PaperDebugger: A Plugin-Based Multi-Agent System for In-Editor Academic Writing, Review, and Editing

Junyi Hou
National University of Singapore
Singapore
e0945797@u.nus.edu

Andre Lin Huikai
National University of Singapore
Singapore
andre_lin@u.nus.edu

Nuo Chen
National University of Singapore
Singapore
nuochen@u.nus.edu

Yiwei Gong
Independent Researcher
Singapore
imwithye@gmail.com

Bingsheng He
National University of Singapore
Singapore
dcsheb@nus.edu.sg

Abstract

Large language models are increasingly embedded into academic writing workflows, yet existing assistants remain external to the editor, preventing deep interaction with document state, structure, and revision history. This separation makes it impossible to support agentic, context-aware operations directly within LaTeX editors such as Overleaf. We present **PaperDebugger**, an **in-editor**, **multi-agent**, and **plugin-based** academic writing assistant that brings LLM-driven reasoning directly into the writing environment. Enabling such in-editor interaction is technically non-trivial: it requires reliable bidirectional synchronization with the editor, fine-grained version control and patching, secure state management, multi-agent scheduling, and extensible communication with external tools. PaperDebugger addresses these challenges through a Chrome-approved extension, a Kubernetes-native orchestration layer, and a Model Context Protocol (MCP) toolchain that integrates literature search, reference lookup, document scoring, and revision pipelines. Our demo showcases a fully integrated workflow, including localized edits, structured reviews, parallel agent execution, and diff-based updates, encapsulated within a minimal-intrusion user interface (UI). Early aggregated analytics demonstrate active user engagement and validate the practicality of an editor-native, agentic writing assistant. More details about this demo and video could be found at https://github.com/PaperDebugger/PaperDebugger.

CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**; • **Computing methodologies** → *Natural language processing*; • **Information systems** → *Information retrieval.*

Keywords

In-editor writing assistance, LLM agents, Multi-agent orchestration, Overleaf integration, Academic writing tools

ACM Reference Format:
Junyi Hou, Andre Lin Huikai, Nuo Chen, Yiwei Gong, and Bingsheng He. 2018. PaperDebugger: A Plugin-Based Multi-Agent System for In-Editor Academic Writing, Review, and Editing. In . ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Large language models (LLMs) are increasingly used in assisting academic writing workflows, from brainstorming and outlining to line-level editing and reviewer-response drafting. Recent systems for human–AI co-writing and assisted feedback demonstrate that LLM-based suggestions can improve fluency and reduce mechanical writing effort at scale [3, 4]. Research on writing-support tools further shows that structured interventions can meaningfully improve writing efficiency and user experience [2, 5, 6].

Despite these developments, the majority of existing tools still operate outside the primary editing environment, requiring copy-and-paste workflows and fragmenting interaction history [3, 7]. This external workflow introduces context switching, breaks writing flow, and makes revision history difficult to preserve. Additionally, external tools provide limited revision provenance; feedback, applied changes, and reasoning disappear once the interaction window closes. Tools such as Writefull provide in-editor language suggestions but remain largely surface-level, offering limited transparency into applied changes [8].

To address these challenges, we present **PaperDebugger**, an in-editor LLM agent system that integrates directly into Overleaf, a widely used academic writing editor. Instead of treating the writing process and model interaction as separate workflows, the system enables critique, refinement, and revision to take place in context, inline, and tied to document structure. The system provides persistent interaction history, patch-based edits, and structure-aware feedback while preserving the continuity of writing. Technically, it is implemented as a Chrome extension that communicates with a Kubernetes-based backend using gRPC. The Model Context Control Protocol (MCP) acts as a lightweight extensibility layer, enabling modular functionality and future agent capabilities without altering the core architecture. PaperDebugger is fully implemented with over 24,000 lines of code.

The current implementation of PaperDebugger has been deployed to real users via the Chrome Web Store [1] and used in live academic writing scenarios. Our demo showcases a complete workflow: authors open a LaTeX project, activate PaperDebugger to request critiques for selected passages, inspect proposed revisions in a diff-style view, and apply accepted patches back into the editor with a single click. In addition to revision workflows, PaperDebugger allows users to invoke MCP-based tools, such as related literature retrieval, directly from the editor, thereby facilitating seamless insertion of relevant, high-quality references. Early analytics based on anonymized telemetry indicate sustained user engagement and active adoption, demonstrating both the technical feasibility and practical value of an in-editor, agentic writing assistant.

