B.Sc. in Computer Science and Technology Thesis

Evaluating the Multilingual Capabilities of LLM-based Web UI Agents

Submitted by

Antor Sreejon Biswas
2022BC007

Department of Computer Science and Technology
Beijing Institute of Graphics Communication

Beijing, China

December 2025

CANDIDATE DECLARATION

This is to certify that the work presented in this thesis, titled, “Evaluating the Multilingual Capabilities of LLM-based Web UI Agents”, is the outcome of the investigation and research carried out by me.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Antor Sreejon
Biswas

2022BC007

ACKNOWLEDGEMENT

This thesis is the product of my independent research, which was self-conceived, executed, and funded.

I acknowledge the use of the following tools and resources: computational infrastructure; software tools and APIs (including M2M100, MADLAD-400, NLLB, Gemini, GPT, and DeepSeek); and publicly available datasets and foundational research, notably the Mind2Web dataset and associated literature, which made this work possible.

I confirm that no other individuals were involved in the core research activities of design, execution, analysis, or writing.

Beijing
December
2025

Sreejon Biswas
Antor

Contents

List of Figures

List of Tables

ABSTRACT

Automated UI agents capable of navigating and executing complex user-defined tasks on the web have demonstrated significant potential, primarily in English-language environments. However, research on developing such agents for non-English settings remains limited. In this study, I develop and evaluate a multilingual web agent capable of performing tasks in seven widely spoken languages - *Chinese*, *English*, *Spanish*, *Hindi*, *Portuguese*, *Bengali* and *Russian*. I fine-tune *XLM-RoBERTa* (a small language model) and *mT5* (a large language model) using a translated version of the *Mind2Web* dataset to assess their performance in understanding user intent. I also assess the zero-shot performance of a recent LLM (*Gemini 2.0 Flash*) on a subset of the translated dataset. Additionally, I evaluate the zero-shot performance of state-of-the-art LLMs (*Gemini 2.0 Flash*, *GPT-4o mini*, and *DeepSeek-v3*) on live multilingual websites. To enhance evaluation rigor, I incorporate an automated assessment method using a Vision-Language Model (VLM) alongside human evaluations. findings contribute to the advancement of multilingual web automation research, expose the limitation of html-only agents and highlight the need for more robust datasets to support diverse language-based web interactions.

Chapter 1

Introduction

The rapid advancement of technology has led to the development of automated systems capable of performing complex tasks with minimal human intervention. Among these systems, web-based user interface (UI) agents have gained significant attention due to their ability to navigate and execute tasks on the web. These agents are particularly useful in automating repetitive tasks, such as form filling, data extraction, and navigation, which are common in various online platforms. However, most of the research and development in this area has been focused on English-language environments, leaving a significant gap in the development of multilingual web agents.

The ability to interact with web interfaces in multiple languages is becoming increasingly important as the internet continues to grow globally [1]. With billions of users accessing the web in languages other than English, there is a pressing need for web agents that can understand and execute tasks in diverse linguistic contexts. This thesis aims to address this gap by evaluating the multilingual capabilities of large language model (LLM)-based web UI agents. Specifically, I focus on developing and assessing the performance of web agents in seven widely spoken languages: Chinese, English, Spanish, Hindi, Portuguese, Bengali, and Russian.

The primary objective of this study is to explore the challenges and opportunities in developing multilingual web agents. I begin by reviewing existing literature on web automation, cross-lingual language models, and the capabilities of state-of-the-art LLMs in handling web-based tasks. I then present a detailed methodology for training and evaluating multilingual web agents using a translated version of the *Mind2Web* [2] dataset. My approach involves fine- tuning small and large language models, such as *XLM-RoBERTa* [3] and *mT5* [4], to assess their performance in understanding user intent across different languages.

In addition to fine-tuning, I also evaluate the zero-shot capabilities of recent LLMs, such as *Gemini 2.0 Flash*, *GPT-4o mini* [5], and *DeepSeek-v3* [6], on live multilingual websites. This allows us to compare the performance of models trained on translated datasets with those that rely on their inherent multilingual understanding. To ensure rigorous evaluation, I incorporate both human and automated assessment methods [7], using vision-language models (VLMs) to provide additional insights into the agents' performance.

The findings of this research contribute to the growing body of knowledge on multilingual web automation. By highlighting the strengths and limitations of current approaches, I aim to provide a foundation for future research in this area. My work underscores the need for more robust datasets and evaluation frameworks to support the development of web agents that can effectively interact with users in diverse linguistic environments.

In summary, this thesis explores the potential of LLM-based web UI agents in multilingual settings, offering insights into their capabilities and challenges. I hope that my findings will inspire further research and innovation, ultimately leading to the creation of more accessible and effective web agents for non-English speakers.

1.1 Overview of My Methodology

1.1.1 Testing on translated Mind2Web Dataset

The Mind2Web dataset contains snapshots of over 2,000 open-ended tasks collected from 137 websites, along with their corresponding action sequences. To create a multilingual version, I translated English *task descriptions* into six of the most spoken languages - *Chinese*, *Spanish*, *Hindi*, *Portuguese*, *Bengali* and *Russian*.

Similar to the agent in the Mind2Web paper, I implemented a two-stage architecture

1. **Candidate Generation**: Top-k candidate DOM elements from the webpage are selected using a fine-tuned small ranking LM.

2. **Action Prediction**: The pruned HTML snippets and the candidates serve as inputs to an LLM, which treats element selection as a multi-choice question answering problem.

1.1.2 Testing on Live Websites

I also evaluated three frontier LLMs - *Gemini 2.0 Flash*, *GPT4o-mini*, and *DeepSeek v3* - on live websites. Rather than matching predefined trajectories as seen in the *Mind2Web* paper, I developed an agent that provides the LLMs with an HTML snapshot of the webpage and executes the actions predicted by the models and proceeds. To assess their performance, I curated a set of five representative tasks for each of the seven languages and tested them on websites corresponding to that language.

I conducted both human and automated evaluations. For automated evaluation, I used a VLM by providing it with a screenshot of each step along with the action performed by the agent. The VLM was then asked to determine whether the executed action was correct for that step.

1.2 List of Contributions

Our key contributions are as follows:

- Developed a multilingual web agent using LLMs capable of understanding and executing tasks in seven languages.
- Evaluated the performance of the agent on the translated Mind2Web dataset.
- Evaluated the zero-shot performance of frontier LLMs on the translated Mind2Web dataset and live websites.
- Conducted automated VLM-based evaluation to assess the performance of multilingual web agents.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows:

- **Chapter 2: Literature Review** provides an overview of existing research on web automation, cross-lingual language models, and the capabilities of LLMs in handling web- based tasks.
- **Chapter 3: Methodology** outlines the methodology used to train and evaluate multilingual web agents on the Mind2Web dataset and live websites.
- **Chapter 4: Experiments and Results** present the experimental setup, results, and analysis of the performance of the multilingual web agents.
- **Chapter 5: Limitations** discuss the key limitations of HTML-based web agents and the challenges in developing effective web automation systems.
- **Chapter 6: Future Works** explores potential directions for enhancing the performance of HTML-based web agents.
- **Chapter 7: Conclusion** summarizes the key findings of the thesis and outlines the implications for future research in multilingual web automation.

Chapter 2

Literature Review

In this chapter, I will review all the literatures that I have mentioned throughout the book.

2.1 MIND2WEB: Towards a Generalist Agent for the Web

MIND2WEB is a dataset for training generalist web agents to perform diverse tasks across real-world websites. It enables models to generalize across domains and tasks by capturing complex web interactions.

MIND2WEB consists of three components:

1. **Diverse Domains and Tasks**: With over 2,000 tasks from 137 websites spanning 31 domains, MIND2WEB covers search, form-filling, navigation, dynamic content selection, and pop-ups, ensuring adaptability across different web structures.
2. **Real-World Web Environments**: Unlike simulated datasets, MIND2WEB includes real interaction traces, HTML snapshots, and network logs, presenting challenges like dynamic layouts, asynchronous updates, and interactive elements.
3. **Multi-Step Interaction Sequences**: Tasks involve sequential actions like logging in, navigating menus, and filling forms. The dataset records actions such as clicking, typing, and selecting, enabling training for long-horizon dependencies.

MIND2WEB employs **MINDACT**, a two-stage modeling approach:

1. **Candidate Selection**: MINDACT starts with a ranking model that filters and prioritizes webpage elements based on task relevance. Since web pages contain thousands of elements, this filtering reduces computational complexity while maintaining accuracy. The ranking model considers factors like element position, visibility, and textual content.

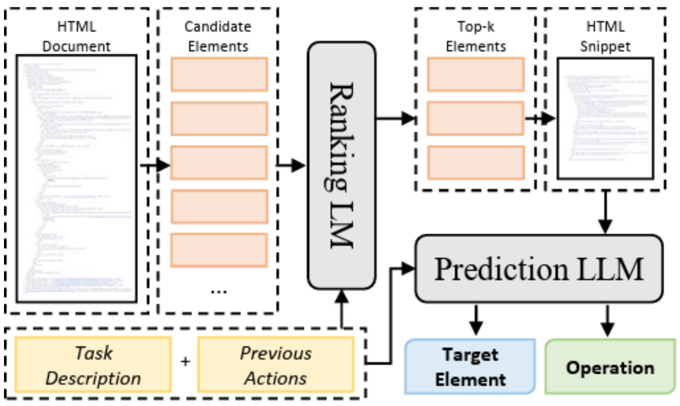


Figure 2.1: The overall pipeline for MINDACT with a small ranking LM for candidate generation, and a large prediction LM for action prediction.

2. **Action Prediction**: After filtering, an LLM processes the selected elements and determines the best action. Instead of parsing the entire webpage, the LLM uses a structured subset of information to predict interactions such as clicking buttons, filling in text fields, and selecting dropdowns. MINDACT employs a multi-choice decision framework to ensure logical action selection at each step.

This method enhances efficiency and accuracy, allowing web agents to handle complex workflows and adapt to new websites with minimal retraining.

2.2 Cross-lingual Language Model Pretraining

Cross-lingual Language Model Pretraining (XLM) [8] enhances multilingual natural language understanding by aligning multilingual embeddings within a shared representation space. It leverages both supervised and unsupervised techniques to improve cross-lingual transfer learning.

XLM consists of three key training objectives:

1. **Causal Language Modeling (CLM)**: A Transformer-based model is trained to predict the next word in a sequence using monolingual data. This enables contextualized embeddings that capture language-specific syntactic and semantic structures.

2. **Masked Language Modeling (MLM)**: Inspired by BERT, this objective randomly masks words in the input and requires the model to predict them using the surrounding context. This approach enables bidirectional language representations, improving generalization across languages.

3. **Translation Language Modeling (TLM)**: Extends MLM by leveraging parallel sentence pairs across different languages. Words are masked in both source and target sentences, encouraging the model to leverage cross-lingual context for predictions, thereby improving alignment between languages.

