Text Summarization Plugin using LLMs

Anurag Choudhary
The University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
anuragc3@illinois.edu

Hammad Ali
The University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
hali53@illinois.edu

Manu Ravichandrakumar
The University of Illinois at Urbana-Champaign
Illinois, USA
manur2@illinois.edu

Sahil Bhende
The University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
sbhende2@illinois.edu

Abstract

This paper presents a Chrome extension that leverages large language models (LLMs) to generate concise summaries of web articles, enabling users to quickly assess content relevance and quality before committing to a full read. Designed to address the rise of misleading headlines, information overload, and the cognitive cost of tab switching, the tool provides in-page summarization, related content retrieval, and optional accessibility features such as text-to-speech. Users can customize the summary length and input additional instructions to tailor the output to their needs. The system architecture combines browser-based content extraction with cloud-based LLM APIs (e.g., ChatGPT, Gemini), offering a fast and extensible summarization pipeline.

The source code for the plugin is publicly available on GitHub [1].

CCS Concepts

• **Computing methodologies** → **Natural language processing**; • **Information systems** → **Summarization**; • **Software and its engineering** → *Software design engineering*; • **General and reference** → *Design.*

Keywords

Text Summarization, Large Language Models, ChatGPT Gemini APIs, Context Retrieval, Preprocessing, Token Extraction, Prompt engineering, Information Retrieval, Natural Language Processing.

ACM Reference Format:
Anurag Choudhary, Hammad Ali, Manu Ravichandrakumar, and Sahil Bhende. 2025. Text Summarization Plugin using LLMs. In *Proceedings of May 09–11, 2025.* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

May 09–11, 2025,

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Online readers are frequently overwhelmed by the volume of digital content they encounter, such as news stories, research blogs, opinion pieces, and more. Although many users are interested in staying informed, they often lack the time or motivation to read entire articles upfront. This raises a key question: can readers quickly determine whether an article is worth their time before investing in a full read? Inspired by the "three-pass approach" to academic paper reading, where the first pass provides a high-level understanding, we present a browser-based tool that brings a similar paradigm to web and news articles.

Our Chrome extension automatically summarizes the main content of the currently open webpage using large language model (LLM), such as ChatGPT and Gemini. These summaries provide readers with a concise overview of the subject matter of the article, enabling them to assess its relevance at a glance. Whether users want to determine if the article aligns with their interests, verify its topical scope, or simply absorb its key message quickly, this tool facilitates the initial low-effort pass through the content.

While summarization functionality is the central feature, the tool also offers optional support mechanisms to enrich the reading experience. It can identify and define key terms, fetch related articles for broader context, and display this information in a lightweight side-panel overlay, without forcing users to leave the current page. The system is designed for students, journalists, researchers, and general readers alike, and represents a step toward more efficient, informed, and user-directed web consumption. By helping users decide when to engage deeply or move on, the tool supports both curiosity and efficiency in digital reading.

The plugin is implemented using modern web extension technologies, including JavaScript and Chrome's Manifest V3 framework, and integrates large language model services through modular API calls. This architecture enables the tool to remain lightweight, efficient, and easy to extend for future functionality such as personalization or summary evaluation support.

2 Motivation

Building upon the problem outlined in the introduction, it is important to recognize that the need for quick and reliable summarization is not just about convenience of reading less/short content. It's also a defense mechanism in an increasingly noisy and manipulative online information landscape. In today's media ecosystem, many websites (especially news outlets and opinion platforms) prioritize

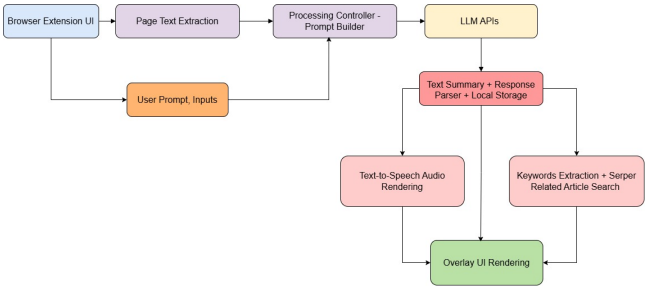


Figure 1: System Architecture

engagement metrics like clicks, shares, and ad impressions over reader clarity or journalistic responsibility. As a result, users are frequently exposed to articles with misleading or sensational headlines, carefully crafted to provoke curiosity without reflecting the actual content. Subtle rhetorical tricks, selective phrasing, or out-of-context claims are often embedded within the article itself to preserve plausible deniability while still guiding readers toward emotionally charged or biased interpretations.

This environment makes it difficult for readers to assess the reliability, intent, or actual substance of what they're about to read. Often, by the time the reader realizes that the content is misleading or empty, they've already invested valuable time and attention. Worse, even casual exposure to such content can reinforce misinformation or ideological bias. This repeated experience contributes to a growing sense of skepticism and fatigue, making it increasingly difficult for users to discern which content is worth their time. This tool was born out of the frustration with that very pattern.

