Using LLMs to Customize the UI of Webpages

Amanda Li
xal@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Jason Wu
jsonwu@cmu.edu
HCI Institute, Carnegie Mellon University
Pittsburgh, PA, USA

Jeffrey Bigham
jbigham@cmu.edu
HCI Institute, Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

LLMs have capabilities to understand natural language and code, which makes them a great candidate for user-driven customization of webpages. A process that focuses on natural language can be useful for those who are less technologically literate. In this paper, we explore the potential of using LLMs to modify webpages, and what kinds of opportunities and challenges that come with it. We observe that specific prompts referring to color or targeted components can succeed, vague requests and any complex website tend to perform poorly.

KEYWORDS

natural language interfaces, large language models, human-computer interactions

ACM Reference Format:
Amanda Li, Jason Wu, and Jeffrey Bigham. 2023. Using LLMs to Customize the UI of Webpages. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23 Adjunct), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3586182.3616671

1 INTRODUCTION

The growing popularity of large language models (LLMs) allows numerous applications to take advantage of its natural language understanding and generation, especially since users can automate tasks or discover niche knowledge in an efficient manner. Its utility in correcting grammar and debugging makes it evident that we should explore combining both use cases into one task. Previous systems still require some technical expertise and can only apply changes in a localized area [10]. Prior studies in accessibility assess that less technologically literate benefit from more general systems [18]. It would be beneficial for LLMs to not only interpret the natural language command, but also find all related components of a webpage and apply the changes directly, without the need of additional adjustments.

In this paper, we explored the use of off-the-shelf LLMs to modify entire webpages based on natural language directives. Based on prior work, we identified a set of features that users might invoke through natural language for customization. We conducted an early exploration of prompts that could automatically make changes to web user interfaces (UIs) based on these prompts. From our experiments, we provide observations that can inform the design of future systems that support natural language adaptation of UIs.

2 RELATED WORK

Not all webpages are designed well, and there may be times users would want to customize the interface to fit their own needs. Previously, in order to edit the UI of the webpage, people would need to tinker with browser settings, or go into Developer Tools and methodically understand the source code, but these approaches require a higher level of technical literacy. There are some commercial tools that can help build websites with simple UI and AI-assisted technology, but they focus on the creation, rather than the modification, of websites [3]. Other tools, like CrowdUI and CrowdAdapt, focus on structural component layout, and not aesthetics of individual components. [14, 15]. We want to further this development and actualize improvement to specific components on a webpage.

The recent rise of LLMs motivates the exploration of its effectiveness in this area of editing webpage UI. There are some initial tools developed that manage the natural language aspect, such as editing word documents, emails, and the like [1, 5]. This does not utilize LLMs to its full potential, which also includes summarizing and manipulating code [6, 9]. Models are trained on different programming languages, including HTML and Javascript, such that it could be used for specific webscraping tasks [7, 8, 11]. Advancements in prompt engineering for code-related tasks invite opportunities to transfer current few-shot learning and prompting techniques to programming [12, 13, 17]. Thus, it makes sense to combine both natural language and code to help users customize user interfaces.

Some prior work have started using LLMs to modify UIs. A mobile version that explores how LLMs can interpret a mobile UI feeds an HTML hierarchy into the model, implying that webpage source code could be promising to dissect [16]. Stylette allows users to edit webpages through LLMs, but only allows users to modify through selecting one UI component and tinkering with other dropdowns and selections, which is inconvenient if they want to apply the same edit to other similar fields [10]. While this can be helpful for certain customization, we aim to simplify the whole process through solely natural language. This way, the user can make specific or general requests and the result would show up automatically.

3 METHODOLOGY

We used LLMs to interpret the natural language commands and webpage source code together and output modified code. It ran on some sample websites with prompts to analyze how specific or vague a request can be, and what kind of visual anomalies it could result in.