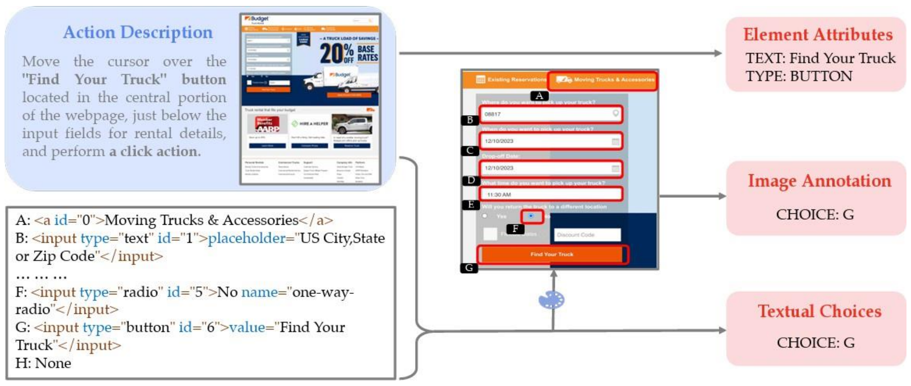
By integrating these objectives, XLM achieves state-of-the-art performance in zero-shot and few-shot cross-lingual classification tasks. It significantly enhances machine translation, information retrieval, and multilingual text understanding.

2.3 GPT-4V(ision) as a Generalist Web Agent

GPT-4V(ision) extends the capabilities of large multimodal models (LMMs) beyond traditional vision-language tasks by enabling interaction with web environments. By integrating vision and language understanding, it aims to function as a generalist web agent capable of executing tasks on real-world websites [9].

2.3.1 Key Components of GPT-4V as a Web Agent

1. **Visual Perception and Task Planning**: GPT-4V processes webpage screenshots to generate execution plans, overcoming the limitations of text-only LLMs, which struggle with incomplete HTML inputs.
2. **Action Grounding Strategies**: Several methods convert textual plans into actions:
   - **Element Attribute Grounding**: Matches predicted textual descriptions with HTML attributes.
   - **Textual Choice Grounding**: Provides candidate HTML elements as multi-choice selections for GPT-4V.
   - **Image Annotation Grounding**: Uses bounding boxes on rendered webpages to improve element identification.
3. **Evaluation and Performance**: On **MIND2WEB**, GPT-4V with oracle grounding achieves **51.1%** success on live websites, surpassing GPT-4 (13.3%) and FLAN-T5 (8.9%). However, grounding errors create a **20-30% gap** from oracle performance.
4. **Online vs. Offline Evaluation**: The study introduces an online evaluation framework where web agents interact with live websites rather than cached ones. This method provides a more accurate measure of real-world performance and highlights discrepancies in offline evaluation.



Move the cursor over the "Find Your Truck" button located in the central portion of the webpage, just below the input fields for rental details, and perform **a click action.**

```
A: <a id="0">Moving Trucks & Accessories</a>
B: <input type="text" id="1">placeholder="US City,State
or Zip Code"</input>
... ... ...
F: <input type="radio" id="5">No name="one-way-
radio"</input>
G: <input type="button" id="6">value="Find Your
Truck"</input>
H: None
```

Figure 2.2: Illustration of grounding strategies in GPT-4V-based web agents.

2.4 Autonomous Evaluation and Refinement of Digital Agents

Autonomous evaluation models play a crucial role in enhancing digital agents' performance in web navigation and device control [7]. By removing the need for human supervision, these models enable scalable refinement of agent behavior across various tasks and environments.

The proposed framework consists of two primary evaluation approaches:

1. **End-to-End Evaluation**: This approach directly maps agent interactions and visual states to an evaluation score using a vision-language model (VLM) such as GPT-4V. The model generates a step-by-step reasoning process and assigns a final success score, making it a powerful but computationally expensive method.

2. **Modular Caption-then-Reason Evaluation**: A two-step approach where a VLM first generates textual descriptions of visual observations, followed by a language model (LM) that reasons over these descriptions and assesses task success. This approach enhances explainability and enables the use of open-source models, reducing reliance on proprietary systems.

These evaluators have demonstrated high reliability, achieving agreement rates of **92.9%** with oracle metrics in Android device control settings and **82.1%** accuracy in web-based tasks such as WebArena. Furthermore, they enhance agent training through:

- **Inference-Time Refinement via Reflexion**: The evaluation models serve as reward functions for Reflexion, improving web agent success rates by **29%** on WebArena tasks.

- **Filtered Behavior Cloning for Device Control**: By filtering and selecting high-quality trajectories for training, these evaluators improve iOS device control success by **75%**.

By providing automated, generalizable feedback, these evaluation models enable robust real-world deployment of digital agents. They reduce dependence on manually curated benchmarks and facilitate adaptive learning in dynamic environments.

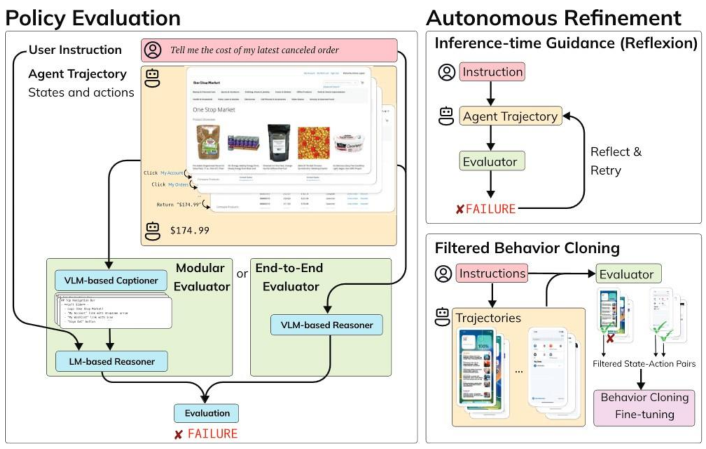


Figure 2.3: Framework of autonomous evaluation and integration with Reflexion.

2.5 WebCanvas: Benchmarking Web Agents in Online Environments

WebCanvas [10] is an online evaluation framework designed to assess the adaptability of web agents to the evolving nature of web environments. Unlike traditional static benchmarks, WebCanvas accounts for UI changes, content updates, and dynamic user interactions, making it a more realistic testing platform.

2.5.1 Key Features of WebCanvas

1. **Progress-Aware Evaluation**: Introduces a novel key node annotation system that tracks critical steps in task completion. This approach enables detailed analysis of agent performance while filtering out noise from irrelevant page changes.
2. **Mind2Web-Live Benchmark**: A dynamic dataset derived from the static Mind2Web dataset, containing 542 real-world tasks with 2,439 intermediate evaluation states. It provides up-to-date tasks reflecting the latest web changes, ensuring a realistic evaluation environment.
3. **Collaborative Annotation and Testing Tools**: WebCanvas features lightweight annotation tools and an online testing pipeline that allow researchers to maintain and expand the dataset continuously. This community-driven approach ensures long-term benchmark relevance.

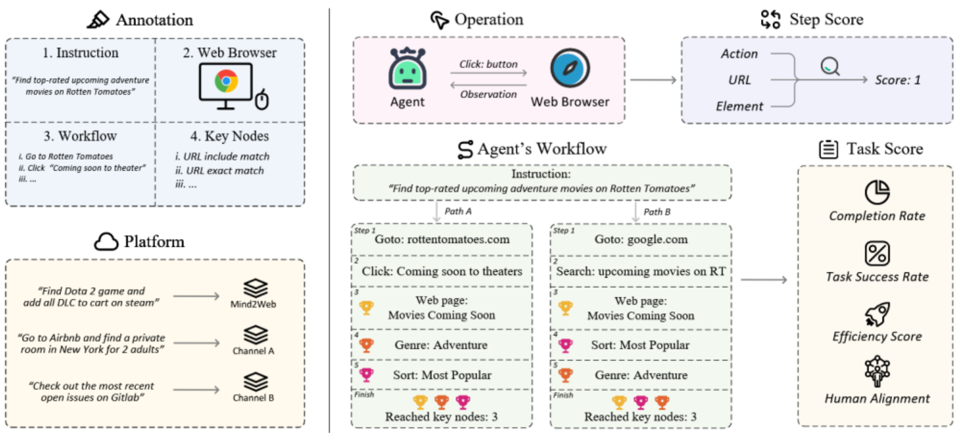


Figure 2.4: Overview of the WebCanvas framework and evaluation process.

2.5.2 Performance and Insights

WebCanvas reveals substantial discrepancies between offline and online evaluations. The best-performing agent achieves a **23.1% success rate** and a **48.8% task completion rate** on the Mind2Web-Live test set. Additionally, results highlight performance variations across different websites and domains, emphasizing the need for adaptive agents capable of handling real-time web changes.

Chapter 3

Methodology

In this chapter I describe dataset description, preparation and preprocessing steps, architecture of my models and experiment methods.

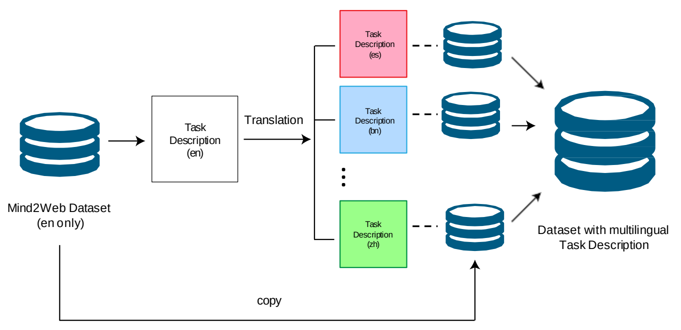
3.1 Dataset Description

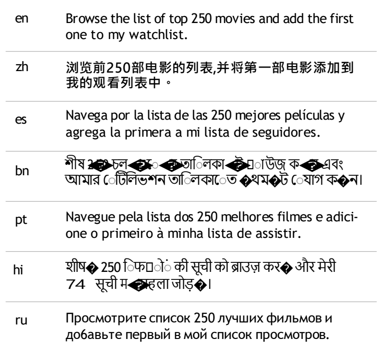
The **Mind2Web** dataset contains more than **2000** open-ended tasks collected from **137** websites spanning **31** domains and crowd-sourced action sequences for the tasks. Each instance in my dataset contains three key components:

- **Task description** – A high-level specification of the task's objective.
- **Action sequence** - A sequence of actions required to complete the task. Each action is represented as a *(Target Element, Operation)* pair. The *Target Element* is an interactable element on the current webpage, and the *Operation* refers to the action to be executed on that element. The dataset supports three common operations: CLICK, TYPE, and SELECT. For Type and Select Option, they also require an additional value as an argument.
- **Webpage snapshots** – A representation of the environment in which the task is performed. The dataset provides webpage information in multiple formats, including HTML, DOM tree, screenshots, and HAR files, supporting different modeling approaches. In my experiments, I utilized only the **HTML** format.

The test set of the dataset is splitted into three splits:

- **Cross Task** - Tasks from websites that were included in the training set.
- **Cross Website** – Tasks from unseen websites, but within the same domain as those in the training set.





en	Browse the list of top 250 movies and add the first one to my watchlist.
zh	浏览前250部电影的列表,并将第一部电影添加到我的观看列表中。
es	Navega por la lista de las 250 mejores películas y agrega la primera a mi lista de seguidores.
bn	শীষ 2�� চল�� � তালিকা �� াউজ ক�� এবং আমার েটিলিভশন তালিকােত �থম�ট েযাগ ক�ন।
pt	Navegue pela lista dos 250 melhores filmes e adicione o primeiro à minha lista de assistir.
hi	शीष� 250 िफ□ों की सूची को ब्राउज़ कर� और मेरी 74 सूची म�पहला जोड़�।
ru	Просмотрите список 250 лучших фильмов и добавьте первый в мой список просмотров.

