WebPerceptor: An Open Source Chromium Plugin for Real-Time LLM-Based In-Line, In-Browser Re-Writing of Website Content

Joseph O'Hagan
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
joseph.ohagan@glasgow.ac.uk

Graham Wilson
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
graham.wilson@glasgow.ac.uk

Mark McGill
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
mark.mcgill@glasgow.ac.uk

Figure 1: Example output showing part of the CHI webpage rewritten by the *WebPerceptor*. The CHI 2026 welcome message (left) the unedited website, (centre) re-written as a pirate, and (right) re-written as Donald Trump angry about not going to CHI.

Abstract

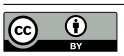
We introduce the "AI Mediated Web", an automatic, personalised, in-line, real-time remixing of web content by an LLM based on given prompts and presented in-browser such that the user perceives the end result as a published web page. We present *WebPerceptor* an open source, client-side Chromium plugin which, for any webpage, identifies text content, relays this to a local or cloud-based LLM with a user-defined prompt and then modifies the existing text (to replace, append, or remove it) based on the LLM response. We discuss the potential benefits and harms of an AI mediated web where the idea of a shared, canonical, objective web will no longer hold as AI becomes the mediator of what we perceive when we browse online.

CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI)**; **Web-based interaction**.

Keywords

AI, Web Browsing, AI mediation, Browsers, Large Language Models, Toolkit, Open Source



CHI EA '26, Barcelona, Spain

ACM ISBN 979-8-4007-2281-3/26/04
https://doi.org/10.1145/3772363.3798730

ACM Reference Format:
Joseph O'Hagan, Graham Wilson, and Mark McGill. 2026. WebPerceptor: An Open Source Chromium Plugin for Real-Time LLM-Based In-Line, In-Browser Re-Writing of Website Content. In *Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems (CHI EA '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3772363.3798730

1 Introduction

Large language models (LLMs) have been extensively used to generate or modify text content to a variety of ends, from altering the style of writing [12], to adapting textual complexity to support comprehension [8], to enhancing the persuasiveness of messaging [4], etc. Yet when it comes to the web, this capacity for rewriting or remixing textual content has largely been utilized by the authors of published content, with LLMs being an authorial/editorial tool used during content creation. For users/readers, we are seeing increasing integration of AI (LLMs/VLMS) into the web browsing experience, for example in web searches [5] (e.g. Google's "AI Mode" search), or as companion Chromium-based plugins or browser spin-offs (e.g. ChatGPT Atlas as an agentic "super-assistant"). These give local and remote AI access to the users browsing context and history, supporting tasks such as content summarization, translation, and writing assistance [3]. But crucially, the users of existing web browser AI integrations perceive these AI outputs **separately** and apart from the publishing entity/website e.g. as a overlay sidebar, or an obvious modification enacted post-render driven by user input and actions. Underlying this, there still exists one authoritative version of said web content that is perceived by all users, and whose content remains initially unaltered by web browsers.

But consider instead an alternative approach to the integration of AI into the web browsing experience. An approach where, when a user opens a web page, the pertinent article text from the Document Object Model (DOM) is automatically identified, batch-fed to an LLM alongside a specified prompt, and the resulting output reinserted back into the DOM, either replacing or appending the original article text. This process, in effect, employs LLM-driven text generation and in-browser DOM manipulation to produce a personalized, in-line, real-time remixing of web content, presented seamlessly within the browser such that, from the user's perspective, the mediated version appears as the published web page itself.

Such an *"AI Mediated Web"* could provide many benefits including, e.g. **comprehension:** adapting content to reading level [8], or making personalized accommodations to create a more generally cognitively accessible web; **engagement:** altering content in terms of tone, sentiment, emotional resonance, etc [12]; **factuality:** providing in-line fact checking [21], identification of bias [7], or increased representation such as incorporating different points of view where limited voices are heard; **safety:** censoring triggers [23], or otherwise adapting content to be safer for vulnerable groups (e.g. a parent giving a prompt to ensure age appropriate, child-friendly browsing, filtering inappropriate or harmful wording or content) [16]; **search:** highlighting pertinent information related to the overall search query, or otherwise remove, diminish, filter or sort less relevant information; and more.

