Web Content Summarization and Accessibility with Voice Output

Sanjana Sanga, Suhas Panuganti, Rishikesh Reddy Mamilla
Department of Computer Science
Old Dominion University
Norfolk, VA, USA

***Abstract*—Visually impaired users often struggle to browse modern websites that are overloaded with advertisements, pop-ups, multimedia widgets and dynamically loaded content. Traditional screen readers follow a strictly sequential strategy: they read every element in the Document Object Model (DOM), including menus, sidebars and promotional components, which forces users to listen to large amounts of irrelevant information before reaching the main content. This behaviour leads to information overload, slower comprehension and frustration.**

This paper presents *AI Smart Reader*, a browser-based system that combines web content extraction, Natural Language Processing (NLP), Optical Character Recognition (OCR) and Text-to-Speech (TTS) to deliver a more efficient accessibility experience. Instead of reading pages verbatim, the system segments a webpage into semantic regions, generates concise AI-powered summaries for each region and speaks them on demand when the user hovers or issues a command. OCR is used on e-commerce sites to extract text embedded in product images and promotional banners. The solution is implemented as a Manifest V3 browser extension and is designed to operate both in an online mode using OpenAI models and in an offline mode using a local summarizer. Experimental observations on news and shopping websites suggest that the proposed approach can reduce auditory clutter and support faster, more focused browsing for visually impaired users.

***Index Terms*—Accessibility, Screen Readers, Web Summarization, Optical Character Recognition, Text-to-Speech, Browser Extensions, Natural Language Processing.**

I. Introduction

The web is the primary interface for information, communication and commerce. For visually impaired users, interaction with webpages is mediated by screen readers that traverse the document tree and convert text into speech. Although screen readers adhere to accessibility standards such as ARIA roles and landmarks, they treat nearly all textual content with equal priority. On a typical page, menus, navigation links, footers, cookie banners and advertisements are read in the same linear order as the main article or product description. As pages grow longer and more dynamic, this linear presentation becomes inefficient and cognitively demanding.

At the same time, web design trends increasingly embed key information inside non-textual elements such as images, icons and dynamically rendered widgets. For example, discount percentages, delivery dates and deal labels on shopping sites are often part of images. Standard screen readers either skip such information or provide only limited alternative text, leaving users with an incomplete understanding of the page.

Recent advances in NLP and large language models (LLMs) make it possible to perform higher-level reasoning on raw text, including summarization and semantic filtering. By integrating such models directly into the browsing experience, it becomes feasible to move from "reading everything" to "reading what matters". This paper explores that idea through the design and implementation of AI Smart Reader, a browser extension that summarises and speaks the most relevant parts of a webpage.

The contributions of this work are threefold:

- We present a modular client-side architecture that integrates DOM analysis, OCR, LLM-based summarisation and TTS within a Manifest V3 extension.
- We describe interaction techniques—hover-based playback and voice commands—that allow non-linear exploration of summarised content.
- We provide a qualitative evaluation across multiple website categories and discuss limitations and opportunities for future research.

II. Background and Related Work

Accessibility technologies for visually impaired users have evolved from command-line screen readers to highly sophisticated tools that integrate with modern operating systems and browsers. Screen readers such as NVDA and JAWS rely heavily on the accessibility tree exposed by the browser and operating system. While these tools have excellent keyboard navigation support, they do not perform semantic summarisation of content; instead, they expose the page structure as-is.

Web summarisation has been studied in the context of search results, news aggregation and document condensation. Classical approaches rely on extractive techniques such as sentence scoring, while more recent approaches use abstractive models powered by transformers. These methods, however, are typically deployed as standalone services or offline tools and are not tightly coupled with accessibility workflows.

OCR technology has also matured significantly. Tesseract and its derivatives are able to extract text from a wide range of printed documents and web images. Within the context of accessibility, OCR has been used in mobile apps to read menus, street signs and printed documents. However, integrating OCR directly into browser extensions to augment web page summaries remains relatively unexplored.

AI Smart Reader sits at the intersection of these areas. It leverages summarisation and OCR but packages them in a

way that is directly usable inside a browser and tailored to the needs of screen-reader users.

III. Problem Statement and Design Goals

The problem addressed in this work is to reduce the auditory and cognitive load imposed by traditional screen readers on complex, content-heavy webpages, while still preserving access to the essential information.

From this problem statement, we derive the following design goals:

1) **G1: Content Prioritisation.** The system should identify and highlight content that is likely to be important (e.g., article body, product details) while deemphasising navigation and decorative elements.
2) **G2: Concise Summaries.** Instead of reading raw text, the system should provide short, natural-language summaries that capture key points.
3) **G3: Multimodal Content Support.** The system must handle text embedded in images through OCR, especially in domains like e-commerce.
4) **G4: Low Overhead.** The solution should run on the client side as a lightweight extension without requiring users to install extra software or send entire pages to remote servers.
5) **G5: Flexible Interaction.** Users should be able to consume summaries through both keyboard and mouse interactions, and optionally through voice commands.

