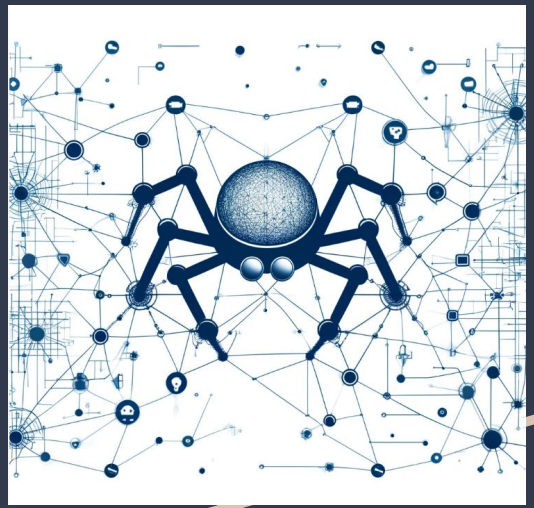
A Platform for Testing Conversational Web Navigation

Mohamed Mohamed and Omar Lahlou

Introduction



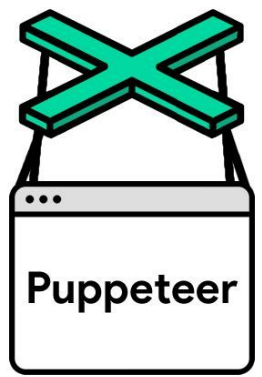
Task:

- Platform for testing WebLINX models via Chrome extension

Background:

- WebLINX is a dataset for conversational web navigation
  - Enables development of helpful, real world conversational web navigation agents
  - Important for accessibility
- Previous work:
  - Non-conversational (SeeAct, VisualWebArena)
  - Simulated environments (VisualWebArena)
  - Not publicly available (ACT-1)
- WebLINX is currently just a benchmark dataset, work needed to apply it to be easily used across the internet

Automation tools



What are they ? 🧐

- Automate web browsing tasks
  - Ex- opening webpage, clicking links
- Enable testing and validation on browsers

Common tools:

- Playwright
- Puppeteer
- Selenium

Things to consider when choosing automation tools:

- Testing speed
- Browser support
- Documentation/accessibility

Implementation

System structure:

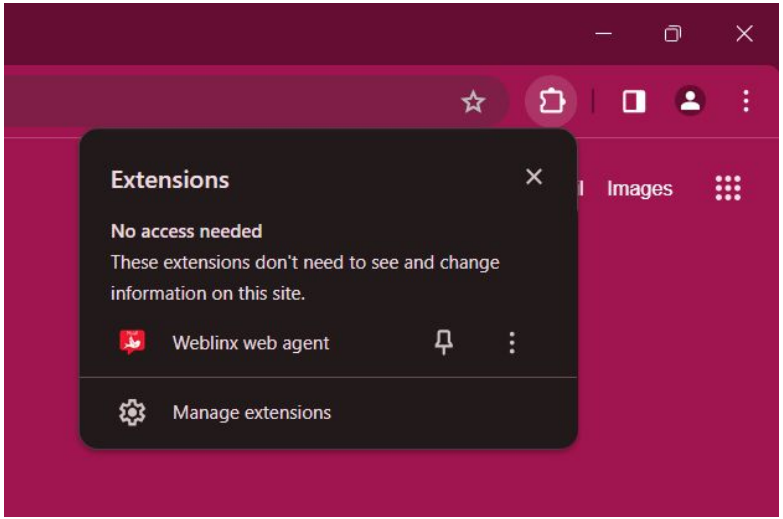
- Chrome extension chat box
- Automation tool
  - The automation tool, acts as an intermediary between the model and the user
- Models
  - DMR, selects the most relevant elements
  - Action model predicts the next action

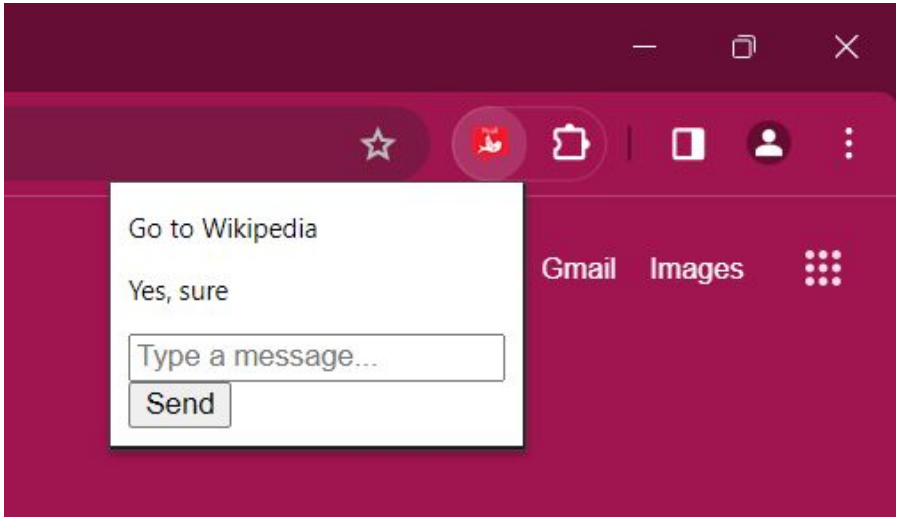
Integration

- Each sub-system is tested independently before the entire system is integrated to together

Chrome extension

The chrome extension is a chat box that allows the user to interact with the system where he specifies his request which will be handled by the model and automated using Selenium





Automation tool

The Selenium automation tool is used to **extract the relevant information** to give to the model such as the clean_html, viewport, screenshots etc as well as to **handle the different 'next actions'** that are specified by the model

```python
def execute_action(driver, action_str):
    command_type, args = parse_action_string(action_str)

    if command_type == "change":
        handle_change(driver, **args)
    elif command_type == "click":
        handle_click(driver, **args)
    elif command_type == "load":
        handle_load(driver, **args)
    elif command_type == "say":
        handle_say(**args)
    elif command_type == "scroll":
        handle_scroll(driver, **args)
    elif command_type == "submit":
        handle_submit(driver, **args)
    elif command_type == "text_input":
        handle_text_input(driver, **args)
```

Models

Two models are used to predict the *'next action'*

1. The **DMR model** which simplifies the predictions process of the action models by **finding the top candidates**
2. The **action model** that predicts the ***'next action'***

The model bases its prediction based on

- {clean_html} - > clean html page
- {utterances} - > The user's first and last 4 utterances
- {viewport} - > View port
- {candidates} - > the top candidates for this turn
- {action_history} -> Only the last 5 turns are provided

Simplified demo

Loading the wikipedia page