However, said tool would also more fundamentally give users, and any other stakeholders that influence the AI input prompts or control the browser experience (e.g. parents, employers, governments, developers, etc) the capacity to mediate the web as perceived by the user. This could include, e.g. **bias:** rewriting news to remove or add perceived bias [14]; **censorship:** distressing content (e.g. conflicts, famine etc.) is rewritten to be more positive [15] or hidden from view entirely; **information disorder:** a political party provides a prompt or plugin to help "fact check" opposition policies [2] that in-turn acts to control the wording/framing/content perceived by supports, exacerbating echo chambers; re-writing material as an attack on the author or subjects character or credibility, etc; **extremism:** amplifying narratives, e.g. widening the spread of false narratives, extremist views, and political viewpoints [22] e.g. dehumanizing groups or demographics when discussed by removing references or rewording sympathetic views.

This approach represents a shift from the static publication model of the web to one of dynamic mediation, where web pages cease to be fixed documents that are authored once and distributed identically to all readers. Instead, web content is made mutable, co-authored in real time by the model and prompts, offering unprecedented personalisation whilst destabilising the notion of a common, shared web experience.

In this paper, we propose this idea of an "AI Mediated Web", and discuss server, client-side, and hybrid architectures towards serving LLM-remixed web content to end users, and present an open source Chromium plugin ***WebPerceptor*** [1]. ***WebPerceptor*** [11] is a user controlled, client-side LLM-based web mediator capable of, for any website, automatically extracting target textual content during rendering, processing this content via a local or cloud-based LLM, and then seamlessly replacing that content in the DOM. We present an initial benchmark testing of *WebPerceptor* demonstrating its effectiveness to remix webpages within Google's recommend display times for webpage content loads.

[1] https://github.com/theartofhci/WebPerceptor

2 Proposed Architecture(s) for LLM-Based Web Mediation

At a high level, we outline server-side and client-side approaches to enable LLM-based web mediation. Throughout, we refer to textual content being modified (be it a header, post, long-form article, etc) as the *article* for shorthand.

2.1 Server-side

If we consider an article being published, at the point of committing the authoritative, canonical article we envision a server-side LLM could execute a series of prompts to generate and cache multiple rewritten variants ready to serve to specific users or user groups, each aligned with a specific ideological, demographic, or stylistic profile. For example, a news article [6] might be rewritten with both moderate or extreme left/right-wing variants to be served based on the tracked behavioural or declared profile of the visitor to maximise engagement. This approach centralises control at the point of publication. This has the benefit of minimizing the amount of LLM computation required, whilst retaining a degree of commonality of experience in what is served (i.e. people within a certain grouping would experience the same content), as a form of invisible group-wise filter bubble [13]. This is arguably an evolution of algorithmic approaches to content selection and serving [1, 17], where the content itself is adapted rather than the selection of the content.

2.2 Client-side

If we consider the point at which any article is rendered in the user's browser, on page load a client-side script or extension can identify and extract pertinent article text from the DOM, hide it from rendering, and batch feed this text to a local or cloud-based LLM alongside a specified prompt. The resulting output can then be inserted back into the DOM, replacing or appending the original article content. This approach gives the client control over article mediation at the cost of greater computational requirements as each web page served, for each user, requires processing by an LLM. Article mediation can be context-dependent, e.g., rewriting with different prompts depending on the domain, the identified author of an article, the identified user interacting with the browser, etc. When run using local LLMs, this architecture provides privacy-preserving [20] web mediation, aligned with on-device AI development efforts. However, user autonomy and control over when and how mediation occurs can vary significantly, ranging from being entirely user controlled to entirely provider controlled.

In a user controlled approach, users can specify prompts provided to the LLM and when/what articles should be modified (e.g. for all websites, select websites, paragraph text only, etc). This provides each user with complete control over when and how mediation occurs, enabling a personalised re-framing of their experience browsing the web.