Figure 3.1: Preparation of Dataset with multilingual task description from Mind2Web dataset

- **Cross Domain** – Tasks from entirely different domains than those present in the training set.

3.2 Dataset Translation

I select the seven most natively spoken language as my target languages - *Mandarin Chinese*, *Spanish*, *English*, *Hindi*, *Portuguese*, *Bengali* and *Russian*. To create multilingual variations of my dataset, I generate seven copies and translate the *Confirmed Task* field into each target language before merging them.

To enhance translation quality, my pipeline leverages three state-of-the-art (SoTA) open-source translation models: **M2M100-1.2B**, **MADLAD-400-3B**, and **NLLB-200-distilled-1.3B**. Among these, the best translation is selected based on LaBSE scores, ensuring high semantic similarity with the original text.

Finally, I introduce a form of quality control by filtering suboptimal translations through thresholding on the **LaBSE** scores and language identification with **FastText** (Thresholds: LaBSE_score ≥ 0.7 and FastText_score ≥ 0.85).

3.3 Architectural overview

I used the baseline *MindAct* framework introduced in the Mind2Web paper. It uses a two- stage process that synergizes the strength of small and large LMs, as shown in Figure 3.2. In the first stage, a fine-tuned small LM is used to rank the elements present on a webpage, yielding a small pool of promising candidates. In the second stage, these candidate elements are consolidated to form a representative snippet of the webpage, which is then processed by an LLM to predict the final action, including predicting both the element for interaction and the corresponding operation.

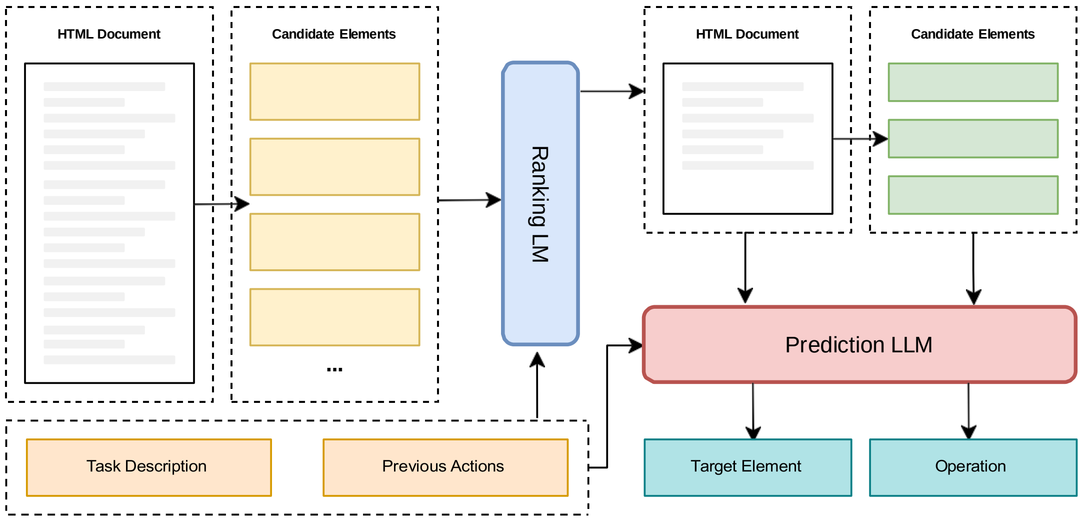


Figure 3.2: The overall pipeline for MINDACT with a small ranking LM for candidate generation, and a large prediction LM for action prediction.

3.3.1 Stage 1: Candidate Generation with small LMs

Given the task description, the snapshot of the webpage at step t, and the actions performed in the preceding $t-1$ steps, I treat candidate generation as a ranking task. The task is to select the top-k candidate DOM elements from the webpage that best align with both the task description and the current step. I formulate the task query by concatenating the task description with previous actions. The textual representation of each candidate DOM element is derived from a combination of the element's tag, its textual content, and salient attribute values, as well as the textual representation of its parent and child elements. I pair each DOM element with the task query and feed it to an encoder-only LM through the cross-encoder architecture, yielding a matching score. At training time, I randomly sample negative elements from the webpage, and use the target element as the positive example. The matching score is passed through a sigmoid activation function and optimized with a binary cross entropy loss. At inference time, I score all elements in the webpage and pick the top-k elements with the largest logits as input to the second stage.

3.3.2 Stage 2: Action Prediction with LLMs

After obtaining the top-k candidates, I prune the webpage snapshot and construct snippets that only include the selected candidates and their neighboring elements. This refined context is then fed into a large language model (LLM). Recent studies have suggested that training LMs for discrimination rather than generation is more generalizable and sample-efficient for other grounding tasks. Inspired by that, I frame the element selection task as a *multiple-choice question-answering (QA)* problem, where the model chooses from a predefined list of options rather than generating the target element directly.

I incorporate up to 5 candidate elements within each input, together with a *None* option, and partition the top-k candidates into several groups. During training, I construct the target sequence using ground-truth actions and fine-tune the model using a left-to-right language modeling objective. During inference, the top-k candidates are divided into clusters of five options each. If multiple options are selected in a given round, they are regrouped into new clusters for further evaluation. This process repeats until a single element is selected, or all options are rejected by the model, i.e., the model chooses the None option for all groups.

3.4 Zero-shot Action Generation with LLM

Recent advancements in large language models (LLMs) have significantly enhanced their ability to perform complex tasks without requiring fine-tuning. Instead of the two-stage architecture of the *MindAct* framework and choosing from a set of candidates in a MCQ style, I explore the raw capabilities of state-of-the-art (SoTA) LLMs in understanding multilingual user intent and interpreting HTML structures.

In this approach, I directly provide the LLM with the task description, previous actions, and an HTML snapshot of the current step. The model is then tasked with identifying the appropriate element from the HTML and generating the corresponding action. To evaluate its effectiveness, I sample 60 examples from each of the seven languages from the translated dataset and use ***Gemini 2.0 Flash*** as my LLM.

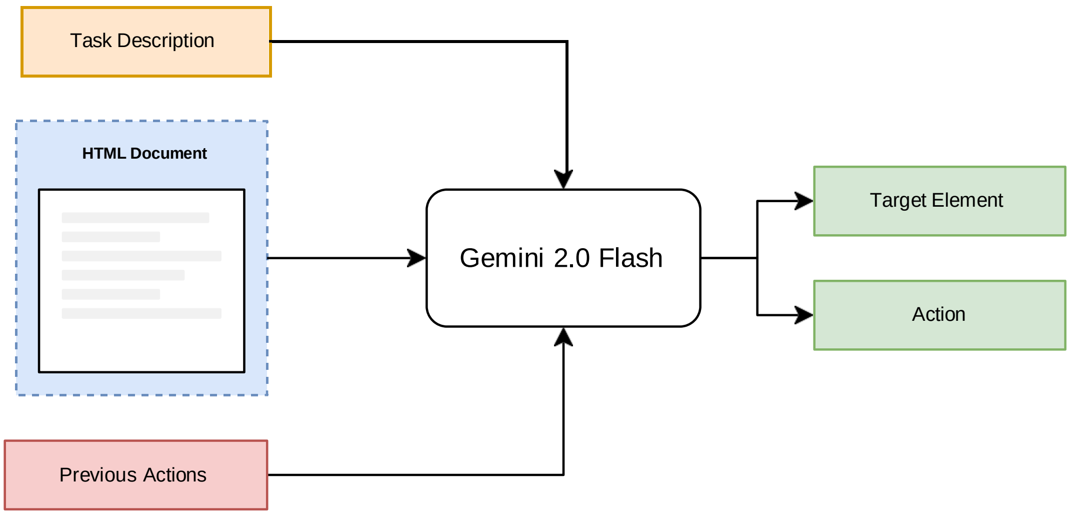


Figure 3.3: Zero-shot Action Prediction using Gemini 2.0 flash

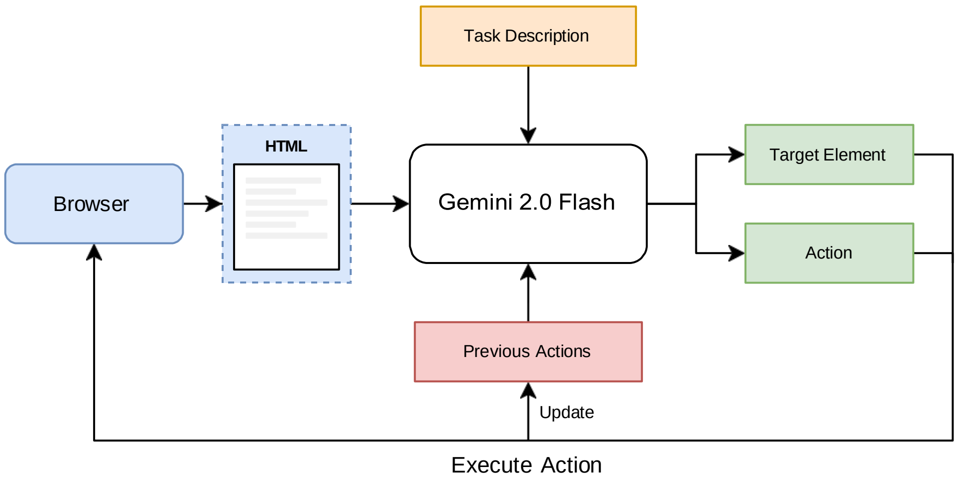


Figure 3.4: Agent Pipeline on live websites

3.5 Benchmarking LLMs on live websites

One of the primary limitations of fixed-trajectory datasets like *Mind2Web* is that they enforce a single predefined way to complete a task. However, in real-world scenarios, tasks can often be accomplished through multiple valid approaches. To better simulate real-world conditions, I conduct experiments on live websites. Figure 3.4 illustrates the pipeline of my experiment.

I used three frontier LLMs - **Gemini 2.0 Flash**, **GPT 4o mini** and **Deepseek-v3** as the backbone of my agent. The agent workflow was as follows:

1. The browser's HTML content is retrieved and fed into the LLM alongside the task description and the sequence of previously executed actions.
2. The LLM predicts the next action, which was then executed in the browser.
3. This process continued iteratively until either the agent determines that the task is completed and terminates the sequence, or, the execution has reached a predefined step limit.

To assess performance, I conduct both human evaluation and automated evaluation. If the agent makes an incorrect prediction at any step, I manually correct it, allowing the agent to proceed while marking that step as a failure. Additionally, I capture screenshots to facilitate automated evaluation.

3.6 Auto-Evaluation using VLM

Human evaluation does not scale well for large task sets. To address this, I implement an automated evaluation pipeline to assess how this evaluator compares to human evaluation in a multilingual setting for a smaller set of tasks.

