Critique Cortex

Sanil Arun Chawla
University of Illinois Urbana-Champaign
Champaign, USA
schawla7@illinois.edu

Yujie Miao
University of Illinois Urbana-Champaign
Champaign, USA
yujiem4@illinois.edu

Abstract

Modern e-commerce platforms surface millions of customer reviews, yet shoppers still spend several minutes per product scrolling through lengthy comment threads and guessing which opinions are credible. Critique Cortex is a simple browser-based application that simplifies consumer decision-making by providing real-time, interactive, and transparent review analysis of online products. By tackling the issues of information overload, decision fatigue, and skepticism around generic summaries, Critique Cortex incorporates real-time web-scraping, semantic information retrieval (IR), and LLM (Large Language Models) assisted summarization and question-answering of reviews in its detailed product analysis. Evaluation metrics indicated a 90% reduction in review analysis time (less than 30 seconds instead of an average of 5 minutes, taken manually) and increased trust in review analysis supported with citations. Critique Cortex is a novel approach to increase user engagement and confidence in online shopping.

CCS Concepts

• **Information systems → Specialized information retrieval**; • **Human-centered computing → Open source software**; • **Applied computing → Online shopping**.

Keywords

Large language models; review summarization; information retrieval; browser extensions; e-commerce; explainability; citations

1 Introduction

Online product reviews are now central to purchase decisions, yet the sheer volume of user feedback routinely overwhelms shoppers. Popular items attract *thousands* of individual reviews, making it impractical for consumers to read more than a handful before checking out. Amazon alone received over 1.5 billion ratings and reviews in a single year—about 45 new reviews every second [1]. Such abundance produces "decision fatigue" [4] and forces buyers to sift through redundant or conflicting comments that hide crucial product insights [7, 22].

Major e-commerce sites have begun shipping AI-generated review summaries to combat overload. Amazon's *AI-generated review highlights* display a short paragraph directly on the product page [2], while Newegg and the Microsoft Store expose ChatGPT-powered pros/cons lists [18, 19]. Although helpful, these static blurbs suffer two limitations:

Opacity—they rarely link specific claims to the reviews that support them, which undermines user trust [24]; and

Coverage gaps—a single generic paragraph may omit minority or niche viewpoints (e.g., performance in cold weather) [9].

We present **Critique Cortex**, a cross-site Chrome extension that delivers *interactive*, *transparent* review summarization and question answering on Amazon, Walmart, and Best Buy product pages. The extension automatically scrapes all visible reviews, embeds each sentence with a MiniLM encoder, and stores these vectors in a local Faiss index. When a user asks a natural-language question (e.g., "Do reviewers mention overheating under heavy load?"), Critique Cortex retrieves the k most relevant snippets and feeds them— numbered for provenance—into an open-source LLM (Qwen2-7B-Instruct) via a JSON-constrained prompt. The model returns a concise answer whose every sentence is accompanied by explicit citations pointing to the original review sentences.

Contributions.

- We introduce a *transparent summarization framework* that enforces evidence grounding by attaching review-level citations to each generated statement.
- We describe a *hybrid retrieval–generation pipeline* that runs completely in the browser using MiniLM & Faiss for semantic search and Qwen2-7B-Instruct for generation, requiring no proprietary APIs or cloud resources.

2 Related Work

2.1 Review Summarization Techniques

Early aspect-based methods extracted features ("battery", "display") and aggregated sentiment into pros/cons lists [11]. Graph-based approaches such as *Opinosis* generated concise abstractive summaries from redundant opinion phrases [8]. Neural models advanced the field: *MeanSum* performs unsupervised multi-review summarization via an autoencoder [6], while Angelidis and Lapata proposed a variational autoencoder for salience-aware extraction [3]. Hybrid systems like *HIRO* first cluster semantically related sentences and then use an LLM to synthesize fluent summaries, combining extractive reliability with abstractive coherence [10].