This plugin aims to restore some balance to that equation. By surfacing the essence of a webpage up front, independent of how it's framed or marketed, the tool empowers users to make informed decisions before committing to a full read. It acts as a neutral intermediary, summarizing the content without amplifying the emotional tone or rhetorical spin. In doing so, it helps readers regain control over their attention and shields them, at least partially, from the manipulation embedded in modern digital discourse.

Several browser extensions, including TL;DR, SMMRY, offer in-browser summarization but suffer from notable limitations. Many rely on keyword extraction or heuristic-based methods, resulting in shallow or imprecise summaries. Others disrupt the user experience by redirecting to external dashboards, breaking the reading flow. Most lack integrated support for related content, term definitions, or handling dynamic and paywalled pages, highlighting a gap between existing tools and the need for seamless, context-aware reading support.

3 System Architecture

As shown in Figure 1, the system architecture of the summarization plugin is organized into three main stages: (1) user interaction and prompt construction, (2) LLM communication and additional API calls, and (3) final UI rendering. This modular structure allows the tool to balance responsiveness and extensibility while leveraging external large language model APIs effectively.

3.1 User Input Handling and Prompt Construction

The summarization process begins with the browser extension UI , where the user can initiate a summary request and optionally specify constraints such as a maximum word count or additional contextual instructions. These inputs are captured by the extension's content script and forwarded to the *Processing Controller*.

Simultaneously, the script extracts the primary content of the currently active webpage. This is typically done via DOM traversal and readability heuristics to isolate relevant text blocks (e.g., article body, blog content, etc.). Once both the user inputs and the page content are available, the processing controller assembles a prompt in natural language format that is compatible with the selected LLM backend (e.g., ChatGPT, Gemini). This prompt is dynamically constructed to reflect user preferences while preserving enough of the source context to generate an accurate and coherent summary.

3.2 LLM Communication and Content Enrichment

The constructed prompt is then sent to an external LLM API endpoint using a secure and asynchronous fetch mechanism. The system supports integration with multiple LLM providers, allowing flexibility in terms of pricing, model behavior, or deployment environment. In the early implementation, API calls to the LLMs and enrichment modules were made sequentially, which led to noticeable latency. To address this, we introduced parallelization in the request pipeline, allowing summarization and related content retrieval processes to run concurrently. This significantly reduced the total summary generation time and improved the responsiveness of the overall system. The returned response from the LLM typically includes the summarized content in plain text format, along with optional metadata for downstream parsing.

After receiving the LLM generated summary, the system executes a dual path postprocessing flow. First, the response is passed through a parser module that extracts key elements (e.g., headings, bullet points, tone indicators) and stores them in local storage for fast retrieval. Second, the same response is optionally converted into speech using a Text-to-Speech (TTS) engine. This feature is particularly useful for accessibility purposes and for users who prefer audio-based content consumption.

Concurrently, the original page content is also processed for related information. This involves keyword extraction (e.g., named entities, topics) followed by a lightweight query generation module that interacts with third party search APIs (e.g., Serper) to retrieve supporting articles or alternative perspectives. These related snippets complement the summary by helping users explore additional angles without leaving the page.

3.3 UI Rendering and In-Browser Presentation

The final stage of the pipeline is dedicated to rendering the results within the active browser context. The *Overlay UI Renderer* is a collapsible panel injected directly into the current page layout, ensuring that users can view summaries and related content without navigating away from the original article.

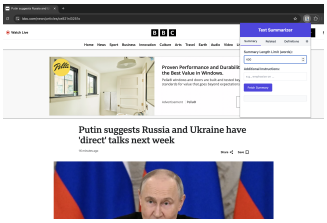


Figure 2

This renderer fetches the parsed summary, audio output (if enabled), and related articles from local storage or memory and structures them into visually distinct sections. The design is optimized for readability, with clear typography, inline tooltips for definitions, and minimal cognitive load. Importantly, the rendering layer is decoupled from the core processing logic, making it easy to extend the UI to support feedback mechanisms (e.g., thumbs up/down), future personalization, or even topic filtering. This component completes the system loop by ensuring that the information presented to the user is clear, accessible, and tightly integrated into their existing browsing experience.

4 User Workflow

This section walks through a typical user's interaction with the extension, from initial installation and API key configuration to summary generation, and exploration of additional relevant content.

4.1 Installation and API Key Configuration

Users install the extension (see our Github repository [1] for details) and navigate to the extension's Settings tab to enter and save their OpenAI and/or GeminiAPI keys (Figure 3a).