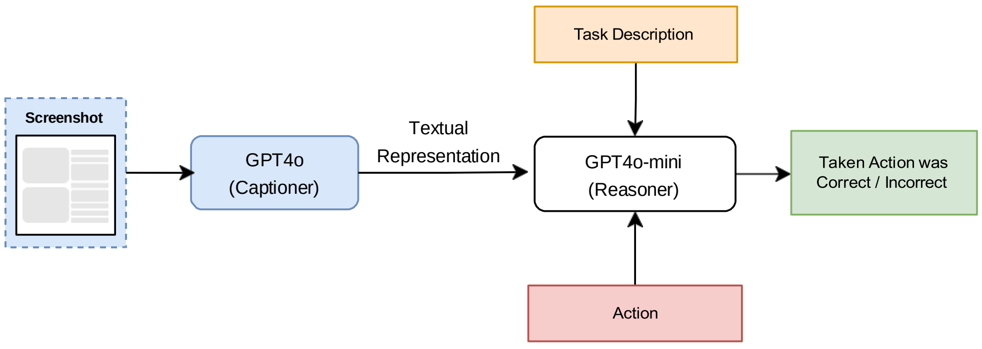


Figure 3.5: Automated evaluation pipeline

Figure 3.5 illustrates my auto-evaluation pipeline. First, I use **GPT-4o** as a captioner to generate a textual representation of the screenshot collected in the previous step. This textual representation, along with the task description and the action predicted by my agent, is then fed into **GPT-4o Mini**, which acts as a reasoner to determine whether the agent's action is correct.

Chapter 4

Experiments and Results

In this chapter I will show the results, and performance evaluation of the proposed model. I first discuss the performance metrics used to evaluate the model. I then present the performance of the fine-tuned models on the test set. Then I present the zero-shot performance of a frontier LLM, Gemini-2.0 Flash, on a fraction of the test set. Finally, I present the zero- shot performance of three frontier LLMs, Gemini-2.0 Flash, GPT-4o mini and DeepSeek-v3, on live multilingual websites. For evaluation of this live test, I compared the performance of the VLM based automated evaluation with human evaluation.

4.1 Performance Metrics

Here, I describe the performance metrics used to evaluate the web agents.

- *Element Accuracy*

  Definition

  Element Accuracy measures the proportion of examples for which the predicted UI element exactly matches the ground-truth element.

  Formula (per-sample and overall) –

  Per-sample element accuracy:

$$\text{element\_acc}_i = \begin{cases} 1 & \text{if } (\text{predicted\_element}_i = \text{true\_element}_i) \\ 0 & \text{otherwise} \end{cases}$$

- **Overall element accuracy** (micro): the mean across all samples,

$$\text{element\_acc} = \frac{1}{N}\sum_{i=1}^{N} \text{element\_acc}_i$$

where N is the total number of samples.

- *Operation F1*

Definition

Operation F1 is the average F1 score of the predicted action string compared to the ground-truth action string, computed token-by-token. Each action string is split into tokens, and the F1 is computed as:

1. Convert predicted action string into a set of tokens, pred_set.
2. Convert ground-truth action string into a set of tokens, label_set.
3. Compute True Positives (TP), False Positives (FP), and False Negatives (FN):

$$\text{TP} = |\text{pred\_set} \cap \text{label\_set}|,$$
$$\text{FP} = |\text{pred\_set} - \text{label\_set}|,$$
$$\text{FN} = |\text{label\_set} - \text{pred\_set}|.$$

4. Compute precision and recall:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

5. Compute F1 score:

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

The code averages this per-sample F1 over all samples to get the **overall operation F1**.

- *Step Success Rate*

Definition

Step Success Rate measures the fraction of steps for which *both* the element is predicted correctly **and** the operation F1 score is 1.0 (i.e., a perfect match for the tokens).

Formula (per-sample and overall) –

Per-sample step success:

$$\text{step\_success}_i = \begin{cases} 1 & \text{if } (\text{element\_acc}_i = 1) \text{ and } (\text{operation\_F1}_i = 1) \\ 0 & \text{otherwise} \end{cases}$$

– **Overall step success** (micro): the mean across all samples,

$$\text{step\_success\_rate} = \frac{1}{N} \sum_{i=1}^{N} \text{step\_success}_i.$$

- *Success Rate/Task Success Rate*

Definition

Success Rate measures the end-to-end success of a task, i.e., the fraction of tasks for which all steps are successful.

Formula

1. For each task, compute the task success:

$$\text{task\_success}(t) = \begin{cases} 1 & \text{if step\_success}_i = 1 \text{ for all steps in task } t \\ 0 & \text{otherwise} \end{cases}$$

2. Then average over all tasks:

$$\text{success\_rate} = \frac{1}{T} \sum_{t=1}^{T} \text{task\_success}(t),$$

where T is the total number of tasks.

- *Macro Element Accuracy*

Definition

Unlike the micro (overall) element accuracy—which is averaged over *all* samples—**Macro Element Accuracy** first computes an average element accuracy *per annotation_id* (or per "task"), then takes the mean of those per-annotation_id averages.

Formula

1. For each annotation ID a, compute the mean element accuracy:

$$\text{element\_acc}(a) = \frac{1}{|I_a|} \sum_{i \in I_a} \text{element\_acc}_i$$

where I_a is the set of samples belonging to annotation ID a.

2. Then average across all annotation IDs:

$$\text{macro\_element\_acc} = \frac{1}{A} \sum_{a=1}^{A} \text{element\_acc}(a),$$

with A being the total number of unique annotation IDs.

- *Macro Operation F1*

Definition

Similarly, **Macro Operation F1** first groups samples by their annotation ID, computes the mean operation F1 within each group, then takes the average over all groups.

Formula

1. For each annotation ID a, compute the mean F1:

$$\text{operation\_F1}(a) = \frac{1}{|I_a|} \sum_{i \in I_a} \text{operation\_F1}_i.$$

2. Then average over all annotation IDs:

$$\text{macro\_action\_F1} = \frac{1}{A} \sum_{a=1}^{A} \text{operation\_F1}(a).$$

- *Macro Step Success Rate*

Definition

Macro Step Success Rate (in the code, 'macro_step_acc') first computes the step success rate per annotation ID, and then averages these rates over all annotation IDs.

Formula

1. For each annotation ID a, compute the mean step success:

$$\text{step\_success}(a) = \frac{1}{|I_a|} \sum_{i \in I_a} \text{step\_success}_i.$$

2. Then average over all annotation IDs:

$$\text{macro\_step\_success} = \frac{1}{A} \sum_{a=1}^{A} \text{step\_success}(a).$$

4.2 Performance on the Mind2Web Dataset

Table 4.1 summarizes the results for the two-stage architecture across different setups. For the *Candidate Generation* step, the *DeBERTa-v3* [11] model was used in the first and fifth experiments, while the *XLM-RoBERTa* model was used for the remaining experiments. The first two experiments use the English-only task descriptions, while the rest use the multilingual task descriptions. Evaluation is conducted across three test splits: Cross-Task, Cross-Website, Cross-Domain.

I. I first reproduce the result of the original Mind2Web paper by fine-tuing and evaluating the MindAct architecture with DeBERTa-v3 for candidate generation and FLAN-T5 [12] for action prediction on the English split of the dataset.

II. I than use my combination of model on the English split - XLM-RoBERTa for candidate generation and mT5 for action prediction. I see a performance drop here from the previous experiment.

III. I then apply my combination of models to the multilingual-task version of the dataset. Here, the step success rate increases in the Cross-Task split but drops in the other two splits. While the task descriptions in the dataset are available in seven languages, the HTML body remains in English. This suggests that the model may be overfitting to the data, which could explain its inconsistent performance across different splits.

IV. *mT5* model is fine-tuned on multilingual task descriptions, but the *candidate generation* results are from *DeBERTa* model, which is fine-tuned on the English-only task dataset. These candidate generation results are then mapped to the multilingual dataset before the *action prediction* step.

Exp	Model	Cross Task		Cross Website		Cross Domain	
		Step SR	Task SR	Step SR	Task SR	Step SR	Task SR
I	FLAN-T5 (En)	41.1	3.97	28.5	1.13	30.3	1.64
II	mT5 (En)	34.7	2.38	29	0.56	29.5	1.54
III	mT5 (Multilingual)	35.5	2.00	24.1	0.65	24.8	1.60
IV	mT5 (DeBERTa)	36.2	1.43	29.5	0.97	26.8	1.43

Table 4.1: Step Success Rate (Step SR) and Task Success Rate (Task SR) for Different Models Evaluated on the Mind2Web dataset

Figure 4.1 shows the language-wise step success rate for the model in experiment III. I see that it scores the highest in English but there is no significant difference among the other lan- guages.

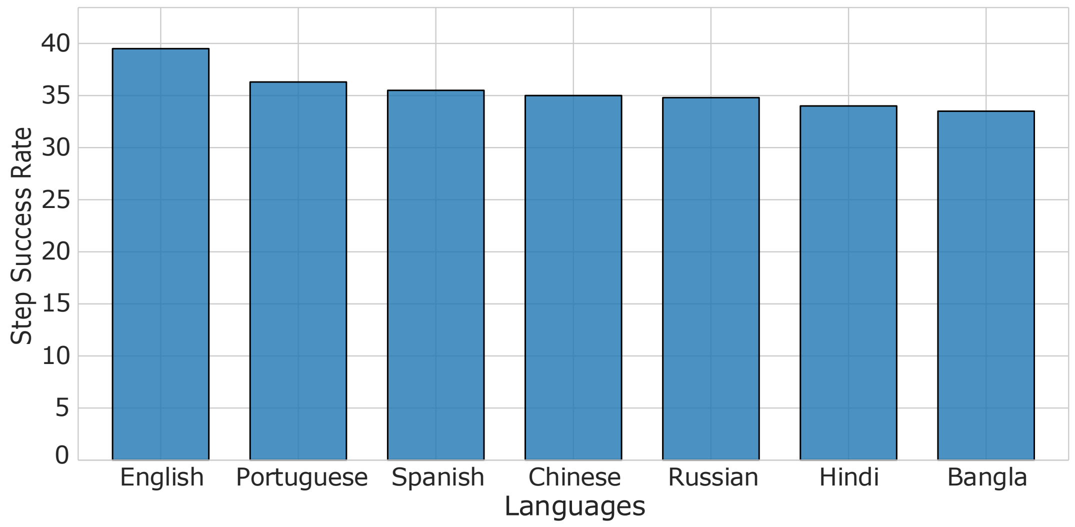


Figure 4.1: Language-wise (macro) step success rate on the Mind2Web dataset

Similar to the training technique in described in [8], I ran another experiment where in the training process, I sampled each batch from a single language. How ever this did not lead to any performance increase. On the contrary, the performance dropped in almost all cases.