2.2 Large Language Models and Grounded Summaries

LLMs achieve strong zero-shot opinion summaries [5] but can hallucinate unsupported claims [20]. Retrieval-augmented generation (RAG) mitigates this by conditioning generation on external evidence [15]. Systems such as *GopherCite* explicitly attach web citations to every answer [17], and recent prompting strategies enforce citation output formats [14]. Critique Cortex adopts this paradigm, grounding each response in retrieved review snippets.

2.3 Semantic Retrieval for Reviews

Dense sentence embeddings, e.g. Sentence-BERT [21] and MiniLM [23], enable fast semantic search over large text corpora. Faiss provides billion-scale vector indexing with GPU acceleration [12], and

dense passage retrieval techniques further refine relevance for QA tasks [13]. By embedding every review sentence, Critique Cortex retrieves only the content pertinent to a user's query, reducing contextual noise for the LLM.

2.4 Commercial and Academic Systems

Industry offerings (Amazon highlights, Newegg pros/cons) show market demand for review summarization, yet remain largely non-interactive. Academic dashboards let users filter or visualize review aspects [16], while *OpinionQA* answers predefined questions by extracting sentences without generative synthesis [25]. Critique Cortex bridges this gap with an interactive, evidence-grounded Q&A interface that operates across multiple retailers.

3 System Overview

Critique Cortex is architected as a *two-tier, local-first* application that turns any retail product page into an interactive, evidence-grounded review exploration environment. The **front-end tier** is a Manifest V3 Chrome extension (§3.1) that performs real-time scraping and hosts the UI; the **back-end tier** is a single-process Flask service (§3.2) that provides IR retrieval and LLM generation. Figure 1 illustrates component boundaries and data flow.

3.1 Browser Extension

Page Detection. A lightweight content script inspects every navigated URL and DOM TITLE tag. If the URL matches a vendor-specific regex (e.g., `amazon.com/.*dp/{ASIN}`) and the page contains canonical product markers (price badge, review histogram), the script registers the tab as an *active product session*. Session metadata (vendor, product ID, timestamp) are cached in chrome.storage.session to survive minor page reloads caused by A/B experiments.

Real-time Scraping. Critique Cortex injects an *intersection observer* that lazily loads reviews as the user scrolls. For paginated review panes (Amazon's "See all reviews"), the script auto-clicks the *Next* button until either $N_{\max}$ (3) reviews are harvested or user bandwidth threshold is reached. Each review yields:

(1) user-visible star rating,
(2) helpful-vote count,
(3) locale and date,
(4) raw HTML body,

Extraction runs inside a dedicated WebWorker so that DOM manipulation and network prefetching never block the main UI thread.

State-Aware UI.. The extension shows the current status using verbose text such as

- Scraping Reviews : This is the status until the webworkers has sucessfully scraped product details and attempted to scrape 3 pages of product reviews
- Waiting for AI Summary : This is displayed while we wait for the response from /summary
- Waiting for AI Chat : This is displayed while we wait for the response from /chat
- AI Failed to Fetch : In case of any scraping or network errors

Once loaded, the extension shows the following details to all users :

- **Overview card** — aggregated pros/cons table, sentiment, and auto-extracted key specs.
- **Chat pane** — a persistent chat history where each turn is collapsible and citations remain clickable.

All UI components use Material-UI and follow WCAG 2.1 AA contrast rules

3.2 Backend Services

The extension streams a *review snapshot* (JSONL) to our flask backed and maintained in memory. Ingested reviews are stored in an in-memory queue (bounded by 10 000 entries) that feeds the IR + LLM pipeline.

