WebMemo A Mixed-Initiative System for Extracting and Structuring Web Content

Anonymous Author(s)

Abstract

When trying to make decisions and make sense of information on websites, users often struggle with the inefficiency and complexity of collecting and organizing web data. We introduce WebMemo, a novel web automation tool that addresses the challenges of information overload and inefficiencies in current bookmark and tab management systems. Leveraging Large Language Models (LLMs) and dynamic hierarchical structures, WebMemo enables users to seamlessly collect, organize, and retrieve information across web pages with minimal effort. WebMemo integrates structured views, dynamic tables, and customized hierarchies to support more efficient web interactions. Through proactive and flexible data collection based on high-level user input, WebMemo reduces the cognitive load and manual effort required for managing web content. Our contributions include a working system prototype and a discussion of the broader implications of AI-assisted information management.

ACM Reference Format:

Anonymous Author(s). 2025. WebMemo A Mixed-Initiative System for Extracting and Structuring Web Content. In *Proceedings of (UIST'25).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The Web is a rich source of information and services. People spend a significant amount of time navigating the Internet, collecting and organizing information in order to make informed decisions and fulfill their intentions [17]. Since people can only memorize and iterate on a limited amount of information in their minds, they have to keep a number of tabs open, frequently revisit previous websites, and locate valuable pieces of information for some data collection or decision-making tasks.

Web users often face the issue of overloaded tabs. The flat structure of the tabs and limited information provided by a tag makes it difficult for users to efficiently manage tabs or extract useful information. Consider the scenario where a user is shopping online for a new pair of headphones. They might open multiple tabs for product reviews and price comparisons. As the number of tabs increases, the user may struggle to switch back to specific tabs or recall which tab contains crucial information about headphone features or discounts. This tab overload can hinder the decision-making process, as vital details are buried under a clutter of indistinct tabs. Previous studies revealed competing pressures pushing for keeping tabs open (interaction and emotional costs) versus pushing for closing tabs (limited attention and resources) [7]. There is a disconnection between the increasing scope and complexity of users' online activities and the design of tabbed browsing. Existing tab-management tools have explored ways to reduce the friction of collecting web content, but they either require users to switch to a new platform or require users to manually identify the intended web content every time [20]. The challenge remains in how to proactively collect valuable information across different tabs without distracting users' attention on the target website.

Another challenge of collecting information from various websites is the time-consuming and repetitive nature of the task. For instance, when conducting a literature review, researchers often have to visit multiple academic databases and journals to find relevant articles. They may need to repetitively copy and paste the paper titles, author names, and sources for future review. This process can consume significant time on repetitive operations. In contrast to manual efforts, web automation techniques can scrape structured data from websites faster and more accurately. However, there is a learning barrier to creating web automation programs for users without a programming background. Existing research focuses on developing programming-by-demonstration (PBD) systems to facilitate non-programming web automation [9, 11, 26]. Since the PBD system synthesizes programs based on a few user demonstrations on the target website, it can only operate on a single website under the condition that the DOM structure remains the same. Therefore, data collection across different websites with different DOM structures remains a problem.

More recently, Large Language Models (LLMs) have been trained on a corpus that includes a large amount of web data. LLMs exhibit a remarkable ability to understand HTML code and UI elements [14], which presents new opportunities in solving the problems mentioned above. LLM-based web assistants are capable of understanding natural language commands from users and retrieving relevant information from web user interfaces(UI) [10, 31], which enables the system to collect information from multiple unseen websites without prior user demonstrations.

In this work, we present WebMemo, an LLM-based web system that allows users to proactively collect structured information they want from websites across different tabs. This project contributes the following:

- WebMemo, a novel system that leverages LLMs to collect and organize structured information from websites across different tabs.
- A streamlined workflow that enhances productivity in web activities.
- A within-subjects user study demonstrating the feasibility of WebMemo in comparison to a state-of-the-art tool OttoGrid [3].

UIST'25, Busan, Korea

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM

https://doi.org/XXXXXXX.XXXXXXX

2 Related Works

WebMemo builds on work in web automation, online sensemaking, and large language models (LLMs). We situate our work in all three.

2.1 Web Automation

Web automation is the process of automating tasks on websites that are typically performed by users, by simulating user events. Web automation can streamline repetitive tasks, improve efficiency, help users overcome accessibility issues (from permanent, temporary, or situational disabilities), and more [22, 27]. However, implementing web automation scripts is difficult and requires familiarity with programming languages such as JavaScript. Even for experienced programmers, it may take a significant amount of time to understand the page's structure and content sufficiently to code the automation scripts [19].

2.1.1 Programming by demonstration (PBD). Programming by demonstration (PBD) approaches attempt to lower the barrier of creating web automation programs for non-experts. Given a sequence of user demonstrations on a website, PBD systems could generate synthesized programs to repeat the same actions and apply them to similar elements on the website. Systems such as CoScripter [22] and Rousillon [8] are examples of PBD systems. However, the visual formats of the results programs from these systems still require familiarity with programming to understand them, which also makes it difficult for users to edit the program when errors occur. Systems such as SemanticOn [26], WebRobot [11], MIWA [9], and DiLogics [27] adopted a more advanced program synthesis technique. This approach allowed users to continuously provide more demonstrations to rewrite the synthesized program. Natural language descriptions and visual highlighting can also help users understand the automation program [9].

While in some ways an improvement to basic web automation, PBD systems have various limitations. First, they cannot handle arbitrary tasks on unseen websites. User demonstrations are often required whenever the DOM structure changes. Second, if the task requires data from pages from different sites (i.e., those that might use different templates), extraction may require complex scraping and multiple runs of PBD. Third, PBD systems require users to take the initiative to specify what content they want from the target website. Every new page may require the user to stop what they are doing, and either pull the data manually or initiate a PBD process. Either will disrupt a user's information consumption 'flow.' In our design, the system would take the initiative to identify and extract relevant information based on users' high-level natural language descriptions as the user browses the websites.

2.2 Online Sensemaking

Online sensemaking involves reading and understanding information online, and then collecting and organizing information into a structured format. Commercial tools such as Notion [5] and Evernote [2] allow users to capture part of the website or the website as a whole and then embed and organize the captured website into the self-defined document. Using these tools requires users to switch between different platforms and increases the mental cost. Then studies highlight the importance of minimizing disruptions in the sensemaking process; therefore, researchers have developed in-situ extensions for browsers. Fuse [20] is an in-situ clipping tool that allows users to manually collect the online information they want and organize it. More recently, LLMs demonstrated the abilities in sensemaking tasks [30]. Selenite[23] is an LLM-based tool that helps users' sensemaking processes.

Existing online sensemaking tools still require non-trivial efforts to extract useful information and structure the extracted information in an organized format that is suitable for decision-making. WebMemo eases the process of both extracting information and organizing it. The system automatically collects information based on high-level natural language instructions as users browse the Internet. Then it fills the output into a structured spreadsheet. The formatted output could help users quickly grasp useful information across different tabs and facilitate the decision-making process.