UIST '23 Adjunct, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0096-5/23/10.
https://doi.org/10.1145/3586182.3616671

Because of the nature of the models we used, we had to break up the source code, feed it individually into the model along with the user prompt, and piece the resulting chunks back together. Multiple passes through the LLM decreased the performance quality. However, the model we used, `Legacy (text-davinci-003)`, has a token limit of 4,097, while a webpage can be tens of thousands, if not hundreds of thousands, tokens long [4], so this process was necessary.

There were 3 main models, or endpoints, that we tested: Text Completion (`Legacy (text-davinci=003)`), Code Completion (`Legacy (code-davinci-002)`), and Code Edit (`Legacy (code-davinci-edit-001)`). Information garnered from the given descriptions, plus our own observations, has guided our preference towards using one of the models over the others. We

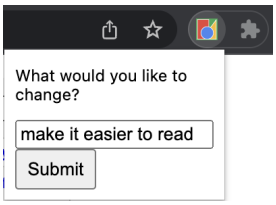

Figure 1: Simple Chrome extension UI

envisioned a system that is easy to surface and direct, as shown in Figure 1. Unlike prior work, after submitting the prompt, there would be no other manipulations that the user would need to do. Ideally, all modifications are already taken care of by the model.

4 RESULTS

4.1 Opportunities

We observed that the model performed best with simple requests related to color and size. It is reasonable that users personalize a webpage with poor text contrast by requesting "make the text purple." It can successfully manipulate websites to change all of their written text to the color purple, with an occasional missed component that was bulleted or in the footer.

It is also possible for changes to target a specific area of a webpage - for example, the header. Changing a specific component on a website is not possible with current browsers, unless the user goes into the source code. With the ease of natural language, we see customizing certain components is still effective. For instance, we can ask "make the heading larger" and it'll only increase the size of the tabs in the navigation bar.

Another use case is for accessibility reasons. In the case where someone is red-green colorblind, it would be helpful to transform the website to fit their needs. To simulate something similar, we tried applying a grayscale filter, a setting also available on iOS, to see how a website would respond. Sending a simple prompt "activate grayscale" would change not only the text and hyperlinks to grayscale, but also the images on the website too. This is promising for future endeavors to explore this topic for accessibility-specific uses.

4.2 Challenges

First, we explored ways to send a combined user prompt and piece of HTML to edit. Three main models that seemed to fit our use case, as described in the previous section. The Text Completion is a GPT-3.5 model that is stated as capable of doing any language task, but is mostly suited for inserting text. However, we see that it is capable of handling code as well. The Code Completion model uses few-shot learning examples and given code to complete the rest of the program. However, because it can only refer to provided code, it will end up making up variable names or make assumptions about what exists, especially if it needs to manipulate components not included in the chunk. For instance, if asked to change the font size of the headers, it might give something like the following:

```
var heading = document.getElementsByTagName('h1')[0];
heading.style.fontSize = '2em';
```

The endpoint is better suited for prompts that can be handled by JavaScript script changes and writing complete functions. Finally, the Code Edit endpoint modifies text or code that already exists, and is best suited for "refactoring, adding documentation, translating between programming languages, and changing coding style" [2]. It was quickly outdated (as of Spring 2023), so we referred to a newer model instead.

We also observe that the model may also output some nonsensical components if it does not know what to do. To mitigate it, we made sure to preprocess script tags so that it doesn't get modified. However, with complex components, the model may create a visually-similar component but without its original functionality. For instance, on the UIST 2023 website, we observe that the original dropdown menu titled "More" has transformed into a similar component named "Dropdown Menu", but without the original functionality.

It also doesn't work well with vague requests. For example, prompts like "make it easier to read" or "make it more modern" on the UIST website results in little to no visual changes. However, we can also see the opposite effect on a more complicated website like that of a course home page. Nonsensical components are created, the layout is cluttered, and the original, desired effect was not achieved. We could mitigate this by inserting a process that takes vague directives and converts them into actionable commands that the model can better interpret. For instance, "make it easier to read" converts to "increase the font size of all elements", which the current LLM can transfer to certain CSS styles.