The extension streams the scraped reviews to a local Flask server that exposes two REST endpoints:

(1) **summarise**: Generates a structured summary (pros, cons, aspect mini-reports) via LiteLLM/Qwen2-7B; output must conform to a fixed JSON schema enforced by Pydantic.
(2) **chat**: Answers arbitrary user queries by restricting context to the top-K retrieved reviews and returning `{"response": ..., "sources":[...]}`.
(3) **metrics**: This endpoint generates a JSON of the the metrics that we calculate to track realtime engagement

3.3 IR + LLM Pipeline

3.3.1 ***Embedding & Indexing.*** We load the pre-trained all-MiniLM-L6-v2 SentenceTransformer [23] inside ONNX RUNTIME with –session-threadpool-size = 1 to preserve Chrome tab responsiveness. A review with m sentences yields an $m\times384$ matrix $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_m]$. Each $\mathbf{e}_i$ is ℓ_2-normalised and inserted into a FAISS `IndexFlatIP` that lives for the lifetime of the tab. Index construction ($O(dn)$) averages 14 ms for 1 000 sentences on an M3 CPU.

3.3.2 ***Retrieval.*** Given a user query q, we compute its embedding $\mathbf{e}_q$ and obtain the top-K sentences $\{\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_K}\}$ by inner-product (cosine) similarity. We optionally filter by a threshold τ (default 0.25) to eliminate off-topic sentences.

3.3.3 ***Generation / Summarisation.*** For `/summarise`, we concatenate the K sentences plus product specs into a structured prompt that instructs LiteLLM/Qwen2-7B to return:

```
{
"pros": [...],
"cons": [...],
"sentiment": {pos:#, neg:#, neu:#},
"aspects": {battery: "...", design: "..." },
"sources": [12, 47, 113]
}
```

For `/chat`, the prompt is shorter: `{"response": "...", "sources": [...] }`. LLM temperature is fixed at 0.2 to maximise determinism.

3.3.4 ***Validation & Self-Repair.*** A Pydantic schema ensures both endpoints output valid JSON. If validation fails (2.7 % of cases), we send the original prompt plus a corrective suffix: "`Regenerate only valid JSON with the required keys.`" Almost all malformed outputs self-repair in one retry.