2.3 Large Language Models(LLMs) for Interactive Applications

Recently, there has been a surge in the development and application of LLMs. LLMs are trained on a large corpus of data and include billions of parameters, enabling the models to capture intricate linguistic patterns and relationships in the text and lead to unparalleled performance across broad NLP tasks. A remarkable feature of LLMs is few-shot or zero-shot learning [18]. LLMs can handle unseen tasks with very few or zero targeted examples. Additionally, models like GPT-3 [12] have shown abilities in in-context learning, which enables them to adapt to new tasks using only the context provided in the input prompt, without the need for direct training.

LLMs are increasingly applied in the field of user interfaces(UI). Some works focus on applying LLMs in Mobile UI [31, 32] and demonstrate that LLMs achieve competitive performance on challenging UI tasks without requiring dedicated training. Web UI, however, is distinct from Mobile UI in terms of more complex and larger content. The intricate and dynamic nature of Web UI makes it more difficult to interact with. Studies have shown that LLMs exhibit a reasonable level of performance in retrieving UI elements relevant to user instructions, despite some issues such as limited context window length and hallucination [14, 16]. WebMemo addresses the issue of limited context window length supported by large language models by filtering out the unnecessary code in raw HTML and extracting the text elements.

Existing LLM-based web automation tools such as Adept AI and Taxy AI [1, 4] are designed to take the agency of users to execute tasks on the websites. Additionally, tools like OttoGrid [3] have explored the use of LLMs within tale interfaces for online information retrieval. These tools suffer from high error rates and raise user concerns such as efficiency, usefulness, and user trust. WebMemo addresses the pain points in web activities in a different manner that proactively collects and organizes information without intervening in users' normal web activities and mental flows. WebMemo leverages the power of LLMs in understanding new websites, extracting relevant information based on high-level user instructions, and formatting unstructured web information into structured data.

3 Formative Study and Design Goals

3.1 Formative Study

In prior work[1], we conducted semi-structured interviews with 24 participants to understand the automation preferences of a broad variety of users [6]. Participants had a range of technical abilities (12 had technical backgrounds or worked in technical fields and 12 did not) and ages (half were over 55 years old, independent of technical background). We asked participants to provide 5–10 examples of web tasks they commonly performed, yielding a total of 150 tasks across participants. We asked participants about how automated agents could help them perform these tasks. Our design of WebMemo is inspired by the several of the findings from [6] and a re-analysis of the results of these interviews to assess how automation can improve users' browsing experience.

3.1.1 Prior Results. As we found in interviews [6], **users prefer to retain control over key decisions, but want the AI agents to provide supporting information.** Participants show a strong interest in using AI-assisted web agents, and semi-automated is preferred (rather than fully– or non-automated). While people were open to gathering more information and suggestions from AI such as "summarizing the pros and cons mentioned in the reviews" (P1) during online shopping, they preferred to "confirm the final step" (P5) due to concerns about errors and trustworthiness of the AI. These findings inspired us to design WebMemo as a semi-automated system, providing users with AI-driven insights and summaries while preserving user autonomy.

We also found that **people see time-saving as the biggest advantage of using AI in web activities** [6]. The prior study found that time-saving was mentioned 81 times across all tasks when the participants were asked about the benefits of automation. Participants mentioned that automation should be faster than manual processes, particularly by reducing repetitive efforts such as refreshing and re-entering the same information. This finding helps us narrow down the focus to reduce repetitive and time-consuming tasks by automating data collection, updating, and organizing.

3.1.2 Additional Findings. Beyond the results presented in [6], we also re-analyzed the results of our interviews in further depth.

Our re-analysis found that **a large number of web tasks involve information retrieval.** Of the 150 tasks described participants, 73% involved some form of *information retrieval*. 58% (87 of 150) required gathering information from multiple sources and 15% (22 of 150) from a single website. Tasks that required multiple sources include online shopping among multiple brands, vacation planning, gathering research information, and more. For example, some online online shopping tasks involve decision-making among different websites. Participants indicated that they would like to "combine answers from different sources synchronously" (P2). P11 mentioned that AI-assisted web automation tools could also help "provide multiple options if I had forgotten something" and "help me make better decisions." These findings inspired us to streamline the process of information retrieval and decrease mental and manual loads in the information-gathering process.

[1]This work is currently under review and thus anonymized.

Our re-analysis also found that **users prefer interacting with embedded web agents within their current browsing environment, rather than being redirected to external platforms.** When participants were asked about their envisioned user interface of the web agent, "an extension" (P5, P6, P14) and "a small window" (P12) were mentioned frequently because these are "embedded in the search engine" (P9) and "simple to use" (P14). On the other hand, participants were concerned that opening up new websites would "increase mental load" (P6). In response to this feedback, WebMemo is designed as an in-browser extension that integrates directly into the user's existing workflow without requiring them to open new windows or navigate to separate websites.

3.2 Design Goals

Based on the challenges and needs identified through the formative study and other prior work [7, 14, 16] we identified design goals for WebMemo. We briefly summarize each goal and provide rationale based on this past work.

DG1: Organize unstructured information from multiple sources into structured data. Studies have shown that people feel pressure for keeping (too) many tabs open for unfinished tasks and revisiting [7]. One of the key findings of the formative study was that 80% of the information retrieval tasks required gathering data from multiple sources. Participants expressed a strong desire to consolidate and structure this information to make comparison and analysis easier (e.g., P2 and P11). This inspired the first design goal of WebMemo, which is to help users organize unstructured information from various resources into structured data, minimizing their cognitive load when switching between tabs or websites.

DG2: Efficiently integrate into routine multi-tasked browsing with minimal effort (low learning curve, low mental and physical load). Previous research suggests that web systems should minimize distractions during information collection [7]. Participants in our formative study expressed concerns about increased mental load from external redirections when (AI) support tools were not embedded directly into their browsing environment (e.g., extensions or small windows). WebMemo was designed to integrate directly into the browser and to support different kinds of information consumption behaviors. Additionally, many users indicated that they sought time-saving solutions, which were mentioned 81 times across different tasks. Additionally, we know that for many individuals, multi-tasked [29] and non-linear [25] information seeking is standard behavior. Ideally, any tool should support the range of non-linear, interleaved, long-term and fragmented information-seeking behaviors that individuals undertake. Our second design goal focuses on creating a seamless, low-effort interface that integrates with users' existing workflows, minimizing disruptions and reducing physical and mental effort when interacting with the tool.