In a provider controlled approach, external providers (i.e. the creator/distributor of a *WebPerceptor*-like plugin) may specify the prompts and content to target, information which may be visible to or hidden from the user. In such instances, computation is decentralised but the interpretation of content is centralised and controlled by the provider. This, unlike a user controlled approach, allows for the distribution of branded/specified web mediators which can retain a degree of commonality of experience in what is served (i.e. users of the same *WebPerceptor*-like system would experience the same content). A primitive version of this approach can already be seen through *Grokipedia* [2] which initially, using prompts hidden from users, presented xAI's canonical edit and rewrite of *Wikipedia* [24] using the Grok LLM [3]. And while *Grokipedia's* stated goals are to *"create an open source, comprehensive collection of all knowledge"* [9] and report only *"the truth, the whole truth and nothing but the truth"* [10], concerns are already being raised with articles being rewritten to introduce bias or be more aligned with the viewpoint of xAI and the website's creators [18].

2.3 Hybrid

In addition to fully client-side or server-side architectures, hybrid approaches may emerge. Consider, for example, a company, organisation, individual, etc, that distributes a client-side instance of a *WebPerceptor* for mediating an individual's experience browsing the web based on the provider's specified prompts and content targets. This *WebPerceptor* instance operates locally in the user's browser but according to a pre-determined interpretive frame, i.e. the distributor sets fixed prompts for the LLM the user is unable to edit, meaning all users experience a shared, canonical, mediated version of articles.

To minimise computation costs, the system could employ a caching mechanism where the first user to encounter a new article triggers the mediation process, wherein the relevant textual content is extracted from the DOM, batch-fed to a local or remote LLM using the distributor's preset prompt, and the generated text then inserted back into the article. This rewritten article is then cached by the distributor and subsequently served to all other users of the system who access the article. Through this the network's collective activity populates the provider's cache, creating a distributed, shared corpus of mediated articles. This approach also minimises redundant LLM inference and preserves a degree of commonality of experience across users. And such an approach could easily adopt tiered access models for users. For example, free users might rely exclusively on cached articles or local computation (if possible), while paid users could access on-demand mediation through a cloud LLM.

3 WebPerceptor: System Design and Features

WebPerceptor [11] is an open source Chromium browser extension we have built to enable AI mediated web browsing. It is a client-side, user controlled LLM-based web mediator, meaning the user has full control over how and when webpage content is modified.

WebPerceptor automatically identifies text elements on any webpage, sends each text element to an LLM (running either locally or on the cloud) with a user defined prompt to modify the text, then replaces the identified text element with the model's output. This replacement text could be an exact copy of the input text, a full rewrite of the input text (e.g. to be written in a different tone, language, style, reading level, etc), or an appended version of the text (e.g. appending commentary highlighting detected inaccuracies or bias in a text, etc). The resulting effect is, when browsing, webpages are transformed seamlessly in real-time based on the specification of the user. Examples of modified content are shown in Figure 1.

3.1 Implementation

WebPerceptor uses a two-part, content/background architecture. The content script interacts with the webpage, i.e. identifying text elements to process, watching for new content as the page updates, sending text elements to the background service for processing, and replacing text elements with the background service's response. The background script runs separately from any webpage, i.e. receiving text elements to process from the content script, constructing the query based on the user settings and text element to send to the LLM, and returning the LLM's response to the content script. *WebPerceptor* uses a generic/site-specific architecture consisting of a generic content/background module which works on any webpage, complemented by site-specific modules that override or extend the default behaviour for particular domains. This allows *WebPerceptor* to work on any webpage while enabling specialised processing on any domain for which custom processing behaviours are created.

3.2 Functionality

3.2.1 ***Text Modification:*** *WebPerceptor* supports two modes of text modification: (1) rewriting, and (2) appending. In the rewriting mode, the original article text is replaced entirely with a version generated by the LLM. In the appending mode, the original article text remains unchanged and the LLM's output is added after it. Both rewriting and appending by default (i.e. using the generic module) are transparent in their text modification. During processing, targeted text elements are reduced in opacity but not fully, creating a faded text effect. Once the model's output is applied the opacity is restored to full. This effect provides users with a real-time visual indication of which parts are actively being transformed, although can be overridden, i.e. to make text fully opaque and hidden until the model's output is applied.