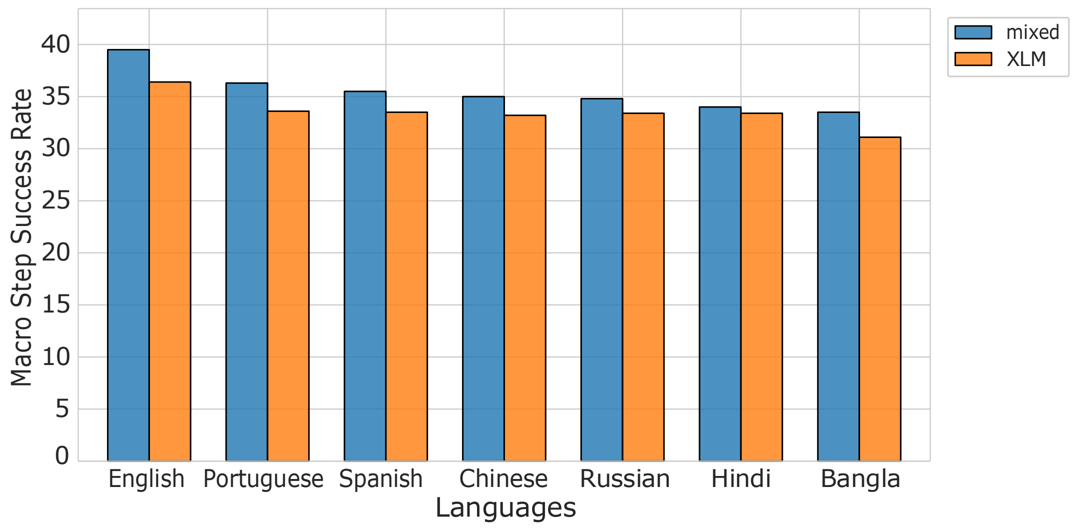


Figure 4.2: Language-wise comparison between XLM and non-XLM (macro) step success rate on the Mind2Web dataset

4.3 Zero-Shot Performance of LLMS on Mind2Web

In addition to the two-stage architecture, I experiment with a single-stage agent utilizing zero-shot prompting as shown in fig. 3.3. I sample 60 random data points from each language and evaluate the results generated by Gemini 2.0 Flash and compare them.

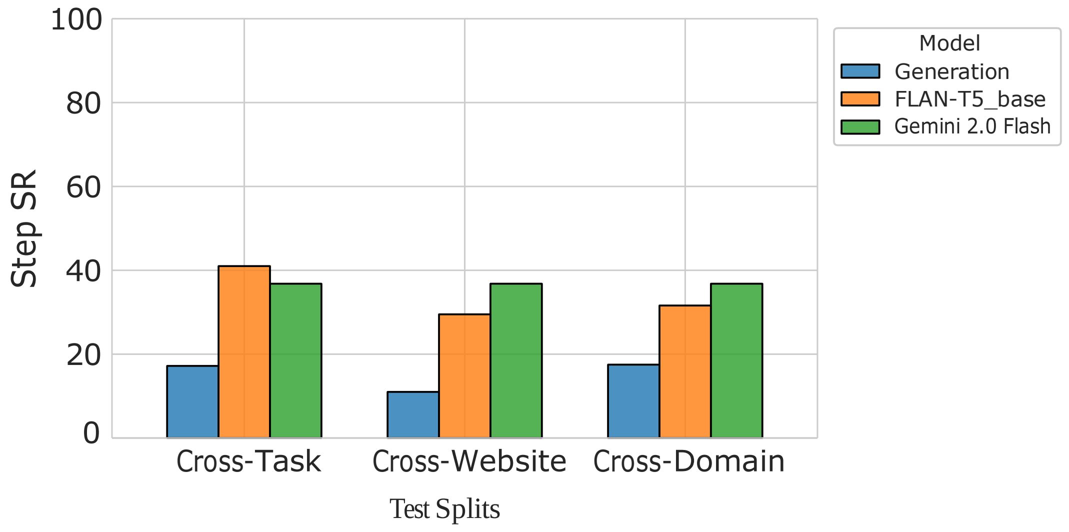


Figure 4.3: Step Success comparison on English Data of Mind2web

Figure 4.3 compares the step success rate of my Gemini-based agent with the values of the mind2web paper. *FLAN-T5_base* represents the *MindAct* agent. *Generation* represents the two-stage architecture but without the MCQ format prompt in the second stage. This is similar to my pipeline with Gemini. Instead of asking to choose from a pool of candidate elements, we are directly asking the agent to predict the target element.

Here I see a big jump in step success rate from the Generation method. My agent also performs very close to the MindAct agent on the Cross-Task split and exceeds it in the Cross- Website and Cross-Domain split.

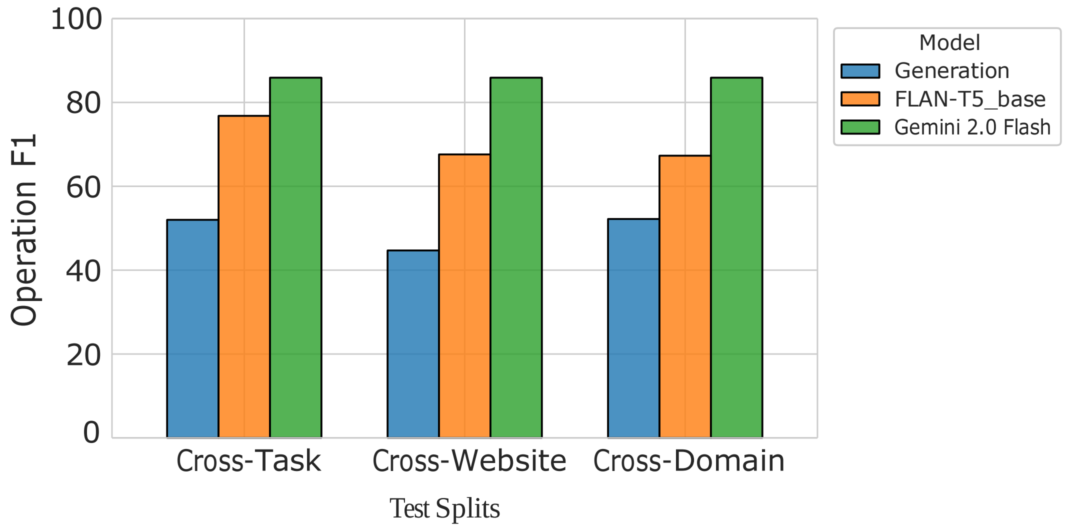


Figure 4.4: Operation F1 comparison on English Data of Mind2web

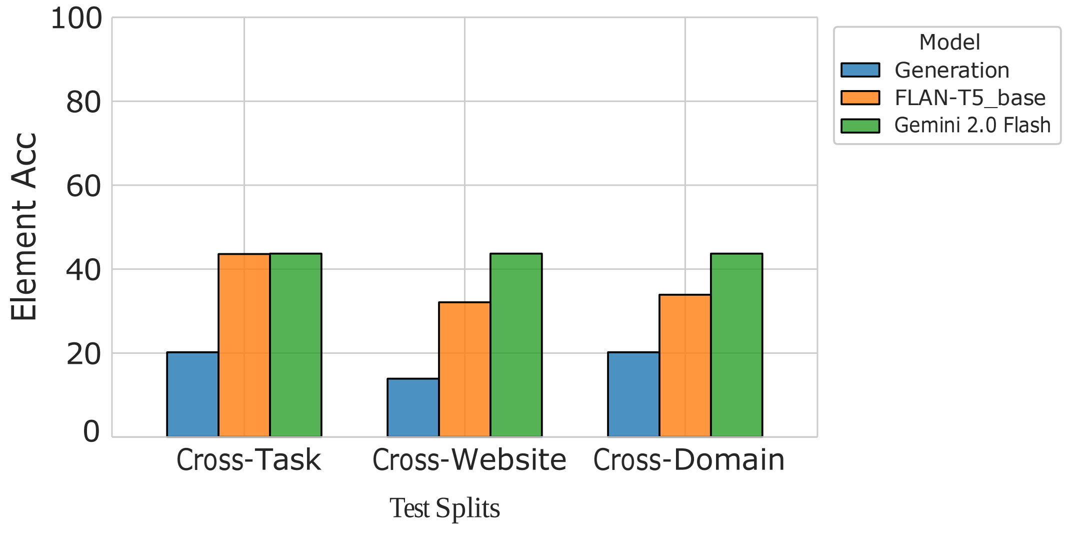
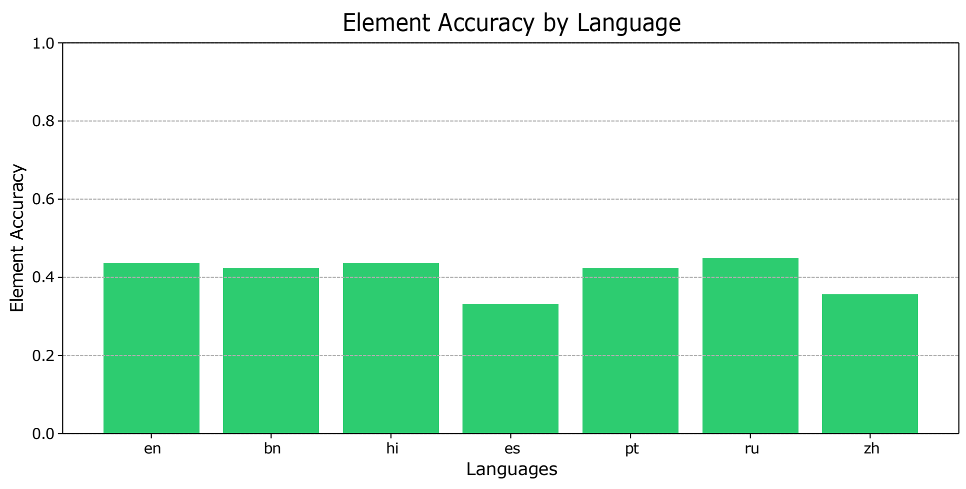
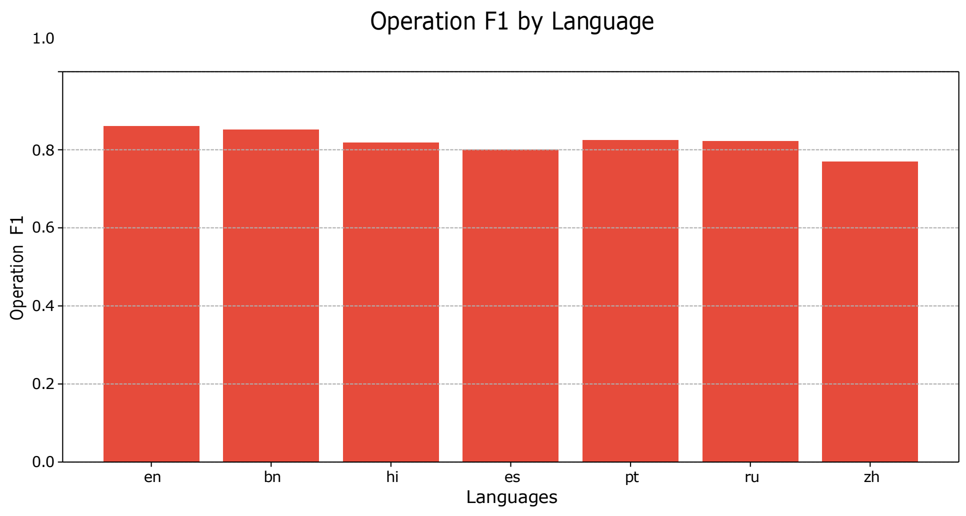