DG3: Ensuring collected data is easily validated and supports downstream tasks. Automation offers a number of advantages to users. New technologies such as large language models offer distinct advantages over previous approaches but also introduce their own problems (e.g., hallucinations, sycophancy [28], etc.). In the formative study, participants emphasized the need for control over decision-making tasks, expressing a preference for systems

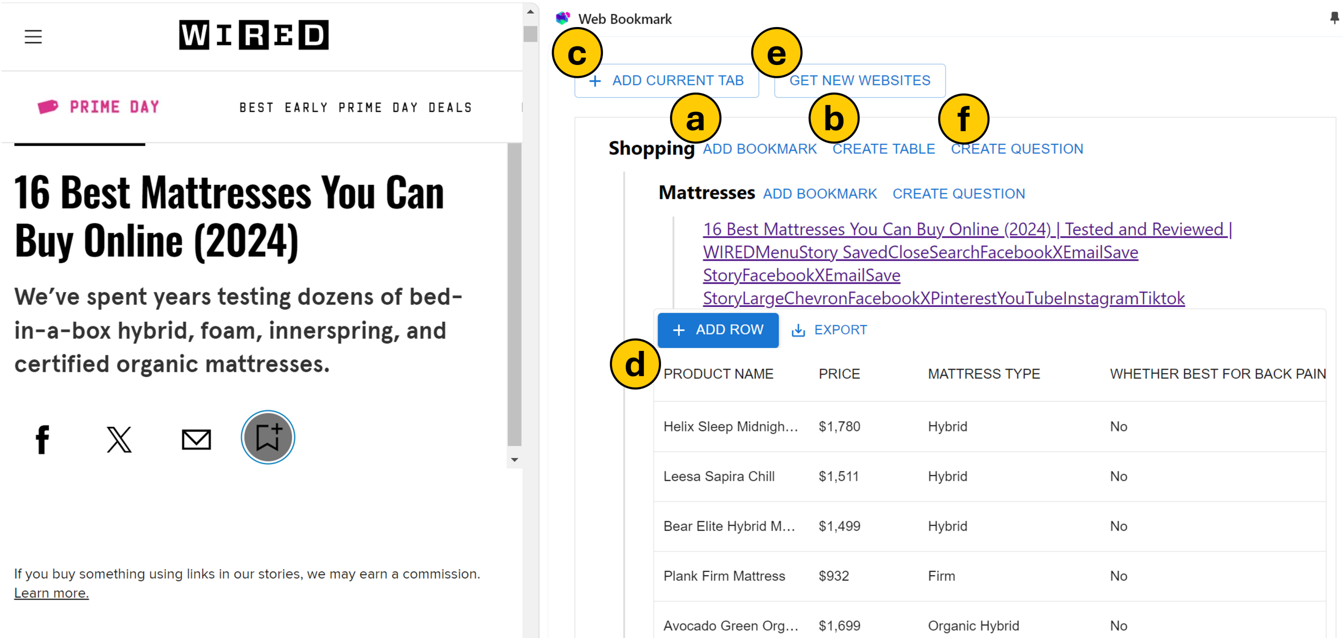

Figure 1: The main user interface of WebMemo. Users can create bookmarks ⓐ and tables ⓑ, and specify the column names in the table. When users encounter a website that they would like to scrape from, they can add the website to a bookmark ⓒ. As they scroll down the website, the table will be populated proactively ⓓ. Users can also directly retrieve data from new websites suggested by WebMemo (ⓔ).

that allow them to review and validate data before making final decisions. This feedback (e.g., P5 and P1) underscores the importance of designing a tool that not only collects information but presents it in a structured way to facilitate user validation and comparison. Information quality checks can be done both in real-time (as the extraction is happening) as well as post-hoc. In many situations, LLM errors are best mitigated as they happen (e.g., [13]). A well-structured data representation can play a crucial role in helping users validate, analyze, and make decisions based on the data collected from websites. When users gather information, a dynamic structured output allows them to visualize the data and prepare it for downstream tasks such as decision-making and further analysis. DG3 is thus aimed at ensuring that the data is presented and organized in a way that supports both quality assurance tasks *and* enables subsequent decision-making or analysis tasks.

3.3 Design Space

In developing WebMemo, the design space was carefully chosen to align with the design goals. The choices of design space involved key designs around how tables are created and modified, how rows of data are incorporated, and how data validation occurs. As shown in Table 1, these choices reflect critical distinctions from a state-of-the-art tool OttoGrid [3]. OttoGrid allows users to analyze, aggregate, and enrich data tables with AI assistance. In OttoGrid, users create a single table for a project at one time, format the column names, and add data to the table from online or local resources.

OttoGrid	WebMemo
Tables created at the start	Tables created/modified anytime
Rows incorporated in a group	Rows incorporated one at a time
Data validated at the end	Data validated as you go

Table 1: Comparison in design space between OttoGrid and WebMemo

WebMemo's design space choices are directly aligned with its three design goals. WebMemo allows for multiple tables to be created or modified dynamically (**DG1**). This feature contrasts OttoGrid, which requires a single table to be predefined at the start of the project. This ensures that as users browse different websites and gather data, the tool can adapt to changing needs, allowing for the organization of unstructured information into highly structured formats. WebMemo's design incorporates rows of data one at a time, as users scroll through a webpage (**DG2**). This decision enables the system to operate in the background and minimizes interruptions to users' regular browsing activities. By incorporating data incrementally, users don't need to stop and batch-process data, which enhances the multitasking capabilities of WebMemo. WebMemo's real-time data validation addresses a key challenge faced by OttoGrid, which validates data only at the end of data collection (**DG3**). By validating data as users browse, WebMemo enables them to immediately check the accuracy of the information collected.

4 WebMemo System

4.1 Usage Scenario

Consider Alice, a busy mother-to-be, who is researching two important purchases: a mattress that provides back support and a lightweight baby stroller that is suitable for travel. Alice is taking her time to read up on possible options because the mattress is an expensive investment, and the stroller isn't needed for a few months. She knows the mattress business is competitive and doesn't trust the search-engine-optimized recommendation pages that come up at the top of search results. Alice prefers to do her own research or look at pages her close friends forward her. She also realized that some of the pages that she bookmarked or left as open tabs when she first got pregnant had some great suggestions. Digging through these will take some time. Thus, her search is somewhat casual and is done between other tasks over an extended period. Alice takes advantage of WebMemo to support her research.

Using WebMemo, Alice begins by creating a high-level bookmark 'Shopping' with a 'Mattresses' sub-category (Figure 1ⓐ). In anticipation of collecting her data, Alice sets up a table under the 'Mattresses' category with column names relevant to her decision-making criteria, such as 'Price,' 'Type,' and 'Back Support' for the mattress (Figure 1ⓑ). She's found a page of mattress recommendations at Wired.com, a site she had good luck with before. She adds the current tab to the 'Mattresses' bookmark and starts browsing the page (Figure 1ⓒ).

As she browses through the product details, WebMemo proactively scrapes the website in the background, dynamically populating the data table linked to her 'Mattresses' bookmark with information from the product page (**DG1**), as shown in Figure 1ⓓ. The system prompts large language models (LLMs) with web content (text only) and a high-level natural language description (the column names of the data table). Based on this input, the LLMs return data to be filled into the table, which WebMemo then dynamically updates according to the position she has scrolled to (**DG2**). This ensures a seamless integration between Alice's browsing and the information extraction process, minimizing interruptions to her flow. Additionally, WebMemo memorizes the URL for each row in the table, allowing it to navigate back to the original source of a specific entry when needed. If she wants to verify any specific entry, Alice can click on the respective data cell, prompting WebMemo to navigate back to the original webpage and highlight the corresponding information on the site (Figure 3). This highlighting feature also helps Alice quickly spot any discrepancies or errors in the table (**DG3**). She can manually edit any data cells if noticing any incorrect data entries.

