrictions for a Large pizza size, but the trajectory shows the default pizza size is Large. A satisfies B because the default size fulfills the requirement. Any reasonable trajectory executing A, ordering a pizza (not requiring setting a specific pizza size) will satisfy B. [SATISFACTION] YES [/SATISFACTION]
Example 3: Dynamic Content & Non-Satisfaction
- A: Buy the cheapest flight to London.
- B: Book a flight to London for under $500.
- Trajectory: Shows the cheapest flight is currently $450.
Analysis: Executing A doesn't guarantee satisfying B, as prices can change. Counter-example: The price could rise above $500, fulfilling A but not B. [SATISFACTION] NO [/SATISFACTION]
Example 4: Website Structure and Satisfaction
- A: Find a 5-star hotel in Rome.
- B: Find a highly-rated hotel in Rome.
- Trajectory: Shows a filter for star ratings, with 5-stars being the highest option.
Analysis: Since the website structure doesn't allow for higher ratings than 5, A satisfies B. [SATISFACTION] YES [/SATISFACTION]
Example 5: Looser A Fails to Satisfy Specific B (Counter-Example)
- A: Book a room in a hotel with a pool in Paris.
- B: Reserve a room at the Hotel Ritz in Paris.
- Trajectory: User is booking, Ritz hotel in Paris. Trajectory shows Ritz hotel has a pool.
Analysis: A's requirement for a hotel with a pool is looser than B's requirement for Hotel Ritz. The user could choose a different hotel in Paris that has a pool, still fulfilling A but it won't fulfill B. [SATISFACTION] NO [/SATISFACTION]
Example 6: Stricter verb in A Satisfies Looser verb in B
- A: Sign-up for a Fastbreak program.
- B: Find information about the Fastbreak program.
Analysis: Signing-up for something necessitates finding information beforehand about this thing. [SATISFACTION] YES [/SATISFACTION]
Example 7: Looser verb in A Fails to Satisfy a Specific verb in B (Counter-Example)
- A: Find information about the Fastbreak program
- B: Sign-up for the Fastbreak program.
Analysis: A is less strict than B. A counterexample is simply finding the necessary information without enrolling in the program. [SATISFACTION] NO [/SATISFACTION]

Figure A.2: Few-Shot exemplars demonstrating the Satisfaction Relation Prompt (Figure B.1). These simplified examples highlight the nuances of task satisfaction in real-world scenarios.

B Mistral Intention Inference Time

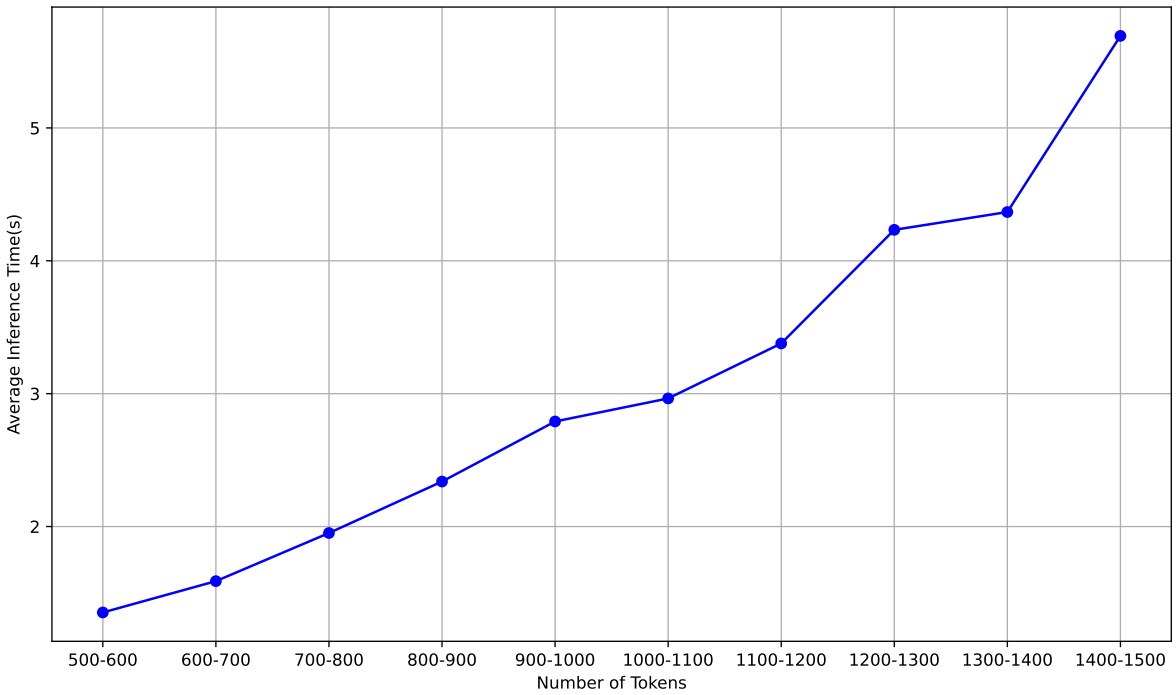

Figure B.1: Mistral-Intention inference time per number of tokens