Figure 4.5: Element accuracy comparison on English Data of Mind2web

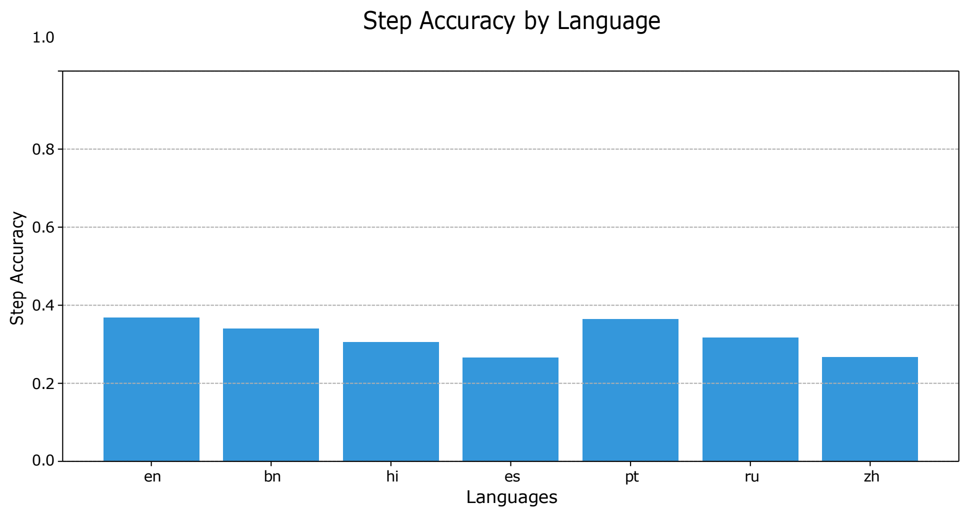
Figure 4.4 and Figure 4.5 shows the comparison between the agents in Operation F1 and Element Acc. Gemini 2.0 achieves the highest Operation F1. But it and the other two has low element accuracy. It suggests that the agents are good at figuring out what it needs to do broadly but fails to predict the correct id of the target element.



(a) Language-wise comparison of element accuracy



(b) Language-wise comparison of operation f1



(c) Language-wise comparison of step success rate

Figure 4.6: Language-wise results for Gemini 2.0 Flash on Mind2Web

Figure 4.6 shows the language-wise performance of the Gemini 2.0 Flash agent on a subset of the Mind2Web dataset. I do not see any significant difference in performance here, the agents performs similarly for almost all languages.

4.4 Zero-Shot Performance of LLMS on live websites

In this section, I present the zero-shot performance of three frontier LLMs, Gemini-2.0 Flash, GPT-4o mini and DeepSeek-v3, on live multilingual websites. To assess their performance, I curated a set of five representative tasks for each of the seven languages and tested them on websites corresponding to that language. Table 4.2 shows the task descriptions corresponding to different websites.

Website	**Task in English**
Amazon	Find a Bluetooth speaker with highest customer rating and add it to the cart
Amazon	Clear everything from cart
Google News	Find and open the latest news on NVIDIA RTX 5070
YouTube	Find the latest video from the Dude Perfect channel in YouTube and like it
Google Flights	Find the cheapest one-way flight from New York to London on 7th April

Table 4.2: Task descriptions corresponding to different websites

Table 4.3 shows the zero-shot performance of the three frontier LLMs on live websites. I conducted both human and automated evaluations. Automated Evaluation was done with GPT-4o. Here, Gemini 2.0 Flash outperforms the other two models in all three metrics.

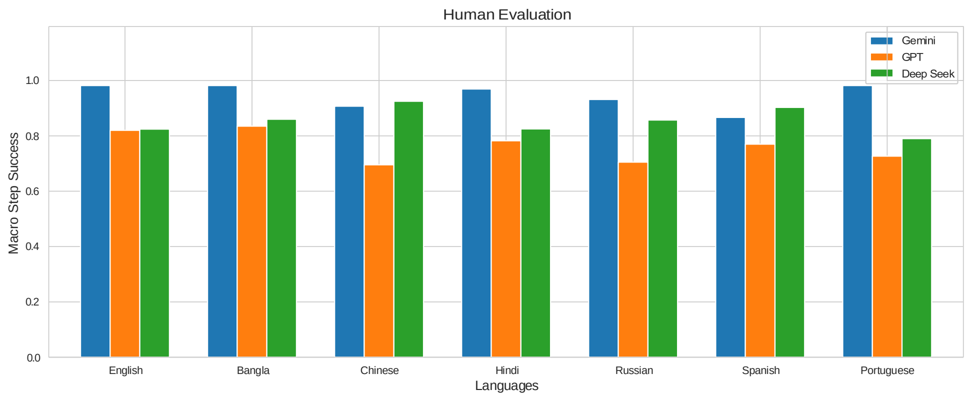
Table 4.3: Comparison of success rates (in %) under Human and Automatic evaluations.

Metric	**Gemini-2.0 Flash**		**GPT-4o-mini**		**Deepseek-v3**	
	Human	**Auto**	**Human**	**Auto**	**Human**	**Auto**
Step Success Rate	92.2	91.6	73.8	80.6	83.1	88.4
Macro Step Success Rate	94.6	93.3	76.2	81.2	85.5	87.7
End-to-End Success Rate	68.6	68.6	28.6	48.6	57.1	60.0

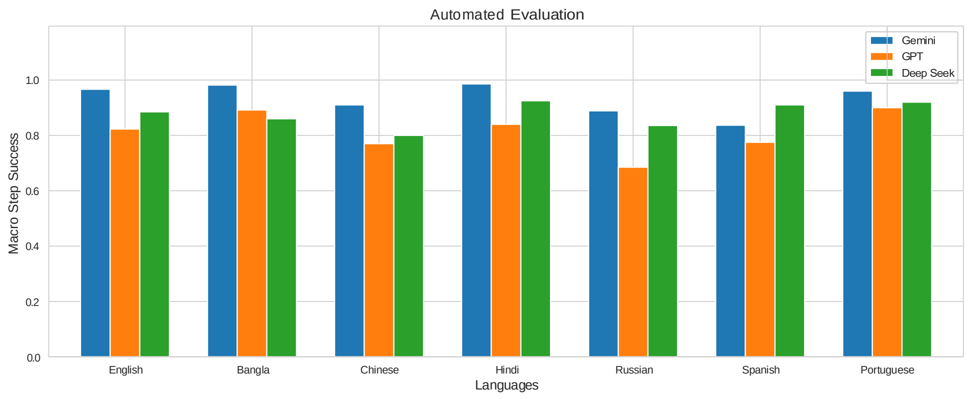
Now, I will compare language-wise performance of the three models. Table 4.4 shows the macro step success rate of the three models on live websites. Gemini-2.0-Flash shows the best performance in English, Portuguese and Bangla. Deepseek-v3 shows the best performance in Chinese. GPT-4o mini shows the best performance in Bangla.

Table 4.4: Macro Step Success Rate of LLMs on Live Websites

Language	Gemini-2.0-Flash		GPT4o-mini		Deepseek-v3	
	Auto Eval	Human Eval	Auto Eval	Human Eval	Auto Eval	Human Eval
English	96.7	98.2	82.3	82.0	88.5	82.4
Portuguese	96.0	98.2	90.0	72.7	91.9	79.0
Spanish	83.7	86.7	77.5	77.0	91.0	90.3
Chinese	91.0	90.7	77.0	69.5	80.0	92.5
Russian	88.9	93.1	68.5	70.5	83.6	85.7
Hindi	98.6	96.9	84.0	78.2	92.5	82.5
Bangla	98.2	98.2	89.2	83.5	86.0	86.0
Overall	93.3	94.6	81.2	76.2	87.7	85.5



(a) Macro Step Success Rate of LLMs under Human Evaluation

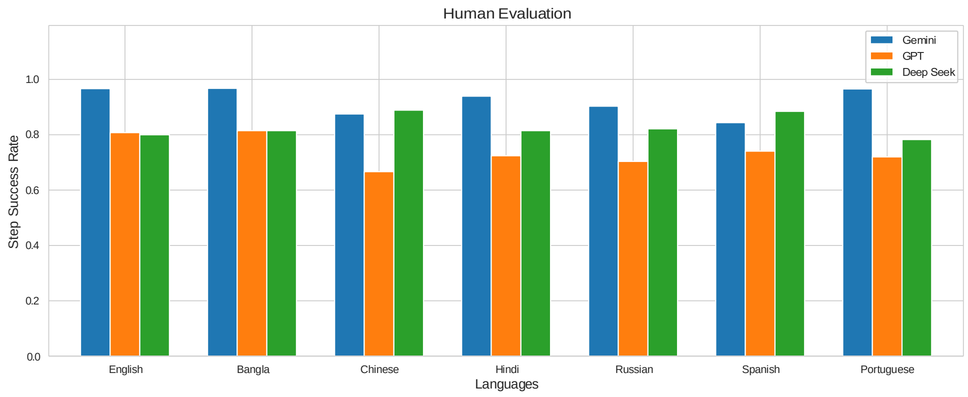


(b) Macro Step Success Rate of LLMs under Automatic Evaluation

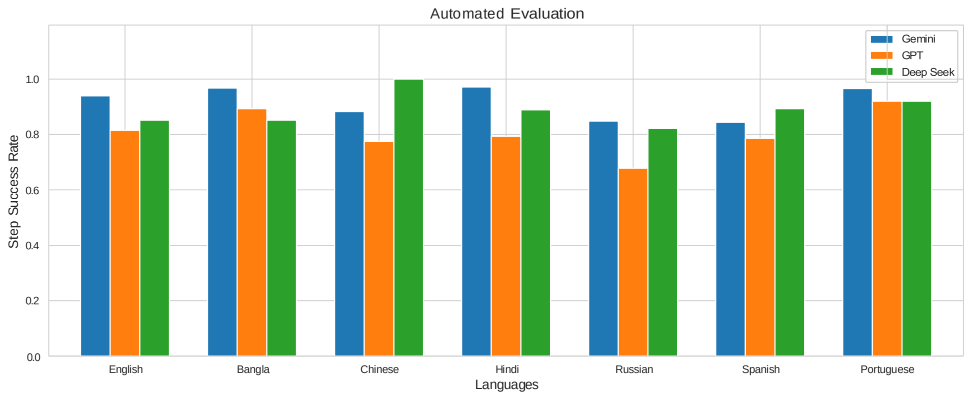
Figure 4.7: Comparison of Macro Step Success Rates under Human and Automatic Evaluations

Table 4.5: Step Success Rate of LLMs on Different Languages (in %)

Language	**Gemini**		**GPT**		**DeepSeek**	
	Auto Eval	**Human Eval**	**Auto Eval**	**Human Eval**	**Auto Eval**	**Human Eval**
English	93.9	96.7	81.5	80.8	85.2	80.0
Portuguese	96.6	96.6	92.0	72.0	92.0	78.3
Spanish	84.4	84.4	78.6	74.1	89.3	88.5
Chinese	88.2	87.5	77.4	66.7	100.0	88.4
Russian	84.9	90.3	67.9	70.4	82.1	82.1
Hindi	97.1	93.9	79.3	72.4	88.9	81.5
Bangla	96.8	96.8	89.3	81.5	85.2	81.5