The remainder of this paper describes how AI Smart Reader addresses these goals.

IV. System Architecture

The overall architecture of AI Smart Reader is organised into several interacting layers that operate inside the browser.

A. Content Capture Layer

The content capture layer runs as a content script inside each webpage. Its responsibilities include:

- traversing the DOM and detecting candidate regions such as paragraphs, headings and product tiles;
- grouping related nodes into coherent blocks;
- applying heuristics to filter out navigation bars, repeated footers and obvious promotional components; and
- identifying images that are likely to contain text, based on size, surrounding markup and domain-specific knowledge (for example, product thumbnails on e-commerce sites).

For each region, the layer produces a structured representation containing plain text, metadata and optional image references.

B. NLP Summarisation Layer

The NLP layer receives cleaned text from the content capture layer and generates short summaries and headings. When operating in online mode, the system queries an OpenAI model using a carefully designed prompt that instructs the model to:

- generate one short heading and one or two sentences of summary;
- keep the summary faithful to the source content; and
- avoid hallucinating details or adding extraneous context.

An offline mode is provided by a local summariser that selects key sentences and performs rule-based compression. To reduce latency and cost, summaries are cached on a per-URL basis and reused for subsequent visits.

C. OCR Engine

The OCR engine is implemented using Tesseract.js bundled with the extension. It is primarily activated on shopping domains such as Amazon, Walmart or Flipkart. When a region contains an image that matches heuristic criteria, the engine:

1) extracts the image or its URL;
2) runs OCR to obtain text; and
3) merges the OCR result with DOM text before summarisation.

This enables the system to capture visually encoded details such as discounts or shipping labels that would otherwise be missed.

D. Output and Interaction Layer

The output layer consists of an overlay user interface and a TTS module. When the user hovers over a region, the system highlights it, shows a compact panel in a fixed position and displays the generated heading and summary. The Web Speech API is used to speak the summary aloud. Keyboard shortcuts provide quick control: space toggles play/pause, escape stops speech and a combination such as Ctrl+Shift+A forces resummarisation.

E. Browser Extension Shell

All layers are packaged within a cross-browser extension shell based on Manifest V3. The manifest declares permissions (e.g., `activeTab`, host permissions for target domains), registers content scripts and loads the background service worker that coordinates caching and API calls. This packaging makes the system easy to distribute and keeps all processing on the client side, which is beneficial for privacy.

V. Implementation Details

The implementation is written primarily in JavaScript and is organised into modular files:

- `content_main.js`: performs DOM scanning and region detection;
- `overlay_ui.js`: manages highlighting, overlay panels and hover events;
- `ai_client.js`: communicates with the OpenAI API and implements the local summariser;
- `ocr_client.js`: wraps Tesseract.js and exposes OCR utilities; and
- `tts_client.js`: encapsulates text-to-speech functionality.

The summarisation client supports two configuration flags: an API key for OpenAI and an "offline-only" mode. In offline mode, calls to external services are disabled and all summaries are generated locally, which is useful for demonstrations and cost-free testing.

Caching is implemented using the browser's storage APIs. For each page, summaries are stored with a hash of the underlying text so that if the content changes, new summaries are computed automatically. The service worker is responsible for invalidating cache entries when URLs change or when the extension version is updated.

Pseudo-code for the summarisation pipeline is shown below:

```
for each region in detectedRegions:
    text = collectDOMText(region)
    if hasInterestingImage(region):
        text += runOCR(region.image)
    key = hash(text)
    if cache.contains(key):
        summary = cache[key]
    else:
        summary = summarize(text, mode)
        cache[key] = summary
    attachSummaryToRegion(region, summary)
```

This pipeline highlights the integration of DOM extraction, OCR, cache management and summarisation into a unified process.

VI. Evaluation Methodology

To assess the effectiveness of AI Smart Reader, we conducted a qualitative evaluation focusing on latency, summarisation quality, OCR accuracy and user interaction efficiency. The goal was not to produce statistically significant benchmarks but to explore how the system behaves on realistic webpages.

A. Test Websites

Three categories of websites were selected:

- **News and blogs:** long-form articles with headings, images and comments.
- **Technical documentation:** pages containing sectioned explanations, code blocks and navigation sidebars.
- **E-commerce sites:** product listing and product detail pages on Amazon and Flipkart.

For each category, between 10 and 15 representative pages were chosen.

B. Metrics

We considered four primary metrics:

- **Summarisation quality:** subjective rating of how well summaries captured the main idea of each region.
- **Latency:** time between hover and availability of speech playback, including model and OCR processing.
- **OCR utility:** proportion of summaries on shopping sites that benefited from additional OCR text.
- **Interaction efficiency:** number of navigation actions required to locate target information with and without the extension.

C. Participants

Five volunteer participants with previous experience using accessibility tools were invited to try the extension. They were asked to perform tasks such as "find the conclusion of this article", "identify the main features of this product" and "check if there is a discount or offer". Navigation was performed using keyboard commands plus the hover interaction.