5 FUTURE WORK

Using LLMs for customizing web interfaces has great potential. We see that it can reasonably handle CSS-styled edits and even prompts targeting specific components on simple websites. However, pre- and post-processing is critical to enhance its capabilities, especially for complex websites with many layers in its HTML hierarchy.

Further work can focus on how to best interpret vague input prompts and map them to formulations that are actionable commands. We also note that as the quality of the LLMs improve, we can expect that the success rate of vague requests would increase. Hopefully, a refined system can bring customization to the forefront for users navigating difficult webpages, with the ease of using natural language.

REFERENCES

[1] [n. d.]. *Design and Code with ChatGPT and Midjourney.* Retrieved July 5, 2023 from https://designcode.io/gpt4-apps

[2] [n. d.]. New GPT-3 capabilities: Edit & insert. https://openai.com/blog/gpt-3-edit-insert. Accessed: 2023-05-07.

[3] [n. d.]. *Pagecloud.* Retrieved July 17, 2023 from https://www.pagecloud.com/

[4] 2023. GPT 3.5. https://platform.openai.com/docs/models/gpt-3-5. Accessed: 2023-07-16.

[5] 2023. *MaxAI.me: Use ChatGPT AI Anywhere Online.* MaxAI.me: UseChatGPTAIAnywhereOnline

[6] Toufique Ahmed and Premkumar Devanbu. 2022. Few-shot training LLMs for project-specific code-summarization. arXiv:2207.04237 [cs.SE]

[7] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs.CL]

[8] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. 2023. Understanding HTML with Large Language Models. arXiv:2210.03945 [cs.LG]

[9] Walid Hariri. 2023. Unlocking the Potential of ChatGPT: A Comprehensive Exploration of its Applications, Advantages, Limitations, and Future Directions in Natural Language Processing. arXiv:2304.02017 [cs.CL]

[10] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 5, 17 pages. https://doi.org/10.1145/3491102.3501931

[11] Rebecca Krosnick and Steve Oney. 2023. Promises and Pitfalls of Using LLMs for Scraping Web UIs. Workshop. (April 2023). https://doi.org/10.1145/nnnnnnn.nnnnnnn

[12] Chao Liu, Xuanlin Bao, Hongyu Zhang, Neng Zhang, Haibo Hu, Xiaohong Zhang, and Meng Yan. 2023. Improving ChatGPT Prompt for Code Generation. arXiv:2305.08360 [cs.SE]

[13] Stephen MacNeil, Andrew Tran, Joanne Kim, Ziheng Huang, Seth Bernstein, and Dan Mogil. 2023. Prompt Middleware: Mapping Prompts for Large Language Models to UI Affordances. arXiv:2307.01142 [cs.HC]

[14] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. 2013. CrowdAdapt: Enabling Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (London, United Kingdom) *(EICS '13)*. Association for Computing Machinery, New York, NY, USA, 23–32. https://doi.org/10.1145/2494603.2480304

[15] Jonas Oppenlaender, Thanassis Tiropanis, and Simo Hosio. 2020. CrowdUI: Supporting Web Design with the Crowd. *Proc. ACM Hum.-Comput. Interact.* 4, EICS, Article 76 (jun 2020), 28 pages. https://doi.org/10.1145/3394978

[16] 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* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 432, 17 pages. https://doi.org/10.1145/3544548.3580895

[17] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs.SE]

[18] Jason Wu, Gabriel Reyes, Sam C. White, Xiaoyi Zhang, and Jeffrey P. Bigham. 2021. When Can Accessibility Help? An Exploration of Accessibility Feature Recommendation on Mobile Devices. In *Proceedings of the 18th International Web for All Conference* (Ljubljana, Slovenia) *(W4A '21)*. Association for Computing Machinery, New York, NY, USA, Article 21, 12 pages. https://doi.org/10.1145/3430263.3452434