(a) Step Success Rate of LLMs under Human Evaluation

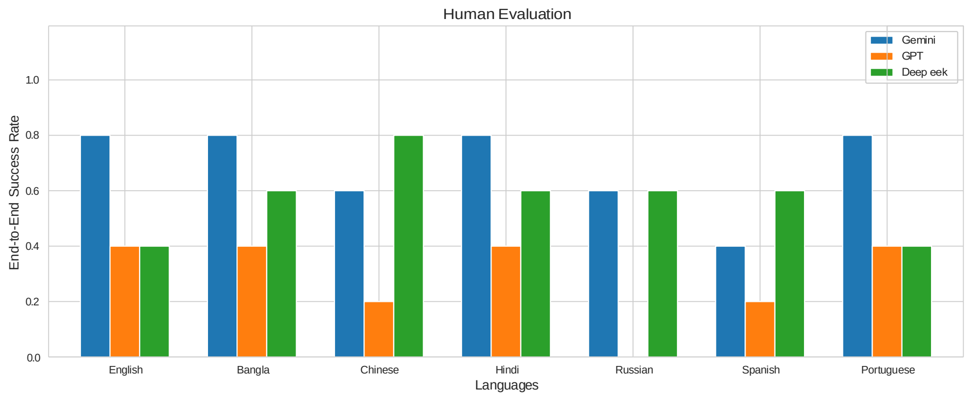


(b) Step Success Rate of LLMs under Automatic Evaluation

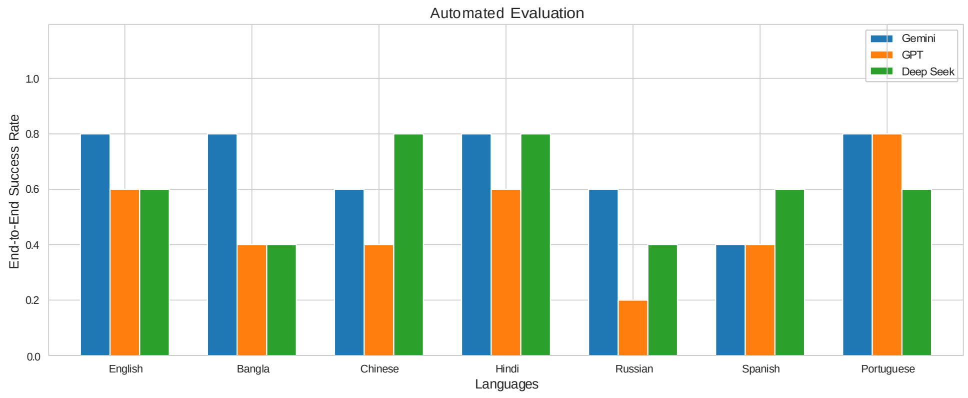
Figure 4.8: Comparison of Step Success Rates under Human and Automatic Evaluations

Language	**Gemini**		**GPT**		**Deep Seek**	
	Auto Eval	**Human Eval**	**Auto Eval**	**Human Eval**	**Auto Eval**	**Human Eval**
English	80	80	60	40	60	40
Portuguese	80	80	80	40	60	40
Spanish	40	40	40	20	60	60
Chinese	60	60	40	20	80	80
Russian	60	60	20	0	40	60
Hindi	80	80	60	40	80	60
Bangla	80	80	40	40	40	60

Table 4.6: End-to-End Success Rate of LLMs on Different Languages (in %)



(a) End-to-End Success Rate of LLMs under Human Evaluation



(b) End-to-End Success Rate of LLMs under Automatic Evaluation

Figure 4.9: Comparison of End-to-End Success Rates under Human and Automatic Evaluations

Chapter 5

Limitations

In this chapter, I discuss the key limitations of HTML-based web agents. These limitations highlight the challenges in developing effective and reliable web automation systems.

- **Large HTML Size:** Webpages often contain thousands of HTML tags, making the HTML structure very large. However, large language models (LLMs) have a limited context window, which restricts how much HTML content they can process at once. For example:

  – Gemini-2.0 Flash: 1 million tokens

  – GPT-4o-mini: 128,000 tokens

  – DeepSeek-v3 (API): 64,000 tokens

  When the HTML exceeds the context window size, I prune it to fit, which can lead to loss of important information and affect the agent's performance.

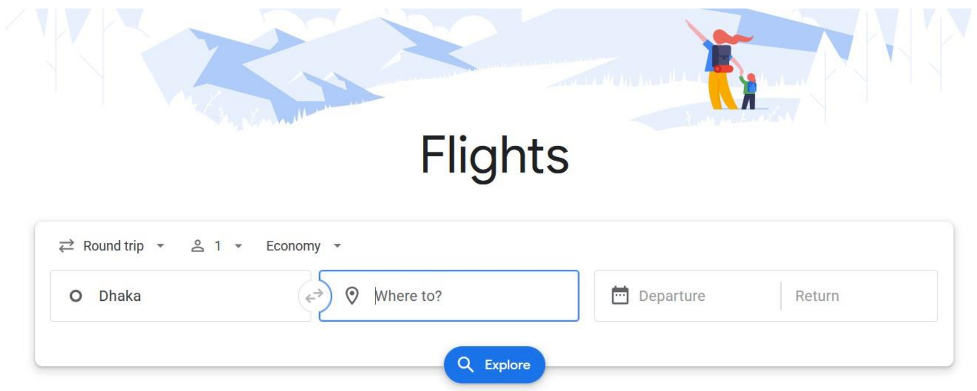
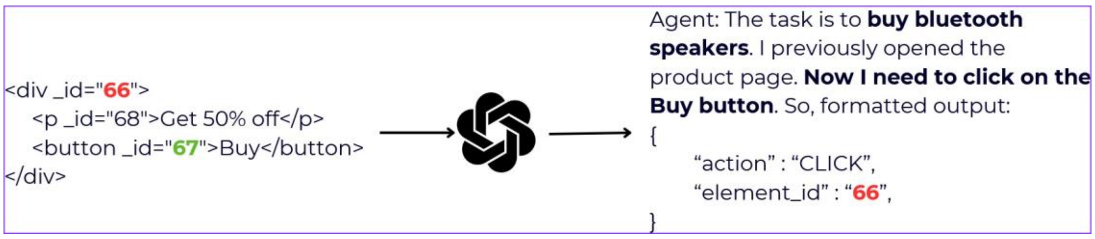


Figure 5.1: Fixed Trajectory for Tasks in Dataset

- **Fixed Task Trajectories:** The dataset used for training and offline experiments follows a fixed sequence of actions for each task. However, in real-world scenarios, tasks can often



```
<div _id="66">
  <p _id="68">Get 50% off</p>
  <button _id="67">Buy</button>
</div>
```

Agent: The task is to **buy bluetooth speakers**. I previously opened the product page. **Now I need to click on the Buy button**. So, formatted output:

```
{
    "action" : "CLICK",
    "element_id" : "66",
}
```

Figure 5.2: Action Grounding Failure

be completed in multiple ways. For example, as shown in Figure 5.1, when booking a flight ticket, users can fill out input fields in any order. The fixed trajectory approach forces the agent to follow a specific sequence, and deviations are considered failures. This limits the agent's flexibility and may underestimate its true capabilities.

- **Action Grounding Issues:** Action grounding refers to the process of executing a predicted action on the webpage. A major challenge is that the agent may correctly predict the action but fail to select the right HTML element to perform it. For instance, in Figure 5.2, the agent correctly identifies the need to click the "Buy" button but selects the wrong element, leading to an execution failure. This is a significant limitation for HTML-based agents.

- **Limited Action Types:** In my experiments, the agent was restricted to four types of actions:

  - Type
  - Click
  - Select
  - Press Enter

  However, real-world web interactions often require additional actions like scrolling, hovering, dragging, and dropping. This limitation can hinder the agent's ability to handle complex tasks effectively.

- **Differences Between Human and Agent Behavior:** The agent may not perform actions in the same way as a human. For example, as shown in Figure 5.3, when searching for a product, a human would typically:

  - Click on the search bar
  - Type the product name
  - Press Enter

  However, the agent might directly type the product name without clicking the search bar, as it has direct access to the HTML. While I tried to mimic human behavior in my experiments, this may not always be the most efficient approach for the agent.

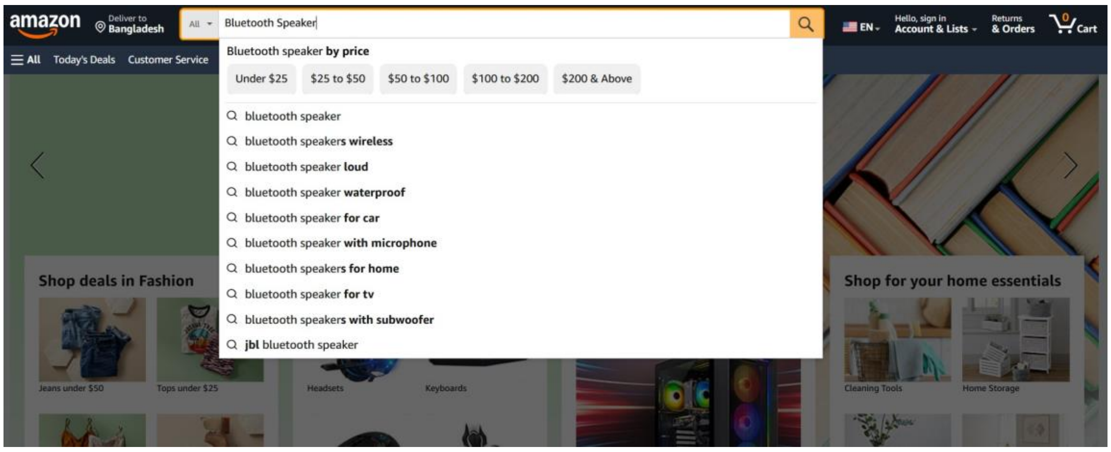


Figure 5.3: Searching for a Product

- **Browser and JavaScript Restrictions:** Certain browser and JavaScript behaviors can cause issues for the agent. For example, in Figure 5.4, clicking on the departure date field triggers a calendar pop-up. Although the calendar is present in the HTML before the field is clicked, the agent might attempt to interact with it prematurely, leading to an execution failure.