3.2.2 ***Online/Offline Models:*** *WebPerceptor* supports offline and online LLM models. In its offline mode, query requests are sent to a locally hosted LLM (i.e. an Ollama served model with a lightweight Node.js API wrapper). This ensures privacy during use, is free to use, and allows the use of personalised, custom models, albeit with some performance costs. In its online mode, query requests are sent to a cloud-based LLM (e.g. OpenAI's API, Grok's API, etc) along with the user's API key and select model. This provides access to state of the art models, removes the need for a user's hardware to be capable of running local models, and can provide the most performant experience for users.

3.2.3 ***Targeted Modification:*** *WebPerceptor* includes several features to enable targeted and controlled modification of web content, allowing users to specify: which webpages are processed, which

[2] https://grokipedia.com
[3] https://x.ai/grok

BBC News Sample Articles (Full Page Rewrite)					
Article Metadata		Cloud-based Processing Time		Local Processing Time	
Article	No. of Text Elements	First 5 Paragraphs	All of Page	First 5 Paragraphs	All of Page
Plan for new Skye to South Uist sub-sea cable progresses	55	1.43s	1.87s	5.04s	19.67s
Hoy breaks leg in 'worst' crash of his life	66	1.53s	2.19s	4.30s	24.39s
AI has entered the classroom - but is it the solution...	92	1.66s	2.26s	4.65s	37.19s

Table 1: Mean time taken to render sample news articles from https://www.bbc.co.uk/news.

types of text content are processed, and to define webpage specific behaviours. *WebPerceptor* has a flexible page filter system which allows users, if desired, to specify which domains/webpages should be processed. This allows users to specify specific URLs or URL patterns for inclusion or exclusion from processing, providing a more controlled web mediation experience. *WebPerceptor's* content filtering system allows users to specify which HTML tag types (e.g. navigation bars, paragraphs, headers, tables, buttons, etc) should be included/excluded from processing. During DOM traversal, any text contained within specified tags (or their descendants) is automatically processed/skipped depending on the set configuration. *WebPerceptor's* architecture allows for the creation of site-specific modules. These modules override the generic modification process, allowing for a pipeline built for the structure and behaviour of the target website. E.g., custom logic can be to target site-specific features/components, content-filtering presets can be established, a site-specific text element identification can be implemented, etc.

3.2.4 ***Developer Options:*** There are two HTML banners which can be enabled/disabled. There is a *"content modification warning"* banner to warn users when content on the current webpage has been modified by *WebPerceptor*. There is also a *"metadata"* banner summarising the total number of text elements detected on the current page and the time taken to process all of the text elements and the visible text elements in the viewport on page load.

4 Benchmarking

We performed an initial set of benchmarking tests to capture the performance of *WebPerceptor* modifying both pages of varying sizes and example use cases. All benchmark testing used the following prompt for rewriting and replacing text: *"You MUST generate text between (originalLength - 5) and (originalLength + 5) characters long. Preserve all HTML tags exactly as in the input. Do not include titles or summary notes. Do not include any obvious AI generated text (e.g. "Okay, here's..."). Only rewrite the inner text. Rewrite the following text at an 8 year old reading level."* Note: *originalLength* was calculated and inserted automatically for each input text element.

A MacBook Air (2025, M4 chip, 32GB RAM) was used for benchmarking. For the cloud-based model, we used OpenAI's *gpt-3.5-turbo*, to capture exemplary performant processing times due to its widespread adoption and well-documented inference speeds. For the local model, we used Ollama to serve *gemma3:1b*, to capture exemplary local performance of a medium sized model, representing the trade-off between performance and resource requirements.