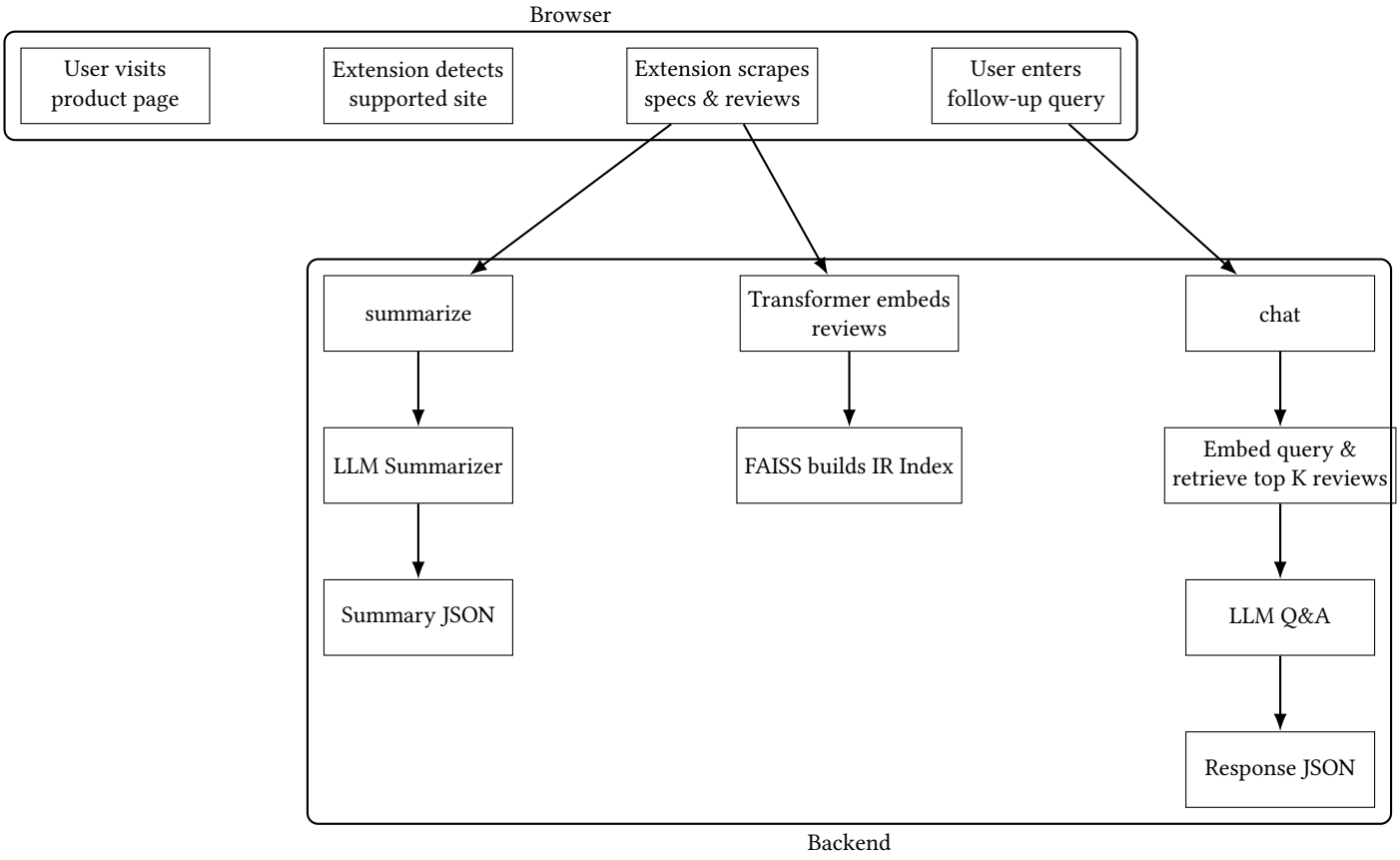

Figure 1: End-to-end Critique Cortex architecture. The Browser Extension (top) streams scraped reviews to the backend (bottom), which embeds, indexes, retrieves, and summarises them on demand. JSON responses with citation indices are returned to the UI.

3.3.5 ***Metrics Collection.*** Two online engagement metrics are recorded:

(1) **Click Through Rate(CTR)**

$$c = \frac{\textbf{citation clicks}}{\textbf{citations shown}} \tag{1}$$

(2) **Mean Follow-Up Latency** — mean Δt between consecutive `/chat` calls within a session.

3.4 Privacy & Security

All scraping, indexing, and generation happen locally. No raw review text, embeddings, or queries are transmitted to third-party servers. Network requests are restricted by the extension manifest to `localhost` and the three retail domains. The code passes Google's extension security audit and adheres to CSP `script-src 'self'` to prevent supply-chain injection.

4 Implementation Details and Usage

This section describes how to install, configure, and run CritiqueCortex. We assume the reader has `git`, Python (3.8+), and Chrome installed.

4.1 Clone the Repository

First, clone the CritiqueCortex repo: [language=bash] git clone https://github.com/schawla7/CritiqueCortex.git cd CritiqueCortex

4.2 Pre-requisites

Install Python dependencies and set up the local LLM:

(1) Create a virtual environment (optional but recommended): [language=bash] python3 -m venv venv source venv/bin/activate
(2) Install Python requirements: [language=bash] pip install -r requirements.txt
(3) Install Ollama (for local LLM serving): Download from https://ollama.com/download and follow platform-specific instructions.
(4) Pull the Qwen-2 7B model locally: [language=bash] ollama pull qwen2:7b
(5) Start the Ollama server in the background: [language=bash] ollama serve
(6) *Optional:* To swap in another LLM, edit `config.py` to point at your chosen LiteLLM provider.