Satisfied with her initial exploration, Alice decides to visit another mattress website. She adds this new site to her 'Mattresses' bookmark and continues the same seamless data-gathering process, with WebMemo automatically extracting key product details (**DG2**). Alice can use built-in sorting features to sort individual columns. She can also directly ask questions about the table (**DG3**), as shown in Figure 1ⓕ. When Alice poses a question, WebMemo prompts the LLMs with both her question and the complete data table (containing the collected information). The LLMs then analyze the data and return an answer, allowing Alice to gain insights without needing to manually sift through the table. This feature enhances the efficiency of her decision-making process by delivering relevant answers based on the data gathered.

Later, when Alice turns her attention to the baby stroller. She follows the same steps: setting up a new bookmark with a table and adding websites to her Baby Strollers bookmark (**DG1**). However, this time, Alice wants to automate part of her research. Instead of manually browsing through multiple stroller websites, she clicks the 'Get New Websites' button (Figure 1ⓔ), and WebMemo displays several relevant websites she might be interested in (**DG2**) suggested by LLMs. As shown in Figure 4, Alice quickly reviews the suggestions and clicks 'Add' to include new websites in her bookmark. Without needing to visit the pages herself, WebMemo scrapes and adds relevant data about baby strollers to her table, allowing her to make well-informed decisions without investing more time in manual browsing.

4.2 System Design Details

Based on the challenges and common issues discussed in the formative study and previous studies [7, 14, 16], we summarize the design goals for WebMemo and elaborate on the rationale for each goal below.

DG1: Organize unstructured information from multiple sources into structured data. WebMemo introduces a dynamic bookmark structure that organizes unstructured information into a structured, hierarchical system. Each bookmark can contain one of two elements: (1) subcategories, or (2) a list of website URLs accompanied by a data table that holds data extracted from those websites. Subcategories can be further broken down into additional levels of organization, such as folders or tables, allowing for a more granular grouping of content. This hierarchical structure allows users to group related content from different websites under meaningful categories, like a folder system, but also supports dynamic tabular views that summarize key information within each category. These tables can then be expanded to reveal specific data points, and users can create further custom views. By integrating hierarchical and table-based representations, WebMemo allows users to structure and visualize their bookmarks in a way that supports efficient data retrieval and decision-making. Naturally, users have the flexibility to add, edit, and delete any bookmarks and data in each table to tailor them precisely based on their specific needs. WebMemo also introduces the following features to support DG1:

Combine unstructured websites into structured formats. WebMemo transforms unstructured web content into structured formats by synchronizing information collected across different tabs, even if the websites have varying DOM structures. It does so by prompting large language models (LLMs) with the raw text content extracted from the HTML data and the column names of the data table, while stripping out the HTML code. This process has little to no performance degradation, ensuring that the transformation from unstructured to structured data is efficient and seamless. This allows users to capture and organize valuable data as they browse, revisiting it at any time from the browser's sidebar. By structuring this information, WebMemo significantly reduces the cognitive load associated with managing numerous similar tabs, eliminating the need to search through them to locate past websites.

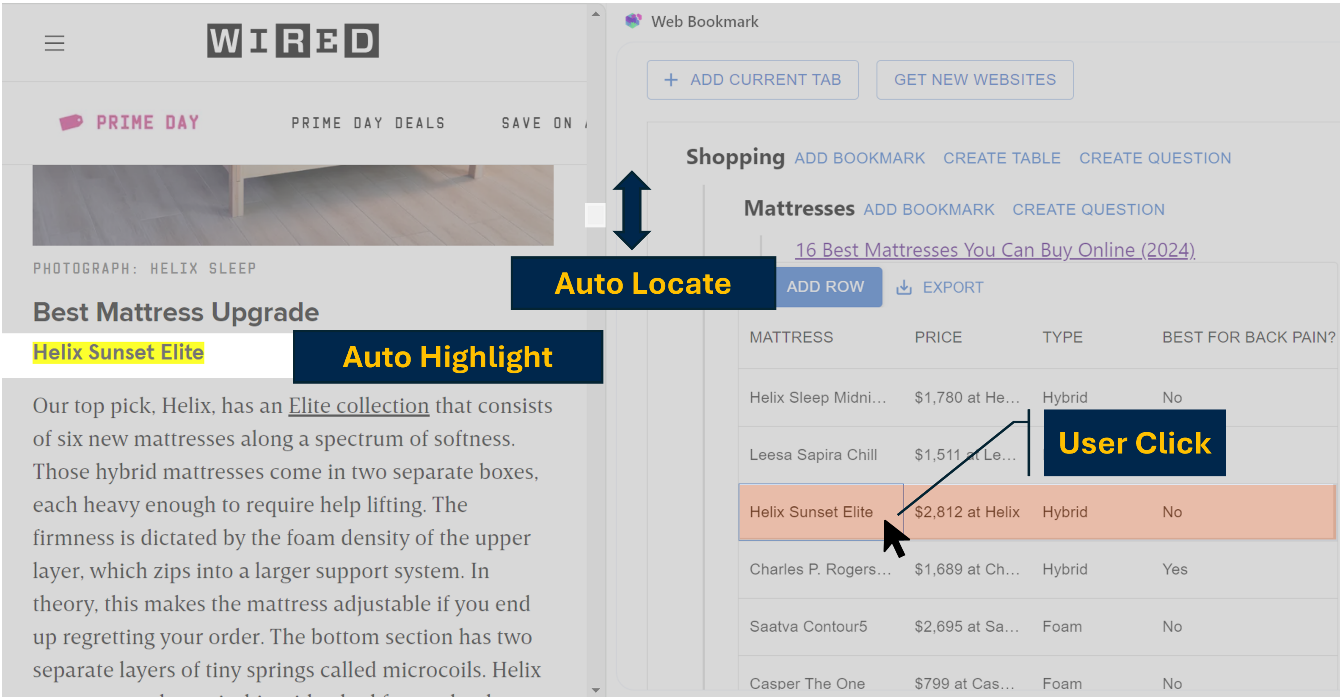

Figure 2: When users click a data cell in the table, WebMemo will automatically redirect them to the original source website, scroll to the exact position of the data, and highlight the corresponding information.