4.0.1 ***Reading a News Website:*** Table 1 summarises the mean times taken to rewrite sample news articles. The cloud-based model was performant, rendering the full page in under 2.5 seconds, a time that is within Google's recommended time for a *Largest Contentful Paint* [19]) and full web page content loads (i.e. under 3 seconds). The local model performed worse, e.g. requiring approximately 20 seconds to fully render the shortest sample article. However, the time taken to render the first five paragraphs (those within the initial viewport when the user opens the page) was more performant, i.e. approximately 5 seconds or less, providing a functional performance for loading content for a user to engage with. Targeting only core text of the page (i.e. paragraph tags) can significantly decrease the time taken for rendering the full page, with our shortest length article being completed in 4.13 seconds (15.54 seconds faster), our medium length article in 9.53 seconds (14.86 seconds faster) and our longest in 26.69 seconds (10.5 seconds faster). These are good, functional times, and performance can further be improved by using more powerful hardware or optimising model selection. For example, using *qwen2.5:0.5b* instead of *gemma3:1b* as the local model reduces the time taken to complete a targeted rewrite of our short length article to 1.81 seconds.

4.0.2 ***Reading X.com:*** Webpages often utilise infinite scrolls with lazy-loaded content in a single-page application, e.g. *x.com*'s content is delivered in this way, as are *Reddit* comments which dynamically loads more content as users scroll. *WebPerceptor* is capable of dynamically detecting such content as it is loaded, identifying and remixing blocks of newly loaded text elements on the page. We measured the time taken to render rewritten dynamic blocks of text elements while scrolling on *x.com/elonmusk*. We captured 25 measurements. The mean render time for the cloud-based model was 1.47 seconds and for the local model was 2.24 seconds. Both were performant and capable of rendering the fully rewritten content within Google's recommended content paint times [19].

5 Conclusion

We introduce the "AI Mediated Web" and discuss server-side and client-side approaches to enable in-browser, personalised, real-time remixing of web pages. Alongside this, we present *WebPerceptor*, an open source Chromium browser extension we have built to create an LLM-based web mediated browsing experience. So far we have only performed formative benchmarking of performance and do not comment on the perceived capability in enacting any use cases we have speculated on. But in our on-going testing, many of the use cases mooted earlier appear eminently feasible and many of the potential benefits and harms are reflected in qualitative feedback captured during pilot testing, e.g. expressing shock *"I can't actually believe what I am seeing"* and fear *"this feels like it should be illegal... you've broken the internet"* but envisioning adoption *"I'd easily use this every day"* and good *"I'd be much more comfortable with my child being online with this"* too. Therefore, as we continue our research, we offer *WebPerceptor* for use by others to explore the

concept of, and generate discussion around, an AI mediated web, to foster new collaboration and research opportunities, and to identify future capabilities to develop to support research.

Acknowledgments

Funded by UK Research and Innovation (UKRI) under the UK's Horizon Europe funding guarantee [EP/Z000068/1] (AUGSOC).