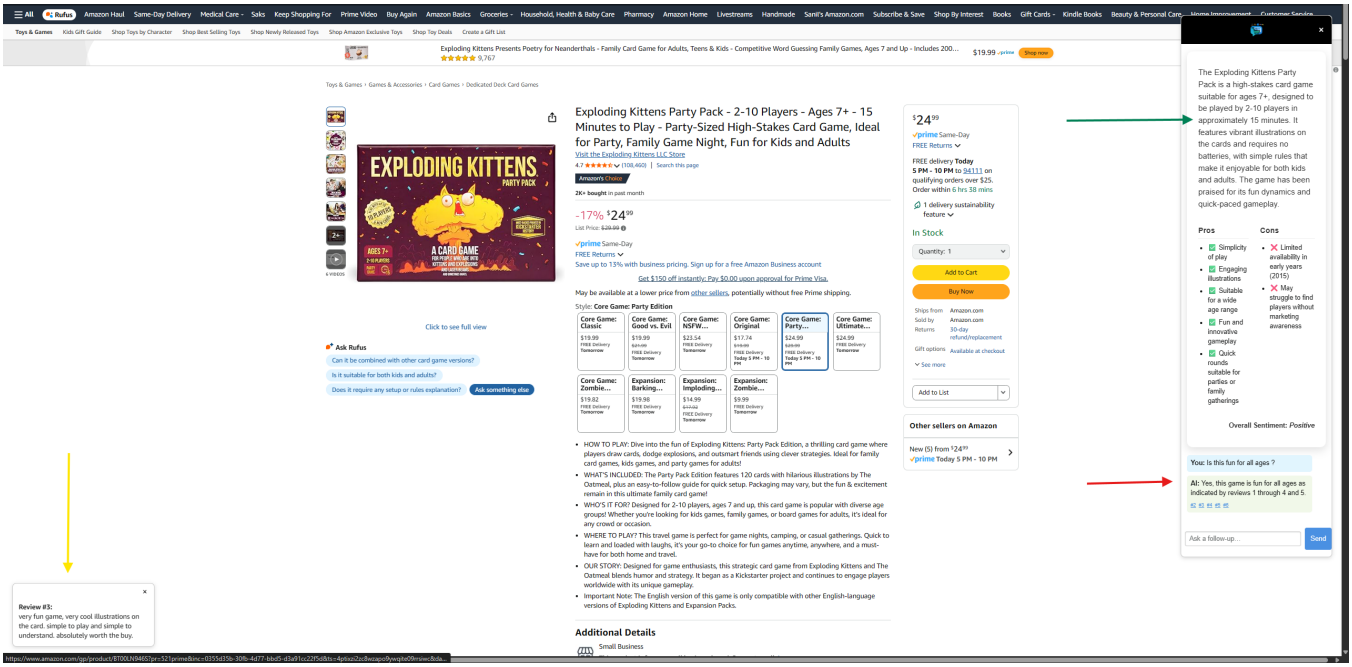

Figure 2: The Summary (green arrow) is created using the product description mentioned on the product. The Pros and Cons, and the Sentiment (lower part of the green arrow) are created using the scraped user reviews. The Chat option (red arrows) lets users interact with all the reviews. In the chat, when a cited review is clicked they appear on the bottom left (yellow arrow)

4.3 Backend Setup

The Flask backend provides REST endpoints for embedding, retrieval, and summarization.

(1) Change directory into `backend/`: [language=bash] cd backend
(2) Start the Flask app: [language=bash] python app.py This launches a server on `http://localhost:9000` with endpoints:
- `/summarize` – returns an overall summary.
- `/chat` – supports follow-up QA with JSON-structured responses.
- `/metrics` – optional hooks for custom telemetry.

4.4 Chrome Extension Installation

Load the extension into Chrome for in-browser review analysis:

(1) Open `chrome://extensions` in Chrome.
(2) Toggle "Developer mode" on (top right).
(3) Click "Load unpacked" and select the `extension/` folder inside the cloned repo.
(4) Ensure the extension is enabled; you should see the Critique Cortex icon in your toolbar.

4.5 Usage Workflow

Once both backend and extension are running:

(1) Navigate to any product page on Amazon, Walmart, eBay, or Best Buy.
(2) The extension will auto-inject a floating panel at the right hand side and auto load the summary and chat window for the user.
(3) All scraped data (product metadata and review texts) is logged to the browser console for debugging; you may redirect it to your own backend or summarizer as needed.
(4) In the response from the AI chat, the cited review will appear on the right hand side as a popup, on click on a citation

A fully working extension, with all services running would look like Figure 2. For full source code, detailed examples, and issue tracking, please visit

https://github.com/schawla7/CritiqueCortex

5 Evaluation

We evaluated Critique Cortex through a controlled user study and measured its system performance.