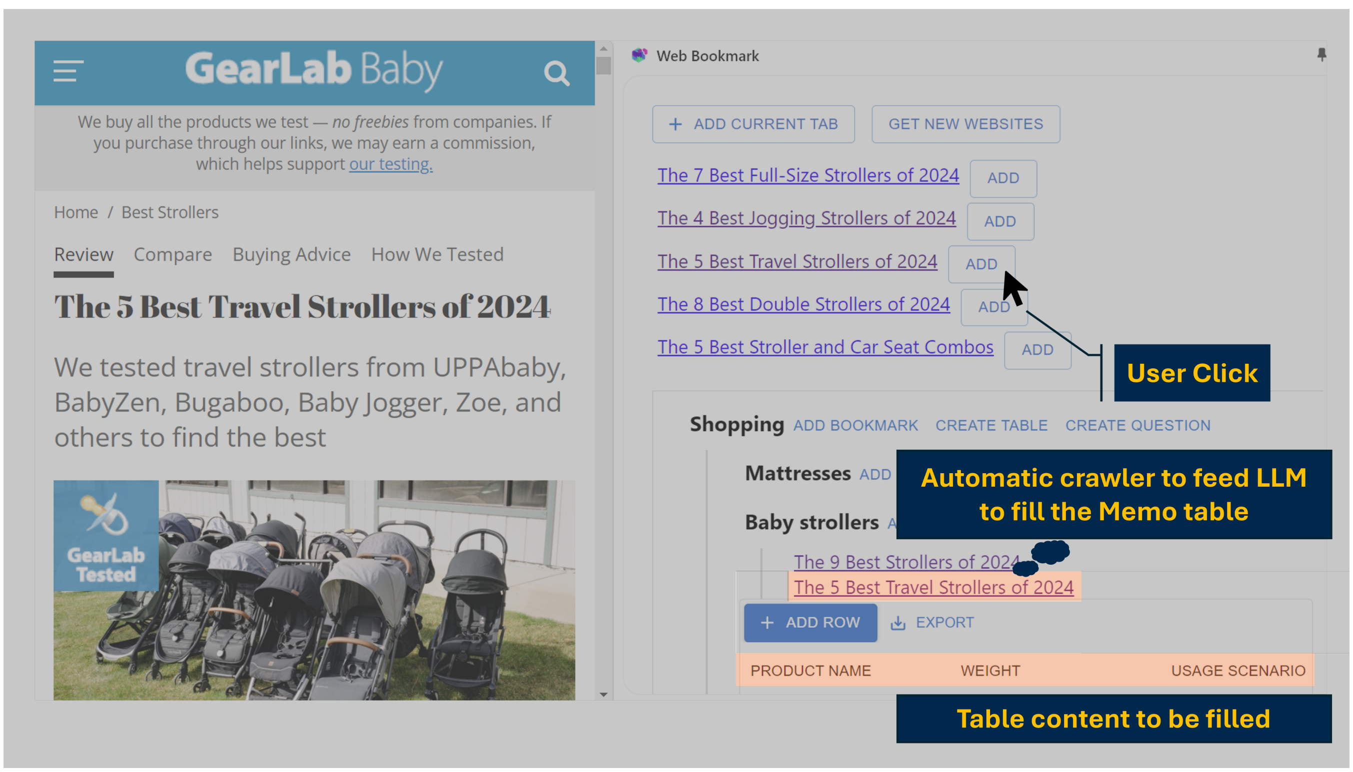

Figure 3: Users can directly retrieve data from the list of suggested websites that are similar to the current website.

Support for multitasking. Multitasking is a common behavior in web browsing, extensively studied in previous work [21, 24]. Spink et al. reported that 81% of the two-query browsing sessions included multitasking [29]. This inspired WebMemo's hierarchical structure, which functions similarly to a bookmarking tool, supporting users in managing multiple tasks seamlessly. The hierarchical structure allows users to create and modify data tables at any point during a browsing session, ensuring flexibility while multitasking.

DG2: Efficiently integrate into routine multi-tasking browsing with minimal effort (low learning curve, low mental and physical load). To minimize the physical, temporal, and mental effort with online information retrieval tasks, WebMemo introduced the following features:

Lightweight and in-situ integration. WebMemo is implemented as a Chrome extension that occupies a sidebar within the browser, allowing users to collect information without switching between platforms. The sidebar remains accessible even when users open new tabs, keeping the tool readily available without interrupting their workflow.

Proactive data collection. WebMemo minimizes user effort by following high-level natural language instructions to collect data. Once users provide brief instructions (e.g., column names for a data table), WebMemo automatically applies the same patterns to gather relevant information across different tabs. In the backend, WebMemo memorizes and computes the y-axis position for each data row. As the user scrolls down a website, WebMemo will populate the table dynamically according to the position that the user has scrolled. This eliminates the need for users to manually indicate what content to scrape from each website.

Dynamic and flexible data integration. When users add a new website to WebMemo's bookmarks, the tool proactively suggests bookmarks and columns based on the website's content and the existing bookmark title. This AI-assisted recommendation feature helps users make sense of new content and accelerates the information-gathering process. As users scroll through websites, rows are dynamically added to the data table. New websites can be integrated into existing bookmarks at any time, and columns can be added automatically based on the data across bookmarked websites. Users can also import data directly from recommended websites relevant to their current context, further enhancing efficiency. WebMemo collects all URLs on the current website and prompts the LLMs with the complete list of URLs along with the existing data table. The LLMs return a selected list of URLs that are likely to contain useful information relevant to the table.

These features help WebMemo integrate effortlessly into users' existing browsing routines, reducing cognitive load and making information collection more efficient.

DG3: Ensuring collected data is easily validated and supports downstream tasks.

A well-structured data representation plays a crucial role in helping users validate, analyze, and make decisions based on the data collected from websites. When users gather information, a structured output allows them to visualize the data and prepare it for downstream tasks such as decision-making and further analysis.

For decision-making, the structured output helps users compare and evaluate various options more efficiently. To enhance data validation and support downstream tasks, WebMemo introduces the following key features:

Click-to-Source Mapping. Users can click on any data cell in the table, and WebMemo will automatically redirect them to the original source website, scroll to the exact position of the data, and highlight the corresponding information. WebMemo memorizes the URL for each row in the table, allowing it to navigate back to the original source of a specific entry when needed. When the user clicks a cell, WebMemo will navigate to the source URL, match the text string pattern, and highlight the corresponding text on the source website. This feature creates a direct mapping between the AI-generated data and the original content, ensuring transparency and facilitating easy cross-referencing.

Sorting and Data Manipulation. After users have completed collecting data in a table, WebMemo provides built-in sorting functionality, allowing users to organize each column in ascending or descending order. This simple yet effective feature helps in quickly making sense of the data.

Question-Answering with LLMs. Users can interact with the data table by asking questions and leveraging large language models to gain insights or clarify specific points about the data. WebMemo prompts the LLMs with both her question and the complete data table (containing the collected information). The LLMs then analyze the data and return an answer. This supports a deeper understanding of the collected information.

These features increase users' trust in the data collected by allowing them to verify AI-generated content against the original sources. The validation and organization capabilities also help users efficiently complete real-world web tasks, such as comparing products for online shopping based on their customized criteria.

5 Implementation

WebMemo is implemented as a Chrome extension. The system is built using HTML, TypeScript, and the React JavaScript library. We used MongoDB for the backend, which handles data storage, website pre-fetching, and connection with the OpenAI API for tabular data generation.

