Promises and Pitfalls of Using LLMs for Scraping Web UIs

Rebecca Krosnick
rkros@umich.edu
University of Michigan, Computer Science & Engineering
Ann Arbor, Michigan, USA

Steve Oney
soney@umich.edu
University of Michigan, School of Information
Ann Arbor, Michigan, USA

ABSTRACT

ChatGPT and other publicly available large language models (LLMs) put AI into the hands of everyday computer users, offering the possibility of automating computer tasks. One candidate task is web scraping. We informally experimented with ChatGPT to explore the potential promises and pitfalls of using it for scraping data from web user interfaces. We share our observations and considerations for future human-LLM web scraping systems.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**; *Empirical studies in HCI*; • **Information systems** → *Data extraction and integration*; • **Computing methodologies** → *Machine learning*.

KEYWORDS

large language models, ChatGPT, web scraping, natural language interfaces

ACM Reference Format:
Rebecca Krosnick and Steve Oney. 2023. Promises and Pitfalls of Using LLMs for Scraping Web UIs. In *CHI '23 Workshop: The Future of Computational Approaches for Understanding and Adapting User Interfaces, April 23, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The recent release of the large language model (LLM) ChatGPT [1] has put generative AI in the hands of everyday computer users. Users simply need to type a natural language request and ChatGPT generates often plausible-looking responses – offering the possibility to automate computer tasks. However, ChatGPT is known to produce incorrect answers while sounding very confident, which can be quite dangerous if left unchecked. Current popular automation use cases for ChatGPT include writing emails, correcting grammar, writing code, and formatting data. Given its text and often formulaic nature, another use case that may benefit from ChatGPT is scraping data from website user interfaces (UIs), e.g., scraping contact information from a directory, scraping rental listings. People have started exploring how LLMs like ChatGPT can power interactions with website UIs [6, 9, 10], but it is not yet well understood their capabilities and how well they can match user intent.

We informally experimented with ChatGPT on a variety of websites to explore its potential promise and pitfalls for web scraping. From our observations, we believe that ChatGPT can often produce the right results when tasked with scraping raw data, but sometimes will mistakenly exclude certain data or "hallucinate" answers. For second order operations that involve reasoning or math over the raw data, ChatGPT will most often be wrong. As a result, we postulate that future human-LLM web scraping systems need to provide users effective tools for validating LLM responses.

2 PRIOR WORK

To programmatically scrape data from web UIs, traditionally people write web scraping scripts using libraries like Selenium [8], Cypress [3], or Puppeteer [7] to mimic users' interactions with UI elements. However, this requires a certain degree of effort and programming expertise, and it can even still be tricky for people with programming expertise to identify correct element selection logic [14]. To make web scraping more accessible to general users, prior work has explored automated ways of creating web scrapers. Program synthesis and HCI researchers have designed programming-by-demonstration (PBD) approaches [12, 19], where the user interacts with website pages to demonstrate examples of the data they want to scrape, and then the system infers the rest of the data to scrape [11, 13, 22], often by inferring patterns in the DOM [4]. Related work has leveraged similar PBD approaches for automating tasks on smartphones [16–18, 20] and answering questions about website content [15]. Web browser extension stores also showcase numerous open source web scraping extensions, which often take a demonstration-based approach or proactively try to search for data. Such research systems and browser extensions can be powerful but each have their own limitations, and typically claim to support only a specific scope of scraping tasks well. Another challenge with these tools is their lack of availability to the public or low discoverability.

Web scraping may be a good candidate task for LLMs, since one way of looking at web scraping is as a text extraction task, and LLMs are good at generating text. LLMs such as ChatGPT have the potential to address the above two challenges with web scraping specific tools: 1) ChatGPT is currently (at the time of this writing) readily available to the public and widely known, and 2) given their generative nature may be able to produce a web scraping answer for a broader set of scenarios, albeit possibly at a lower quality than their domain-specific counterparts.

People have started exploring using language models for interpreting and interacting with website UIs. SemanticOn [21] leverages natural language input and language models to enable users to specify semantic conditions to refine web scraping programs

CHI '23 Workshop, April 23, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

generated by WebRobot [13]. Recent prototypes [6, 9, 10] have leveraged GPT-3 [5] and ChatGPT for a related task – web automation. The user provides a natural language description of their task and these prototypes then programmatically interact with website UIs to complete the task. These tools are promising but it is not yet well understood their capabilities and how well they can match user intent.

3 METHODS

During the week of February 20, 2023, we informally experimented with ChatGPT (Model: Legacy (GPT-3.5)) on several websites to get a sense of its web scraping abilities and the kinds of mistakes it makes. Table 1 shows most of the websites we tried, and samples of the many prompt variations we tried. We chose some websites that were higher profile (e.g., IMDb and NBA) and whose content ChatGPT was likely trained on, and some that were lower profile (e.g., a local tennis tournament, a school district, a local restaurant chain).