VII. Results and Discussion

On article-style pages, the extension was able to segment sections accurately and generate summaries that captured key points of each paragraph. Hover-based playback allowed rapid skimming: users could move the mouse down the page and hear a two-sentence summary of each section without listening to full paragraphs. Participants reported that this interaction felt closer to "browsing headlines" than to listening to a long synthetic monologue.

On technical documentation pages, the system correctly highlighted main sections and largely ignored navigation menus and sidebars. However, summarising code-heavy regions remained challenging; in such cases the local summariser simply selected key sentences around the code blocks rather than attempting to explain the code itself.

On shopping sites, OCR successfully extracted information such as discount percentages and delivery dates from product images. When combined with DOM text, the resulting summaries were richer than those produced from DOM text alone. Participants particularly appreciated summaries that combined price, discount and delivery information in a single spoken sentence.

A qualitative comparison between GPT-based summarisation and the local summariser showed that GPT produced more fluent and context-aware summaries, especially when paragraphs contained multiple sentences. The local summariser was faster but sometimes omitted important details or produced slightly fragmented output. In offline mode, participants still considered the system helpful because it filtered out boilerplate content even when summaries were less polished.

VIII. System Evaluation and Performance Analysis

Latency measurements indicated that end-to-end summarisation time per region ranged from 0.4 to 0.7 seconds for news sites in online mode, and from 0.6 to 1.2 seconds for e-commerce sites when OCR was enabled. Caching reduced these times by more than 80% when revisiting a page or triggering the same region again.

OCR accuracy depended strongly on image quality. For high-resolution product images, Tesseract correctly recognised brand names and discount labels in most cases. Low-resolution or highly stylised images caused more errors, but even partial recognition (for example, only extracting "40% off") provided useful additional context in summaries.

Interaction efficiency improved noticeably. When using a baseline screen reader, participants needed multiple commands to navigate through landmarks and headings before reaching

the desired content. With AI Smart Reader, many tasks were completed by simply scanning summaries with the mouse and listening only to those that sounded relevant. Voice commands such as "go to reviews" or "read conclusion" further reduced navigation time, though recognition errors were occasionally observed in noisy environments.

IX. Limitations and Future Enhancements

Despite encouraging results, the system has several limitations that suggest opportunities for future work.

First, the online summarisation mode depends on a remote LLM service and requires network connectivity. While the offline summariser mitigates this, it does not match the semantic richness of GPT. A promising direction is to distil a smaller model that can run entirely in the browser using WebGPU.

Second, OCR performance degrades on low-quality or stylised images. Integrating modern deep-learning OCR architectures and domain-specific fine-tuning may increase robustness. Another possibility is to fall back to alt text when OCR confidence is low.

Third, the current system mainly targets English-language content. Extending support to multiple languages would require multilingual models, language detection and potentially translation pipelines.

Finally, the evaluation presented here is preliminary and qualitative. A full user study with visually impaired participants, using structured tasks and validated questionnaires, would be necessary to rigorously measure gains in efficiency and satisfaction.

X. Conclusion

This paper has presented AI Smart Reader, a browser extension that integrates web content extraction, NLP-based summarisation, OCR and TTS to provide an enhanced accessibility experience for visually impaired users. By operating at the level of semantic regions rather than raw DOM order, the system reduces auditory clutter and enables faster access to essential information.

The prototype demonstrates that large language models can be effectively embedded into client-side accessibility tools and combined with OCR to recover visually encoded text. Although challenges remain in terms of latency, OCR robustness and multilingual support, the results suggest that summarisation-based accessibility has strong potential. Future work will focus on quantitative evaluation with visually impaired participants, support for additional languages, adaptive summary length based on user preference and deeper integration with existing screen readers.

References

[1] W3C Web Accessibility Initiative, "Web Content Accessibility Guidelines (WCAG)," World Wide Web Consortium, 2018. [Online]. Available: https://www.w3.org/WAI/

[2] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. ICDAR*, 2007, pp. 629–633.

[3] OpenAI, "GPT-4 Technical Report," 2024. [Online]. Available: https://openai.com

[4] J. Lazar, A. Allen, J. Kleinman and C. Malarkey, "What frustrates screen reader users on the web: A study of 100 blind users," *International Journal of Human-Computer Interaction*, vol. 22, no. 3, pp. 247–269, 2007.

[5] S. Babu and A. Sharma, "A survey of modern optical character recognition techniques," *International Journal of Computer Applications*, vol. 179, no. 2, pp. 1–7, 2018.

[6] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*. Springer, 2012, pp. 43–76.

[7] C. Power, H. Petrie, and D. Swallow, "Guidelines are only half the story: Accessibility problems encountered by blind users on the web," in *Proc. CHI*, 2012, pp. 433–452.

[8] Google Chrome Developers, "Manifest V3: The next generation of extension platforms," 2021. [Online]. Available: https://developer.chrome.com/docs/extensions/mv3/

[9] T. Bigham et al., "Web content extraction for accessibility: Structuring cluttered webpages for improved usability," in *Proc. W4A*, 2017, pp. 1–10.