To ensure consistent data collection, we implemented a separate backend process that pre-fetches the entire web page as soon as the user opens a new tab, rather than processing the page incrementally as the user scrolls. This approach allows us to provide a complete and coherent view of the page content to the language model (LLM), such as GPT-4, at the earliest stage. By processing the entire page in one pass, we ensure that the collected data remains consistent and prevent the risk of fragmenting information that might occur if the page were processed bit by bit. This consistency is especially crucial when using LLMs, as they can generate more accurate and context-aware tabular data when given access to the full web page from the start.

We used the JSON mode in OpenAI API so that the response from GPT is in JSON format. Upon receiving the response from API, the extension can format and display the data in the spreadsheet. We selected GPT-4o as the model due to its strong performance and efficiency, making it suitable for our prototyping needs and adaptable across multiple application areas. Nevertheless, the core of our contribution lies in the dynamic bookmark structure, the seamless integration of hierarchical and tabular views, and the user-centric interface that supports efficient multitasking and data organization—all of which are not tied to any particular model or the accuracy of the model.

6 User Study

A within-subject study was conducted to evaluate WebMemo's efficacy in extracting and structuring web content. We used OttoGrid as the baseline tool [3], which was designed with a high-level goal similar to that of WebMemo. However, OttoGrid represents a very different point in the design space of information extraction tools–enforcing a workflow that is more focused, rigid, and 'modularized.' In this study, we were interested in the following research questions:

- RQ1: How does WebMemo influence the efficiency of users in understanding websites and making informed decisions?
- RQ2: Does WebMemo seamlessly integrate into users' browsing activities without causing disruptions, while also reducing cognitive load?
- RQ3: Does WebMemo help users link AI-generated data with its original source and build higher user confidence?

6.1 Method

6.1.1 Participants. We recruited 12 participants (five male, seven female) through social media and mailing lists. Participants were required to be 18 or older and fluent in English. All participants were reported to be familiar with web technologies and had substantial experience using the internet for both professional and personal purposes. When asked about their willingness to use a web-based AI-assisted automation tool in the future, participants responded with an average score of 6.08 on a 7-point scale, indicating a generally positive attitude towards incorporating such tools into their online routines.

6.1.2 Tasks. The study employed a within-subjects design in which participants were given two tasks to complete, each under a different condition, with the order of conditions counterbalanced. Participants participated in two different types of tasks during the study: an information retrieval task focused on academic researchers and an online shopping task focused on product comparison.

Task 1 (Researcher Identification): Participants were tasked with identifying and listing at least three researchers who were not affiliated with Harvard University. They were given access to websites listing research fellows and guided to use specific keywords to identify suitable individuals. This task tested how well WebMemo supports the efficient extraction and understanding of structured information from complex academic websites (RQ1) and whether users could easily link AI-generated data back to its original source for increased confidence in the results (RQ3).

Task 2 (Online Shopping): Participants completed two shopping-related subtasks. In the first subtask, they searched for hybrid mattresses costing less than $1,500 and suitable for individuals with back pain, using review websites with varying product criteria. The second subtask involved finding two baby strollers that were travel-friendly and weighed less than 20 lbs, based on product comparison sites. These tasks allowed us to evaluate how well WebMemo integrates into users' browsing activities without causing disruptions (RQ2), and how it helps users gather and organize product data under customized constraints (RQ1). Furthermore, users' ability to trust the AI-generated data, linked to the original product sources, was key to understanding how the tool builds user confidence (RQ3).

6.1.3 Protocols. In the user study, participants began by completing a demographic survey and a consent form. They then watched a tutorial and demo video for one of the web tools (either WebMemo or OttoGrid), followed by a 10-minute period of free exploration to familiarize themselves with the tool. After this, participants completed Task 1 using the assigned tool and took a quiz to assess their understanding of the task. Following the quiz, they filled out a post-task survey to capture their feedback and experiences. The survey also included a series of Likert-scale questions, where participants rated the effectiveness and usefulness of key features in the tool. To assess the cognitive load of using each tool, the survey incorporated six NASA Task Load Index (TLX) questions to evaluate participants' perceived workload [15]. Participants were also asked to indicate whether they would be interested in using the tool for their routine web activities, and what improvements could be made to enhance the tool. Then participants repeated the same process with the other tool to complete Task 2. We imposed a 30-minute limit per task to keep participants from spending excessive time on any one activity.

6.2 Results

6.2.1 User Performance. Figure 5 shows the performance of the participants using WebMemo versus OttoGrid in terms of task completion time. When using WebMemo, all 12 participants successfully completed the assigned task. When using OttoGrid, one participant was unable to complete the task due to exceeding the 30-minute time limit. Table 2 shows the average task completion times and standard deviation for WebMemo and OttoGrid. WebMemo consistently took less time on average to complete both tasks. The larger standard deviations for OttoGrid in both tasks indicate greater variability in user performance with that tool compared to WebMemo. These results suggest that WebMemo helped participants retrieve information from multiple websites and make inferences more efficiently. Based on the observation data, there are several reasons why WebMemo save time compared to OttoGrid. First, WebMemo users have fewer attempts to complete tasks. When using WebMemo, participants made an average of 1.25 attempts, while they made an average of 1.45 attempts when using OttoGrid. Specifically, an attempt was defined as a single effort to retrieve data from a website. If the tool failed to retrieve data and populate the table, the number of attempts per URL was limited to a maximum of three. Second, the user comments from the observation data suggest that WebMemo is "clear and straightforward" (P1), while Otto has feedback indicating challenges, such as "slow loading" (P2) and "difficulty in setting up the table" (P5). WebMemo enables users to "verify the data by visual correspondence anytime" (P6) and do not have to "read a new website from the beginning" (P6). On the other hand, users spent more time switching between tabs to check the correctness of data collected in OttoGrid.

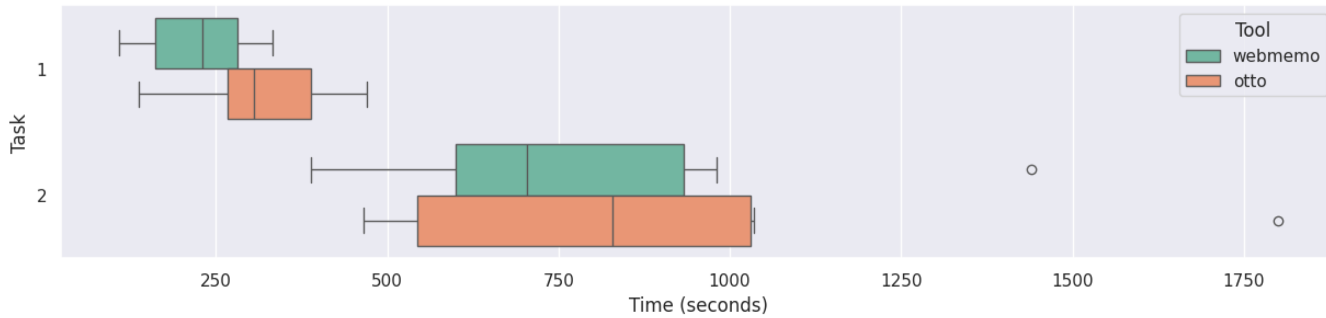

Figure 4: Task Completion Time for WebMemo vs OttoGrid

Task	Condition	Mean Time (seconds)	Std. Deviation (seconds)
Task 1 (Researcher Information)	WebMemo	223.17	85.45
	OttoGrid	314.67	117.04
Task 2 (Online Shopping)	WebMemo	801.67	370.77
	OttoGrid	911.33	500.11

Table 2: Mean task completion times and standard deviations for WebMemo and OttoGrid in Task 1 and Task 2.