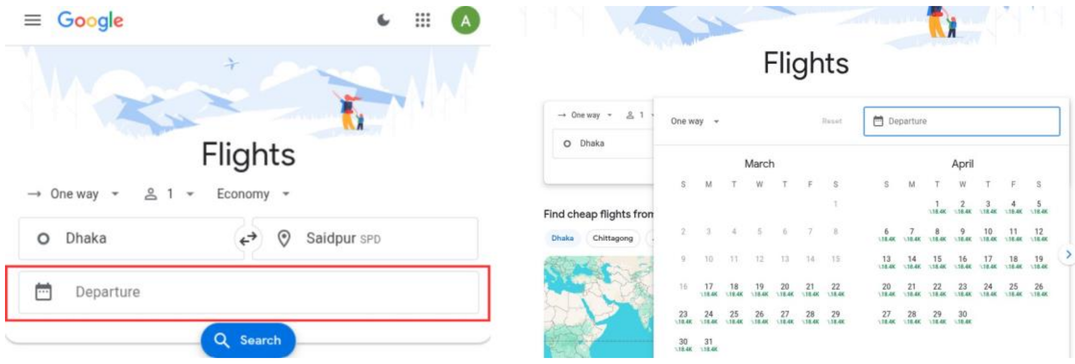


Figure 5.4: Browser/JavaScript Restrictions

- **HTML Cleaning:** To reduce the size of the HTML, I removed unnecessary CSS attributes like color and font size, as they are not critical for interactions. However, this can sometimes backfire. For instance, in Figure 5.5, when booking a movie ticket, the agent needs to distinguish between available and booked seats based on color. Removing the color attribute can cause the agent to select a booked seat, resulting in a failure.

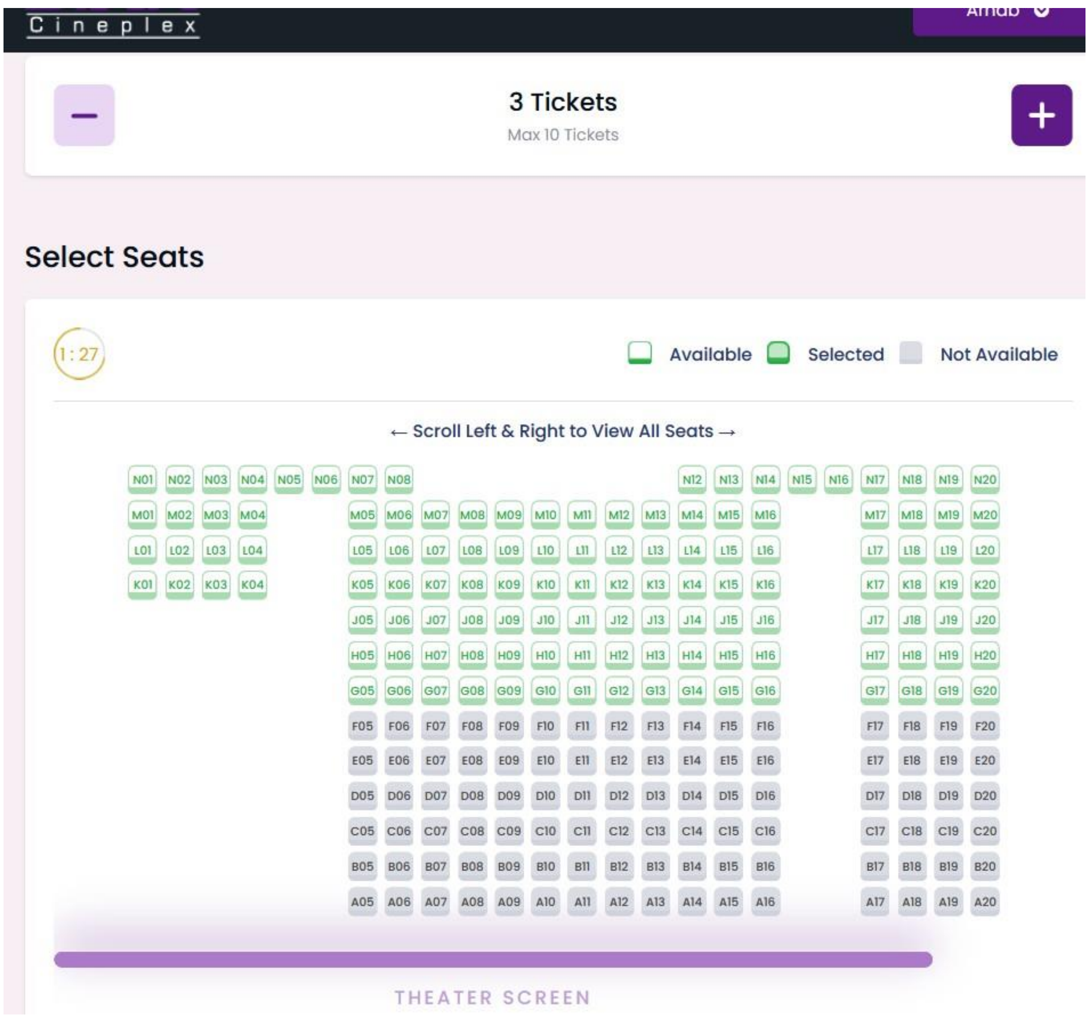

Cineplex
Amab
–
3 Tickets
Max 10 Tickets
+
Select Seats
1 : 27
Available
Selected
Not Available
← Scroll Left & Right to View All Seats →
N01 N02 N03 N04 N05 N06 N07 N08 N12 N13 N14 N15 N16 N17 N18 N19 N20
M01 M02 M03 M04 M05 M06 M07 M08 M09 M10 M11 M12 M13 M14 M15 M16 M17 M18 M19 M20
L01 L02 L03 L04 L05 L06 L07 L08 L09 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19 L20
K01 K02 K03 K04 K05 K06 K07 K08 K09 K10 K11 K12 K13 K14 K15 K16 K17 K18 K19 K20
J05 J06 J07 J08 J09 J10 J11 J12 J13 J14 J15 J16 J17 J18 J19 J20
H05 H06 H07 H08 H09 H10 H11 H12 H13 H14 H15 H16 H17 H18 H19 H20
G05 G06 G07 G08 G09 G10 G11 G12 G13 G14 G15 G16 G17 G18 G19 G20
F05 F06 F07 F08 F09 F10 F11 F12 F13 F14 F15 F16 F17 F18 F19 F20
E05 E06 E07 E08 E09 E10 E11 E12 E13 E14 E15 E16 E17 E18 E19 E20
D05 D06 D07 D08 D09 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20
C05 C06 C07 C08 C09 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20
B05 B06 B07 B08 B09 B10 B11 B12 B13 B14 B15 B16 B17 B18 B19 B20
A05 A06 A07 A08 A09 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20
THEATER SCREEN

Figure 5.5: HTML Cleaning

These limitations highlight the need for further research and improvements to make HTML based web agents more robust and effective in handling real-world web interactions.

Chapter 6

Future Work

In this chapter, I explore potential directions for enhancing the performance of HTML-based web agents. Below are some key areas for future improvement:

- **Multi-modal Agents:** Many challenges faced by HTML-based web agents could be addressed by developing multi-modal agents that utilize both HTML and visual data. Similar approaches have already shown promise in English-only settings, such as SeeAct [9], WebVoyager [13], and Android in the Wild [14]. Extending these methods to multilingual environments could significantly improve performance.
- **Supporting Multiple Trajectories:** The issue of fixed trajectories can be resolved by enabling multiple pathways for completing a task. Research in English-only environments, like WebCanvas, has demonstrated the benefits of this approach. Adapting these methods to multilingual contexts could enhance flexibility and effectiveness.
- **Enhancing Action Grounding:** The challenge of accurately grounding actions in real-world websites could be tackled using Vision-and-Language Models (VLMs). For example, GPT-4V-ACT has shown potential in this area. Expanding such models to multilingual environments could improve action precision and reliability.
- **Building Better Auto-evaluators:** The current limitations of automated evaluators can be addressed by fine-tuning models to create more accurate and multilingual auto-evaluators. This would help in better assessing the performance of web agents across diverse languages and tasks.

By focusing on these areas, future research can overcome existing limitations and pave the way for more robust and versatile multilingual web agents.

Chapter 7

Conclusion

This thesis set out to develop a multilingual web agent capable of understanding and executing tasks across seven widely spoken languages. Through an extensive review of existing literature, it became evident that research on developing such agents for non-English languages is scarce. The development of these agents is particularly challenging due to the inherent linguistic diversity and the lack of comprehensive datasets that support web interactions in multiple languages. Additionally, creating an HTML-based web agent that performs effectively on real-world websites presents significant challenges.

My approach involved working with a translated version of the *Mind2Web* dataset and adopting the two-stage architecture of *MindAct* to develop the agent. The training process proved effective for non-English languages, yielding results comparable to those achieved in English. However, the agent's performance on real-world websites was suboptimal. This shortfall can be attributed to the dataset's support for fixed task trajectories, which limited the agent's ability to generalize to new websites.

To mitigate this issue, I assessed the zero-shot performance of cutting-edge large language models (LLMs) on live multilingual websites. The results demonstrated a significant improvement over the predefined trajectories from the dataset. This suggests that the zero-shot capabilities of advanced LLMs are more adept at handling real-world websites than models fine-tuned on translated datasets.

Despite these advancements, the primary bottleneck in nearly all experiments was the limitations inherent in the HTML-based approach. Additionally, effectively grounding the actions predicted by the agents in real-world websites posed a significant challenge. Addressing these limitations is crucial for future research aimed at developing more robust and effective multi- lingual web agents.

In summary, my work contributes to the progress of multilingual web automation research and underscores the necessity for more robust datasets to facilitate diverse language-based web interactions. I hope that my findings will inspire further research in this domain, ultimately leading to the creation of more effective and accessible web agents for non-English speakers.

References

[1] B. Danet and S. C. Herring, “Introduction: the multilingual internet,” *Journal of Computer-Mediated Communication*, vol. 9, p. JCMC9110, 07 2017.

[2] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su, “Mind2web: Towards a generalist agent for the web,” 2023.

[3] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, “Unsupervised cross-lingual representation learning at scale,” 2020.

[4] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, “mt5: A massively multilingual pre-trained text-to-text transformer,” 2021.

[5] OpenAI, “Gpt-4o system card,” 2024.

[6] DeepSeek-AI, “Deepseek-v3 technical report,” 2025.

[7] J. Pan, Y. Zhang, N. Tomlin, Y. Zhou, S. Levine, and A. Suhr, “Autonomous evaluation and refinement of digital agents,” 2024.

[8] G. Lample and A. Conneau, “Cross-lingual language model pretraining,” 2019.

[9] B. Zheng, B. Gou, J. Kil, H. Sun, and Y. Su, “Gpt-4v(ision) is a generalist web agent, if grounded,” *arXiv preprint arXiv:2401.01614*, 2024.

[10] Y. Pan, D. Kong, S. Zhou, C. Cui, Y. Leng, B. Jiang, H. Liu, Y. Shang, S. Zhou, T. Wu, and Z. Wu, “Webcanvas: Benchmarking web agents in online environments,” 2024.

[11] P. He, J. Gao, and W. Chen, “Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing,” 2023.

[12] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, “Scaling instruction-finetuned language models,” 2022.

[13] H. He, W. Yao, K. Ma, W. Yu, Y. Dai, H. Zhang, Z. Lan, and D. Yu, “Webvoyager: Building an end-to-end web agent with large multimodal models,” 2024.

[14] C. Rawles, A. Li, D. Rodriguez, O. Riva, and T. Lillicrap, “Android in the wild: A large-scale dataset for android device control,” 2023.

Generated using Undegraduate Thesis LATEX Template, Version 2.2. Department of Computer Science and Technology, Beijing Institute of Graphic Communication, Beijing, China.

This thesis was generated on Thursday 7th December, 2025.