4.2 Fetching a Summary

To summarize an article, the user simply opens any news or blog page in Chrome and clicks the extension icon. Optional controls allow specification of a maximum summary length and adding custom instructions to tailor tone or focus (Figure 3b). Upon clicking Fetch Summary, the extension extracts the article text, makes the API calls, and displays the resulting summaries from each LLM (Figure 3c). For accessibility or hands-free convenience, users can make use of the built-in text-to-speech module (Figure 3d).

4.3 Exploring Additional Information

Beyond summaries, the extension offers two more tabs for deeper context: • Related Articles: The Related tab uses key terms from the article to fetch relevant articles, letting users deep-dive into relevant stories. (Figure 3e). • Definitions: The Definitions tab lists important terms along with concise definitions, helping users quickly understand domain-specific concepts (Figure 3f).

By following this streamlined workflow, users can efficiently digest and explore online content.

5 Future Work

5.1 Multi LLM Summarization

Recent work by Fang et al.(2025) [2] introduces a multi-LLM summarization framework, showing that an ensemble of large language models consistently outperforms any single model and aligns more closely with human judgments. A naive single LLM approach for summarization might miss critical details, particularly in long, information-density documents. The authors evaluate two coordination topologies: • Centralized: Each LLM produces an independent draft; a judge LLM then ranks the drafts and delivers a final summary. • Decentralized: LLMs iteratively rank each other's drafts (including their own) in multiple rounds until they reach consensus on the highest-scoring summary.

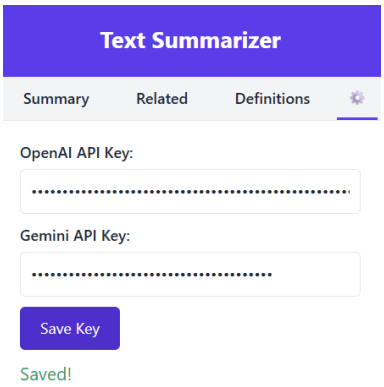
Both strategies mitigate model-specific bias and exploit complementary knowledge. Our current tool already queries OpenAI and Gemini in parallel, but merely displays their outputs side by side. In the next iteration, we hope to implement Fang et al.'s multi-LLM text summarization model to generate more fuller and comprehensive summaries, particularly for long articles.

5.2 Beyond News — Expanding to Other Domains

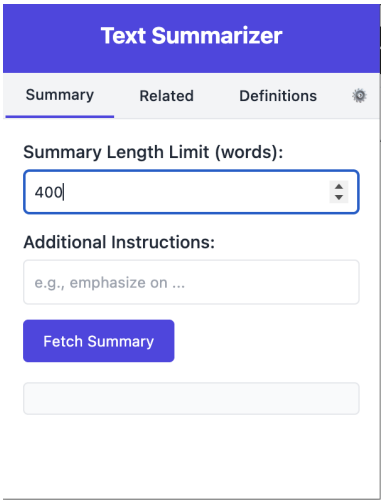
Our current text extraction and summarization pipeline is primarily focused on news articles, which tend to be short, well-structured, and easily parsable. In future versions, our aim is to extend support to long-form and structurally complex documents, particularly in scholarly and legal domains. To handle these, we plan to build on the multi-LLM summarization framework, leveraging the centralized topology for better coverage and coherence. Specifically, we plan to: • Chunk long documents by sections (e.g., abstract, methods, conclusions, clauses), • Assigning parallel summarization tasks to different LLMs for each chunk, • Using a final synthesis LLM to combine the individual chunk summaries into one cohesive document-level summary, • Optionally running multiple refinement rounds to meet quality thresholds or semantic consistency. This approach enables scalability to large documents while preserving nuanced, domain-specific information. It also opens the door to domain-adaptive prompts and evaluation strategies tailored to academic or policy-driven texts.

6 Conclusion

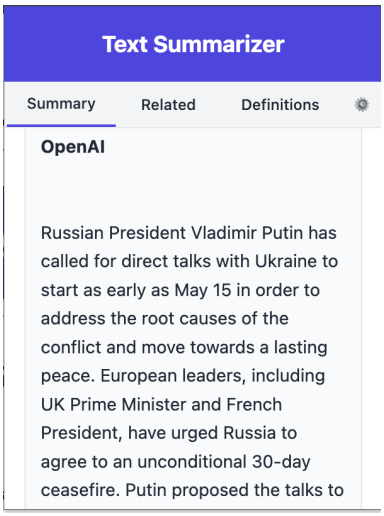
This work presents a browser-based summarization tool designed to help users quickly evaluate the relevance and substance of web articles using LLM generated summaries. By integrating modern content extraction, prompt customization, and contextual augmentation features into a lightweight Chrome extension, the system improves the digital reading experience with minimal user friction. Future work will focus on a multi LLM framework that can dynamically select or ensemble outputs from multiple models based on content type, user preferences, or comparative performance. Additional enhancements, such as fine-grained user feedback loops, and adaptive prompt tuning, are also planned to further improve system versatility and user trust.