6.2.2 User Ratings. Table 3 shows the user ratings for individual goals for WebMemo and OttoGrid. In terms of collecting and memorizing web data without interrupting website browsing, WebMemo significantly outperformed OttoGrid, with a mean score of 4.92 compared to OttoGrid's 3.33 (p = 0.0048). This suggests that WebMemo provides a smoother experience for users when it comes to browsing without interruptions. Participants reported, "no interrupting when switching between multiple websites" (P5). In the observation of the user study, only 2 out of 12 participants manually checked the original websites when using OttoGrid to retrieve data. Users felt "lost in data" and were "not able to find the original source" (P3) when multiple new resources were referenced in OttoGrid.

When it came to organizing unstructured website data into a structured format, both systems performed similarly. WebMemo had a slightly higher mean score of 4.67 compared to OttoGrid's 4.42, but the difference was not statistically significant (p = 0.5428). This indicates that both tools were similarly effective in transforming unstructured data into a usable format.

For the goal of reducing the mental load of memorizing information from multiple resources, WebMemo once again scored higher (M = 4.75, SD = 0.45) compared to OttoGrid (M = 4.25, SD = 1.29), although the difference was not statistically significant (p = 0.2178). While WebMemo showed an advantage, users found both systems somewhat helpful in reducing cognitive effort.

Lastly, saving time compared to manually scraping web data was another area where WebMemo performed significantly better than OttoGrid. WebMemo had a mean score of 4.67, while OttoGrid scored 3.67 (p = 0.0388), indicating that users perceived WebMemo as a more time-efficient solution for gathering data.

6.2.3 User Work Loads. Table 4 summarizes the participants' responses to the NASA Task Load Index (TLX) questionnaire, comparing the perceived workloads when using WebMemo and OttoGrid. For mental demand, WebMemo had a significantly lower score (median = 1.5, mean = 1.83 ± 1.17) compared to OttoGrid (median = 3.0, mean = 3.58 ± 2.37). Users noted that WebMemo reduced the cognitive effort needed to navigate the tool with embedded highlighting, sorting, and question-answering features. In contrast, users found OttoGrid more mentally demanding with the "learning curve and complicated column formatting" (P2). The mental load of using OttoGrid also comes from less user trust in the retrieved data and users feel the need to "check the original websites by myself" (P3).

The mean user trust score for WebMemo was 4.42 (SD = 0.79), while OttoGrid received a lower mean score of 3.33 (SD = 1.37). The difference between the two systems was statistically significant (p = 0.027). P3 explained that "dynamically showing rows makes me understand that WebMemo is reading the website." When users click on a row in WebMemo, the system navigates to the corresponding website, positions the view at the linked data, and highlights the relevant information on the original page. This feature "increases confidence" (P2) in the accuracy and relevance of the data. The highlighting feature also helped users detect errors and mismatches in the data.

For effort, WebMemo scored significantly lower (median = 1.0, mean = 2.00 ± 1.41) compared to OttoGrid (median = 4.0, mean = 4.33 ± 2.09, $p < 0.05$). Users appreciated that WebMemo required minimal effort, describing it as "easy to use" and requiring "less manual interaction" (P7). P12 mentioned that WebMemo streamlined the entire workflow by allowing them to complete website browsing, information retrieval, data validation, and decision-making within a single interface.

6.2.4 User Feedback. In the post-study interview, 11 out of 12 participants expressed willingness to use WebMemo in the future. Participants identified user cases that span both the personal and

Statement	Condition	M	SD	p
Collect and memorize web data relevant to the task without interrupting website browsing.	WebMemo OttoGrid	4.92 3.33	0.29 1.72	0.0048*
Organize unstructured website data into a structured format.	WebMemo OttoGrid	4.67 4.42	0.65 1.24	0.5428
Reduce mental load of memorizing information from multiple resources.	WebMemo OttoGrid	4.75 4.25	0.45 1.29	0.2178
Save time compared to scraping web data manually.	WebMemo OttoGrid	4.67 3.67	0.65 1.44	0.0388*

Table 3: Mean scores (M), standard deviations (SD), and p-values for WebMemo and OttoGrid survey responses (on a 5-point scale). Statistically significant differences ($p < 0.05$) are marked with an asterisk (*).

	Mental demand	Physical demand	Temporal demand	Performance	Effort	Frustration
OttoGrid	3.0 (3.58 ± 2.37)	3.0 (3.58 ± 2.46)	3.0 (3.92 ± 2.29)*	4.0 (3.92 ± 2.04)*	4.0 (4.33 ± 2.09)*	3.0 (3.75 ± 1.62)
WebMemo	1.5 (1.83 ± 1.17)	1.5 (1.83 ± 1.17)	1.0 (1.67 ± 1.03)*	1.5 (1.67 ± 0.78)*	1.0 (2.00 ± 1.41)*	1.0 (1.50 ± 0.67)

Table 4: Participants' responses to NASA TLX questions (on a scale from 0 to 7). Format: median (mean ± standard deviation). Statistically significant differences ($p < 0.05$) through t-tests are marked with an asterisk (*).

professional domains. These use cases include conducting literature reviews, managing graduate school applications, and tracking product comparisons and deals during online shopping. A few participants also highlighted that WebMemo could also be useful for ongoing projects, such as long-term data gathering and business data analysis where continuous updates are needed.

When asked what could be changed to improve the tool, the participants provided insightful suggestions. P10 noted the need for more intuitive guidance on the user interface to help new users quickly understand how to retrieve data. Four participants also mentioned the desire to retrieve data directly without browsing the websites by themselves.

7 Discussion

7.1 Limitations

WebMemo has several limitations. First, the system's reliance on LLMs for extracting data may sometimes lead to inaccuracies and the collection of wrong data into the spreadsheet. The system lacks robust error detection and correction mechanisms. While the tool allows users to manually correct data, it does not proactively detect inconsistencies or flag potentially incorrect entries. If the requested data is unavailable on the source website, the tool leaves the corresponding cells blank or marked as 'N/A'. It can be problematic for users working with large datasets, where such gaps or frequent inaccuracies can accumulate and become increasingly difficult to manage. Second, WebMemo lacks the ability to learn from users' prior activities or interactions. The system does not improve its extraction process based on previous corrections or user preferences. As a result, users are required to repeatedly correct similar errors or reconfigure the tool for websites they frequently visit, which can lead to frustration and inefficiency in long-term usage.