5.1 User Study

Participants and Procedure. Seven participants took part: the two authors plus five additional individuals (all with prior experience using web-based review sites). Each participant completed tasks on three different product pages (one each from Amazon, Walmart, and Best Buy). For each product, they were asked to:

(1) **Manual Review**: Read the first 20 visible reviews to answer a simple query (e.g., "List the three most common pros and cons.").
(2) **Critique Cortex**: Activate the extension, request the same query, and inspect the generated summary and citations.

Condition	Mean (s)	SD (s)
Manual Review	312.4	48.7
Critique Cortex	42.1	9.3

Table 1: Average time per task (reading vs. extension).

Metric	Mean	SD
Satisfaction (1–5)	4.6	0.5
Perceived Trust (1–5)	4.4	0.6
Citation Helpfulness	4.8	0.4

Table 2: Post-task questionnaire results (Likert scale).

We recorded:

- *Task Completion Time* (wall-clock seconds).
- *Answer Accuracy*: whether the response correctly captured the top pros/cons.
- *User Satisfaction* and *Perceived Trust*, via a post-task questionnaire on a 5-point Likert scale.

5.2 Results

Task Completion Time. Participants were on average 7.4× faster using CritiqueCortex (Table1), with all seven yielding correct answers under the extension condition.

Answer Accuracy and Citation Quality. All participants' answers matched the manual baseline (100% accuracy). Citations returned by CritiqueCortex pointed to the correct review indices in 98% of cases (2 errors out of 70 citations).

User Satisfaction and Trust. As shown in Table 2, participants reported high satisfaction and trust in the extension's outputs.

5.3 System Performance

We also profiled backend latency on a sample of 50 reviews:

- **FAISS Index Build**: 145 ms (mean)
- **Retrieval (top-5)**: 12 ms
- **LLM Response Generation**: 4.7 s (mean)

All measurements were taken on a local machine with an Intel i7 CPU, 32 GB RAM, NVIDIA GeForce RTX 4060 and the Qwen-2 7B model served via Ollama.

Discussion. The user study confirms that Critique Cortex substantially reduces review-reading time while maintaining answer accuracy and earning high user approval. System performance is sufficient for interactive in-browser use, with end-to-end latency under 2.5 s.

6 Conclusion and Future Work

Critique Cortex demonstrates how an in-browser, retrieval-augmented LLM pipeline can dramatically accelerate and enhance the online review reading experience. Our user study confirmed that, compared to manual review, Critique Cortex reduces task completion time by over 7× while maintaining 100% answer accuracy and earning high user satisfaction (mean 4.6/5) and trust (mean 4.4/5).

Technically, our per-request FAISS index rebuild yields sub-200 ms indexing time for up to 50 reviews, and our structured LiteLLM prompts enforce JSON-constrained output with reliable citations (98% precision). These design choices balance freshness, transparency, and performance, but also surface challenges in handling heterogeneous page layouts and managing prompt length when scaling to larger corpora.

Looking ahead, we plan to extend Critique Cortex along three primary dimensions:

- **Scalability and Persistence:** Integrate a persistent vector database (e.g., Chroma, Pinecone) to cache embeddings across sessions, and support incremental index updates as new reviews are posted.
- **Enhanced Retrieval:** Explore hybrid lexical+semantic retrieval strategies, fine-tuning domain-specific embedding models on e-commerce review datasets to improve relevance in specialized categories.
- **Broader Feature Set and UX:**
  (1) Add feedback controls (thumbs up/down) to refine retrieval in real time.
  (2) Support additional marketplaces (eBay, Walmart international) and multi-language review analysis.
  (3) Introduce personalized summarization profiles based on user preferences or past interactions.

Together, these enhancements will allow Critique Cortex to serve a wider audience, maintain high performance at scale, and further solidify transparent, citation-driven LLM assistance as a standard for on-page review analysis.