We usually provided ChatGPT all the information it needed in a single prompt, i.e., a *data scraping request* and the *website context*. To embed the website in the ChatGPT prompt, we tried two approaches: embedding the page's raw text (e.g., captured using the JavaScript command `document.querySelector("body").innerText`) or embedding the page's HTML (note that due to ChatGPT's token limit we could embed only a portion of the website HTML). We sometimes asked ChatGPT to regenerate its response so we could see multiple possibilities. Sometimes we also gave ChatGPT follow-up requests if there was a mistake in its response.

We also briefly experimented with giving ChatGPT HTML and asking it to *write code* for scraping.

4 POTENTIAL PROMISE

For many pages, ChatGPT often scraped the exact right data, especially when given the website's **text** rather than HTML. For example, when given the website's text and asked to scrape, it was usually able to return the correct list of NBA players and corresponding heights and countries of origin (Table 1, website #2); actor and character names (Table 1, website #1); university event talk titles, dates, speakers, and locations (Table 1, website #3); volunteer roles available (Table 1, website #4); and menu items and prices (Table 1, website #7). Occasionally it would sometimes leave out an entry (e.g., missing an actor name), and sometimes this was resolved by asking ChatGPT to regenerate its response.

Given a portion of the website's **HTML** and asked to scrape, ChatGPT sometimes returned the full/correct answer for the given HTML (e.g., NBA players, heights, countries; volunteer roles; restaurant menu items) but for others often truncated and only returned a few items, or hallucinated certain results (see pitfalls below).

When given the website's HTML and asked to **write code for scraping**, ChatGPT seemed to mostly generate correct code logic and CSS selectors [2] if the elements to be scraped have salient selectors – e.g., when we asked ChatGPT to scrape player names from the NBA website, it generated the following correct Python code and CSS classes for first and last names:

```
first_name = player_row.select_one('.RosterRow_playerFirstName__NYm50').text
last_name = player_row.select_one('.RosterRow_playerName__G28lg p:nth-of-type(2)').text
```

5 POTENTIAL PITFALLS

5.1 Asking ChatGPT to scrape and return an answer

There were a few kinds of mistakes we saw ChatGPT make. Sometimes it produced nearly the correct output data except it would be missing a small number of entries. For example, when we gave ChatGPT the text of the CODA movie page (Table 1, website #1) and asked it to show the actors listed, it returned 17 correct actors but was missing an 18th (Molly Beth Thomas). We then prompted it saying "you're missing one" and it added an 18th actor, whose name was completely hallucinated (Emely Grisanty). Sometimes asking ChatGPT to regenerate its response did result in the missing name then correctly being included.

ChatGPT seemed to make more mistakes when we gave it the page's HTML rather than text. ChatGPT would frequently prematurely end its scraping. For example, we gave ChatGPT HTML for a portion of the VL/HCC conference program (Table 1, website #6) and asked it to return a list of paper links; it returned the first four paper links in the HTML (all correct), but the HTML actually contained 8 in total. It is unclear why results often get cutoff when ChatGPT is asked to scrape from HTML.

In one especially interesting case where we gave ChatGPT the HTML for the CODA movie page, it returned 11 actor and character pairs, only 3 of which were correct. ChatGPT hallucinated some character names, some of which matched patterns in real character names (e.g., "Gina" Rossi –> not a real character, but other characters in the movie have the same last name "Rossi"; Ms. "Han" –> not a real character, but another character in the movie has the same title,"Ms." Simon). ChatGPT also included some incorrect actor names, e.g., C.J. Jones and Lauren Ridloff, who are deaf actors but not in CODA, which is a movie about a deaf family and whose cast contains deaf actors. Perhaps these hallucinations are due to embeddings ChatGPT has learned.

Even though it seems ChatGPT may be more accurate in scraping tasks when given page text rather than HTML, HTML may be needed for certain kinds of scraping and automation tasks – when desired data is embedded in the HTML and not included as regular text on the page (e.g., links, or data in widgets like dropdown menus), and when the page needs to be navigated to uncover desired data (e.g., clicking on buttons and links, typing into text fields).

Beyond just scraping raw data on the page, we sometimes also asked ChatGPT to filter those results. ChatGPT was wrong nearly every time, regardless of whether the prompt contained page text or HTML. For example, when we asked ChatGPT to show all players shorter than 6 foot 5 (Table 1, website #2), when we gave it HTML it returned only players who are *exactly* 6 foot 5, and when we gave it text it returned some players shorter than 6 foot 5 but missed several others. When we asked ChatGPT to show all menu items on the Serafina menu (Table 1, website #7) under $25, it was almost correct but missed one item. As ChatGPT users have been reporting, ChatGPT seems to make a lot of mistakes on reasoning or math tasks.