7.2 Future Works

To address these limitations, several avenues for future work can be explored.

7.2.1 Robust error detection and correction. Incorporating a more advanced error detection and correction system would significantly enhance the tool's reliability. This could involve adding automated validation mechanisms that flag inconsistencies or missing data before they are added to the spreadsheet. The system can infer the data type in a column from the column name and double-check whether the collected data is in the correct format. If missing data or incorrect formatting is detected, the system could prompt the user to review and correct them proactively.

7.2.2 Learn from user interactions. By implementing a feedback loop, the system could refine its data extraction processes over time according to user preferences and patterns of usage. Future systems can include few-shot learning or reinforcement learning techniques to adjust to individual workflows.

8 Conclusion

In summary, WebMemo addresses the challenges of web data collection by leveraging the capabilities of large language models (LLMs) to proactively gather and organize information across multiple tabs in a browser. The system provides a solution to the problem of tab overload by synchronizing data from different websites into a structured format, enabling users to navigate the web with ease and efficiency. By providing lightweight, in-situ, and proactive information collection, WebMemo ensures that users remain focused on their tasks without needing to manually curate data from various sources.

Furthermore, the system's ability to structure output in a hierarchical bookmarking format facilitates a more efficient decision-making process. By providing a clear representation of the data, WebMemo enables users to quickly analyze and manipulate the information collected, supporting a range of downstream tasks from research to product comparisons. This structured approach significantly reduces the mental burden of managing multiple tabs, offering a streamlined workflow for online information gathering.

WebMemo's implementation as a Chrome extension demonstrates how innovative use of LLMs can simplify complex tasks and enhance productivity. Its design principles, rooted in addressing the challenges faced in web automation and information sensemaking, provide a robust framework for future development. As web interactions continue to evolve, tools like WebMemo represent a step forward in enhancing user experience and efficiency in online activities.

References

[1] [n. d.]. Adept: Useful General Intelligence — adept.ai. https://www.adept.ai/. [Accessed 19-04-2024].

[2] [n. d.]. Best Note Taking App - Organize Your Notes with Evernote — evernote.com. https://evernote.com/. [Accessed 01-05-2024].

[3] [n. d.]. Otto — ottogrid.ai. https://ottogrid.ai/. [Accessed 08-10-2024].

[4] [n. d.]. Taxy AI — taxy.ai. https://taxy.ai/. [Accessed 19-04-2024].

[5] [n. d.]. Your connected workspace for wiki, docs & projects | Notion — notion.so. https://www.notion.so/. [Accessed 01-05-2024].

[6] Anonymized Author(s). 2024. Understanding Challenges and Needs of Using AI in Web Automation Systems. *Under Review* (2024).

[7] Joseph Chee Chang, Nathan Hahn, Yongsung Kim, Julina Coupland, Bradley Breneisen, Hannah S Kim, John Hwong, and Aniket Kittur. 2021. When the tab comes due: challenges in the cost structure of browser tab usage. In *Proceedings of the 2021 CHI conference on human factors in computing systems*. 1–15.

[8] Sarah E Chasins, Maria Mueller, and Rastislav Bodik. 2018. Rousillon: Scraping distributed hierarchical web data. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 963–975.

[9] Weihao Chen, Xiaoyu Liu, Jiacheng Zhang, Ian Iong Lam, Zhicheng Huang, Rui Dong, Xinyu Wang, and Tianyi Zhang. 2023. MIWA: Mixed-Initiative Web Automation for Better User Control and Confidence. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–15.

[10] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems* 36 (2024).

[11] Rui Dong, Zhicheng Huang, Ian Iong Lam, Yan Chen, and Xinyu Wang. 2022. WebRobot: web robotic process automation using interactive programming-by-demonstration. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 152–167.

[12] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines* 30 (2020), 681–694.

[13] Sandy J. J. Gould, Duncan P. Brumby, and Anna L. Cox. 2024. ChatTL;DR – You Really Ought to Check What the LLM Said on Your Behalf. In *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems (CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 552, 7 pages. doi:10.1145/3613905.3644062

[14] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. 2022. Understanding html with large language models. *arXiv preprint arXiv:2210.03945* (2022).

[15] SG Hart. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Human mental workload/Elsevier* (1988).

[16] Faria Huq, Jeffrey P Bigham, and Nikolas Martelaro. 2023. " What's important here?": Opportunities and Challenges of Using LLMs in Retrieving Information from Web Interfaces. *arXiv preprint arXiv:2312.06147* (2023).

[17] Aniket Kittur, Andrew M Peters, Abdigani Diriye, Trupti Telang, and Michael R Bove. 2013. Costs and benefits of structured information foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2989–2998.

[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[19] Rebecca Krosnick and Steve Oney. 2021. Understanding the challenges and needs of programmers writing web automation scripts. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–9.

[20] Andrew Kuznetsov, Joseph Chee Chang, Nathan Hahn, Napol Rachatasumrit, Bradley Breneisen, Julina Coupland, and Aniket Kittur. 2022. Fuse: In-Situ Sensemaking Support in the Browser. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–15.

[21] Janette Lehmann, Mounia Lalmas, Georges Dupret, and Ricardo Baeza-Yates. 2013. Online multitasking and user engagement. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 519–528.

[22] Gilly Leshed, Eben M Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1719–1728.

[23] Michael Xieyang Liu, Tongshuang Wu, Tianying Chen, Franklin Mingzhe Li, Aniket Kittur, and Brad A Myers. 2023. Selenite: Scaffolding decision making with comprehensive overviews elicited from large language models. *arXiv preprint arXiv:2310.02161* (2023).

[24] Rongjun Ma, Henrik Lassila, Leysan Nurgalieva, and Janne Lindqvist. 2023. When browsing gets cluttered: exploring and modeling interactions of browsing clutter, browsing habits, and coping. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–29.

[25] Gary Marchionini. 1992. Interfaces for end-user information seeking. *Journal of the American society for information science* 43, 2 (1992), 156–163.

[26] Kevin Pu, Rainey Fu, Rui Dong, Xinyu Wang, Yan Chen, and Tovi Grossman. 2022. Semanticon: Specifying content-based semantic conditions for web automation programs. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.

[27] Kevin Pu, Jim Yang, Angel Yuan, Minyi Ma, Rui Dong, Xinyu Wang, Yan Chen, and Tovi Grossman. 2023. DiLogics: Creating Web Automation Programs with Diverse Logics. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–15.

[28] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548* (2023).

[29] Amanda Spink, Minsoo Park, Bernard J Jansen, and Jan Pedersen. 2006. Multitasking during web search sessions. *Information Processing & Management* 42, 1 (2006), 264–275.

[30] Sangho Suh, Bryan Min, Srishti Palani, and Haijun Xia. 2023. Sensecape: Enabling multilevel exploration and sensemaking with large language models. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–18.

[31] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.

[32] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Empowering llm to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272* (2023).