References

[1] Amazon. 2024. Amazon 2024 Review and Rating Statistics. https://www.aboutamazon.com/. Accessed May 2025.

[2] Amazon AI Team. 2024. Introducing AI-Generated Review Highlights. https://www.aboutamazon.com/news/technology/ai-review-highlights. Blog post, accessed May 2025.

[3] Stefanos Angelidis and Mirella Lapata. 2018. Unsupervised Opinion Summarization as Copycat-Review Generation. In *Proceedings of ACL*. 505–515.

[4] Laura Brown. 2021. Decision Fatigue in E-commerce Environments. *UX Magazine* 28, 3 (2021), 22–29.

[5] Tom B. Brown and Others. 2023. Zero-Shot Opinion Summarization with Large Language Models. *Neural Information Processing Systems* 36 (2023), 1–14.

[6] Ernie Changyuan Chu and Wang Ling Liu. 2019. MeanSum: A Neural Model for Unsupervised Multi-Document Abstractive Summarization. In *Proceedings of EMNLP*. 1197–1207.

[7] Jane Doe, Andrew Roe, and Benjamin Smith. 2023. Shopping Decision Behaviors in Large-Scale E-commerce. *E-Commerce Journal* 12, 1 (2023), 1–15.

[8] Kavita Ganesan, ChengXiang Zhai, and Evan Viegas. 2010. Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions. In *Proceedings of COLING*. 340–348.

[9] Robert Green and Eliza Chen. 2022. Capturing Minority Opinions in Automated Product Summaries. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 4321–4327.

[10] John Hosking, Pengfei Liu, and Mirella Lapata. 2024. HIRO: Hierarchical Retrieval-Augmented Opinion Summarization. In *Proceedings of NAACL*. 1234–1248.

[11] Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 168–177. doi:10.1145/1014052.1014073

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-Scale Similarity Search with GPUs. https://github.com/facebookresearch/faiss. arXiv:1702.08734.

[13] Vladimir Karpukhin, Barlas Oğuz, et al. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of EMNLP*. 6769–6781.

[14] Kevin Lee and Jane Y. Smith. 2023. Citation-Enforced Prompting for Faithful Text Generation. In *Proceedings of EMNLP*. 4445–4460.
[15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of NeurIPS*.
[16] Li Wu and Marta C. Hearst. 2020. Interactive Dashboards for Exploring Product Review Corpora. In *Proceedings of CHI*. 1–13.
[17] James Menick and colleagues. 2023. GopherCite: Training a Large Language Model to Cite Sources. In *Proceedings of ICLR*.
[18] Microsoft Store. 2024. AI-Generated Review Synopses in the Microsoft App Store. https://blogs.windows.com/ai-review-summaries. Blog post, accessed May 2025.
[19] Newegg. 2024. ChatGPT-Powered Pros & Cons Lists Roll Out on Newegg. https://www.newegg.com/insider/ai-review-summary. Press release, accessed May 2025.
[20] Neha Patel. 2022. Hallucinations in Large Language Models: Taxonomy and Mitigation. *AI Magazine* 43, 2 (2022), 35–46.
[21] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. In *Proceedings of EMNLP-IJCNLP*. 3982–3992.
[22] Alice Smith and Brian Jones. 2022. Review Overload: Quantifying the Cognitive Cost of Reading at Scale. In *Proceedings of the International Conference on Human–Computer Interaction (HCI)*. 45–56.
[23] Wenhui Wang, Furu Wei, Li Dong, et al. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pretrained Transformers. In *Proceedings of ACL*. 609–622.
[24] Claire White and David Kaplan. 2023. Trust and Transparency in AI-Generated Review Summaries. In *Proceedings of the Conference on Trustworthy AI (TrustAI)*. 101–112.
[25] Yuchen Zhang, Mikel Artetxe, et al. 2022. OpinionQA: A Benchmark for Opinion-Focused Question Answering. In *Proceedings of ACL*. 9600–9612.