References

[1] Engin Bozdag. 2013. Bias in algorithmic filtering and personalization. *Ethics and information technology* 15, 3 (2013), 209–227.
[2] Canyu Chen and Kai Shu. 2024. Combating misinformation in the age of LLMs: Opportunities and challenges. *AI Magazine* 45, 3 (2024), 354–368. doi:10.1002/aaai.12188 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aaai.12188.
[3] Xuanqi Gao, Juan Zhai, Shiqing Ma, Siyi Xie, and Chao Shen. 2025. ASSURE: Metamorphic Testing for AI-powered Browser Extensions. doi:10.48550/arXiv.2507.05307 arXiv:2507.05307 [cs].
[4] Miriam Havin, Timna Wharton Kleinman, Moran Koren, Yaniv Dover, and Ariel Goldstein. 2025. Can (A)I Change Your Mind? doi:10.48550/arXiv.2503.01844 arXiv:2503.01844 [cs] version: 1.
[5] Elisabeth Kirsten, Jost Grosse Perdekamp, Mihir Upadhyay, Krishna P. Gummadi, and Muhammad Bilal Zafar. 2025. Characterizing Web Search in The Age of Generative AI. doi:10.48550/arXiv.2510.11560 arXiv:2510.11560 [cs].
[6] Sarah Kreps, R Miles McCain, and Miles Brundage. 2022. All the news that's fit to fabricate: AI-generated text as a tool of media misinformation. *Journal of experimental political science* 9, 1 (2022), 104–117.
[7] Luyang Lin, Lingzhi Wang, Jinsong Guo, and Kam-Fai Wong. 2024. Investigating Bias in LLM-Based Bias Detection: Disparities between LLMs and Human Perception. doi:10.48550/arXiv.2403.14896 arXiv:2403.14896 [cs].
[8] Paloma Martínez, Lourdes Moreno, and Alberto Ramos. 2024. Exploring Large Language Models to generate Easy to Read content. *Frontiers in Computer Science* 6 (Oct. 2024), 1394705. doi:10.3389/fcomp.2024.1394705 arXiv:2407.20046 [cs].
[9] Elon Musk. 2025. *X Post.* https://x.com/elonmusk/status/1983125099973882120
[10] Elon Musk. 2025. *X Post.* https://x.com/elonmusk/status/1983009037156315440
[11] Joseph O'Hagan. 2026. *WebPerceptor Release v1.0.0.* doi:10.5281/zenodo.18724545
[12] William Orwig, Emma R. Edenbaum, Joshua D. Greene, and Daniel L. Schacter. 2024. The Language of Creativity: Evidence from Humans and Large Language Models. *The Journal of Creative Behavior* 58, 1 (2024), 128–136. doi:10.1002/jocb.636 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jocb.636.
[13] Eli Pariser. 2011. *The filter bubble: What the Internet is hiding from you.* Penguin.
[14] Qin Ruan, Jin Xu, Susan Leavy, Brian Mac Namee, and Ruihai Dong. 2024. Rewriting Bias: Mitigating Media Bias in News Recommender Systems through Automated Rewriting. In *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization (UMAP '24).* Association for Computing Machinery, New York, NY, USA, 67–77. doi:10.1145/3627043.3659541
[15] Lei Shu, Liangchen Luo, Jayakumar Hoskere, Yun Zhu, Yinxiao Liu, Simon Tong, Jindong Chen, and Lei Meng. 2023. RewriteLM: An Instruction-Tuned Large Language Model for Text Rewriting. doi:10.48550/arXiv.2305.15685 arXiv:2305.15685 [cs].
[16] Anastasia Smirnova, Kyu beom Chun, Wil Louis Rothman, and Siyona Sarma. 2025. Text Simplification for Children: Evaluating LLMs vis-à-vis Human Experts. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25).* Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3706599.3719889
[17] Zeynep Tufekci. 2015. Algorithmic harms beyond Facebook and Google: Emergent challenges of computational agency. *Colo. Tech. LJ* 13 (2015), 203.
[18] Gian M. Volpicelli. 2025. *Elon Musk's Grokipedia Pushes Far-Right Talking Points.* https://www.wired.com/story/elon-musk-launches-grokipedia-wikipedia-competitor/ Accessed: 2025-11-10.
[19] Philip Walton and Barry Pollard. 2019. *Largest Contentful Paint (LCP).* web.dev. https://web.dev/articles/lcp Last updated September 4, 2025.
[20] Xubin Wang, Zhiqing Tang, Jianxiong Guo, Tianhui Meng, Chenhao Wang, Tian Wang, and Weijia Jia. 2025. Empowering edge intelligence: A comprehensive survey on on-device ai models. *Comput. Surveys* 57, 9 (2025), 1–39.
[21] Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. 2024. Long-form factuality in large language models. doi:10.48550/arXiv.2403.18802 arXiv:2403.18802 [cs].
[22] Joe Whittaker, Seán Looney, Alastair Reed, and Fabio Votta. 2021. Recommender systems and the amplification of extremist content. *Internet Policy Review* 10, 2 (2021).
[23] Matti Wiegmann, Jennifer Rakete, Magdalena Wolska, Benno Stein, and Martin Potthast. 2024. If there's a Trigger Warning, then where's the Trigger? Investigating Trigger Warnings at the Passage Level. doi:10.48550/arXiv.2404.09615 arXiv:2404.09615 [cs].
[24] Wikimedia Foundation. 2025. Wikipedia: The Free Encyclopedia. https://www.wikipedia.org/. Accessed: 2025-11-10.