#	Website	URL	Sample Prompts
1	IMDb movie pages	E.g., CODA: https://www.imdb.com/title/tt10366460/?ref_=adv_li_tt	*Here's a website, show me a list of the actors. [Website HTML or raw text]*
2	NBA players	https://www.nba.com/players	*Here's a website, show me a list of the players and their heights. Then show me all players shorter than 6-5. [Website HTML or raw text]*
3	University events	https://cse.engin.umich.edu/events/	*Here's a website. For each event, extract its name, date, location, and speaker. [Website HTML or raw text]*
4	Volunteer page for a tennis tournament	https://www.citiopentennis.com/volunteers/	*Here's a website. Show me the different volunteer roles. [Website HTML or raw text]*
5	County school system	https://ww2.montgomeryschoolsmd.org/departments/sharedaccountability/glance/index.aspx	*Here's a website. Show me a list of the high schools. [Website HTML or raw text]*
6	Conference program	https://conf.researchr.org/program/vlhcc-2022/program-vlhcc-2022/	*Here's a website. Extract a list of the paper links. [Website HTML or raw text]*
7	Small restaurant chain menu	https://www.serafinarestaurant.com/upper-west-77th-menus/#upper-west-pasta	*Here's a website. Show me the items and their prices. Then show me items under $25. [Website HTML or raw text]*

Table 1: Some of the websites and prompts we tried.

5.2 Asking ChatGPT to write code for scraping

This needs to be explored further, but we observed ChatGPT make mistakes in the code it generated when the items to be scraped did not have salient CSS classes or attributes. For example, the Citi Open tennis tournament (Table 1, website #4) website has minimal styling, so ChatGPT did not have meaningful CSS classes to latch on to. Instead, ChatGPT used a very generic selector `p` and then chained `find` and `find_next_sibling` method calls to parse the DOM to find desired elements; however, this then resulted in a runtime error when an intermediate `find_next_sibling` call evaluated to `None`, because the ChatGPT-generated code did not protect against edge cases. This is just an example of the kinds of challenges ChatGPT may have in generating web scraping code. It seems there are general LLM code generation challenges. It should also be further explored how accurate and robust CSS selectors and element selection logic LLMs can generate.

6 FUTURE OF HUMAN-AI WEB SCRAPING SYSTEMS

ChatGPT has some promise right out of the box for web scraping, but it has many dangers as well. Given how often ChatGPT can be right or close to right for web scraping, people may not check its results closely and instead just trust it. Any future LLM-powered web scraping tool needs to provide users tools to check its work, e.g., by contextualizing results visually with the web page, by providing links to sources. Future work may also explore ways to leverage LLMs more effectively for web scraping, perhaps by combining demonstration, web UI specific models, and LLM-based approaches.

7 ACKNOWLEDGMENTS

This work is supported by NSF Award 2007857.

REFERENCES

[1] [n. d.]. ChatGPT: Optimizing Language Models for Dialogue. https://openai.com/blog/chatgpt/. Accessed: 2023-02-23.
[2] [n. d.]. CSS selectors. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors. Accessed: 2021-03-19.
[3] [n. d.]. Cypress. https://www.cypress.io/. Accessed: 2021-03-19.
[4] [n. d.]. Document Object Model (DOM). https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/. Accessed: 2021-06-29.
[5] [n. d.]. GPT-3 Powers the Next Generation of Apps. https://openai.com/blog/gpt-3-apps/. Accessed: 2023-02-23.
[6] [n. d.]. natbot. https://github.com/nat/natbot. Accessed: 2023-02-23.
[7] [n. d.]. Puppeteer. https://pptr.dev/. Accessed: 2020-09-18.
[8] [n. d.]. Selenium. https://www.selenium.dev/. Accessed: 2020-09-11.
[9] [n. d.]. weblm. https://github.com/aidangomez/weblm. Accessed: 2023-02-23.
[10] [n. d.]. The world's first AI Web Co-Pilot powered by ChatGPT. https://multion.ai/. Accessed: 2023-02-23.
[11] 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.
[12] Allen Cypher and Daniel Conrad Halbert. 1993. *Watch what I do: programming by demonstration.* MIT press.

[13] 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 ACM SIGPLAN Conference on Programming Language Design and Implementation*.

[14] 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.

[15] Rebecca Krosnick and Steve Oney. 2022. ParamMacros: Creating UI Automation Leveraging End-User Natural Language Parameterization. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–10.

[16] Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. 2017. SUGILITE: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 6038–6049.

[17] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M Mitchell, and Brad A Myers. 2018. Appinite: A multi-modal interface for specifying data descriptions in programming by demonstration using natural language instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 105–114.

[18] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M Mitchell, and Brad A Myers. 2019. Pumice: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 577–589.

[19] Henry Lieberman. 2001. *Your wish is my command: Programming by example*. Morgan Kaufmann.

[20] Lihang Pan, Chun Yu, JiaHui Li, Tian Huang, Xiaojun Bi, and Yuanchun Shi. 2022. Automatically Generating and Improving Voice Command Interface from Operation Sequences on Smartphones. In *CHI Conference on Human Factors in Computing Systems*. 1–21.

[21] 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.

[22] Mohammad Raza and Sumit Gulwani. 2020. Web data extraction using hybrid program synthesis: A combination of top-down and bottom-up inference. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1967–1978.