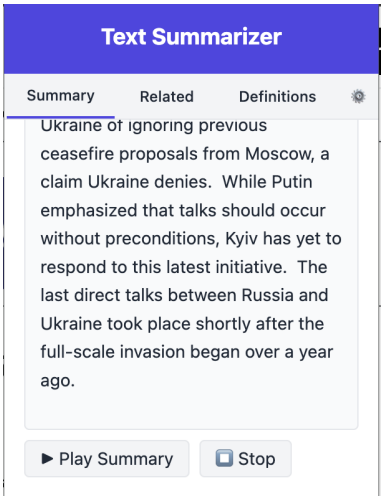
(a) API Key Setup



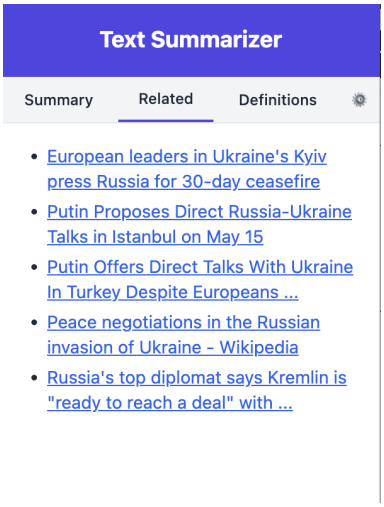
(b) Summary Options



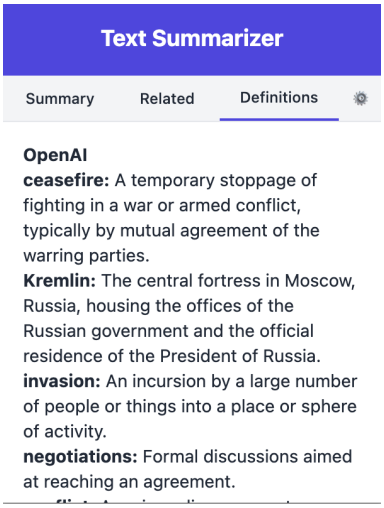
(c) Generated Summary



(d) Text-to-Speech Controls



(e) Related Articles



(f) Definitions Tab

Figure 3: User workflow through the extension: (a) API Key Setup, (b) Summary Options, (c) Generated Summary, (d) Text-to-Speech Controls, (e) Related Articles, (f) Definitions Tab.

7 Online Resources and Implementation References

The development of the browser extension was informed by official documentation provided by Google for the Chrome Extensions platform. Specifically, we followed the Manifest V3 specification, which introduces a modernized architecture for extension performance and security. The documentation includes guidelines on background service workers, content scripts, permissions handling, and UI integration [4].

The summarization capabilities of the plugin rely on large language models accessed via API endpoints. We integrated both OpenAI's ChatGPT [6] and Google's Gemini API [3], using their official SDKs and RESTful API interfaces. These resources provided critical details about model capabilities, rate limits, authentication, prompt formatting, and streaming response handling. Where applicable, we applied prompt engineering best practices to maximize summary relevance and coherence.

To fetch related articles and perform contextual enrichment, we used third-party search APIs such as Serper.dev [7], which wraps Google Search with a developer-friendly JSON API. This allowed us to perform programmatic query formulation and results retrieval. Additional utilities like Mozilla's Readability.js were explored for content extraction, and the Web Speech API [5] was considered for implementing the optional text-to-speech functionality.

Collectively, these documentation sources served as the foundation for both the technical implementation and architectural design choices of the system.

References

[1] Anurag Choudhary, Hammad Ali, Manu Ravichandrakumar, and Sahil Bhende. 2024. Text Summarizer Chrome Extension. https://github.com/anuragc3/cs510-g26. CS 510 - Advanced Information Retrieval, UIUC.

[2] Jiangnan Fang, Cheng-Tse Liu, Jieun Kim, Yash Bhedaru, Ethan Liu, Nikhil Singh, Nedim Lipka, Puneet Mathur, Nesreen K. Ahmed, Franck Dernoncourt, Ryan A. Rossi, and Hanieh Deilamsalehy. 2025. Multi-LLM Text Summarization. arXiv:2412.15487 [cs.CL] https://arxiv.org/abs/2412.15487

[3] Google AI. 2024. Gemini API Documentation. https://ai.google.dev.

[4] Google Developers. 2023. Chrome Extension Manifest V3 Documentation. https://developer.chrome.com/docs/extensions/mv3/.

[5] Mozilla Developer Network. 2023. Web Speech API. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API.

[6] OpenAI. 2023. OpenAI API Documentation. https://platform.openai.com/docs.

[7] Serper.dev. 2023. Serper API for Web Search. https://serper.dev/docs.