To summarize, PaperDebugger makes three key contributions:

- **In-editor academic writing assistance** that integrates directly with Overleaf and operates on selected text, eliminating copy–paste workflows and preserving full writing flow and document context.
- **A scalable multi-agent execution architecture** implemented through Kubernetes-driven pod orchestration, enabling parallel reasoning, structured review, MCP-powered retrieval, AI reviewer, and deterministic diff-based editing.
- **Evidence of real-world usability and adoption**, supported by deployment through the Chrome Web Store and early anonymized telemetry demonstrating repeated use of critique and revision workflows in authentic writing settings.

2 System Overview

2.1 Architecture Overview

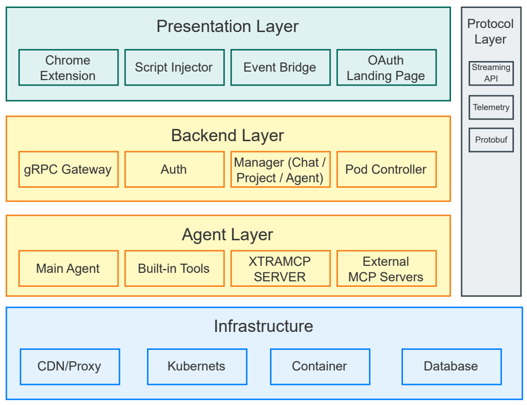


Figure 1: Overall architecture of PaperDebugger, consisting of the presentation layer, backend layer, agent layer, protocol layer and infrastructure.

As shown in Figure 1, PaperDebugger consists of five layers: (1) a presentation layer integrated directly into Overleaf, (2) a backend layer that manages workflow execution, (3) an agent layer running containerized tools, (4) a protocol layer handling structured communication, (5) and an infrastructure layer providing storage and operational services. The presentation layer is implemented as a Chrome extension that injects UI components into Overleaf and synchronizes project context and user actions. The backend layer coordinates authentication, session state, and workflow routing, exposing a streaming interface through a gRPC gateway.

Presentation Layer. PaperDebugger integrates directly into Overleaf via a lightweight Chrome extension. The extension injects a floating panel and inline action buttons next to highlighted LaTeX spans. When users trigger a workflow, the extension captures the selected text and project state, then communicates with the backend using streamable gRPC. Edits are presented as before–after diffs, and accepted patches are applied instantly to the LaTeX source. This elimi nates copy–paste cycles, maintains consistent revision history, and creates a seamless user experience.

Protocol Layer. Communication between the chrome extension and backend leverages a custom message streaming protocol, compatible with OpenAI's server-sent event (SSE) format. This protocol supports real-time streaming of intermediate model outputs, allowing users to receive updates during multi-step workflows.

Backend Layer. The backend is implemented in Go and deployed on Kubernetes. It orchestrates stateless LLM agents, each running inside isolated pods, enabling high concurrency and horizontal scaling. The orchestrator handles routing, model selection, permission checks, and schema validation.

Agent Layer. The agent layer supports two execution modes: *prompt-template agents* and *workflow-based agents*. Prompt-template agents are lightweight, single-shot LLM invocations defined by structured templates. They are designed for low-latency tasks such as grammar polishing. Workflow-based agents are declarative workflows that coordinate multiple LLM calls, tool executions, and validation steps. These workflows handle complex tasks like deep research, relevant paper retrieval, and full-document enhancing.

2.2 Agentic Design

Building on these two execution modes, PaperDebugger extends beyond a single-model rewriting tool through the XtraMCP architecture, a refined variant of MCP tailored for academic writing. XtraMCP exposes a suite of validated tools for literature search, affiliation lookup, and structured data extraction, and enforces our internal Pydantic-based schemas and internal consistency checks to minimize hallucinations. Concretely, three core MCP-powered components back the agents: (i) a low-latency embedding + LLM re-ranking pipeline that provides high-quality semantic retrieval and real-time literature lookup; (ii) a multi-step AI review pipeline, inspired by conference reviewing workflows like AAAI, that guides the Reviewer agent through targeted, segment-level critique; and (iii) XtraGPT [1]. XtraGPT is a model suite tuned for academic writing, ensuring that suggested revisions are context-aware, properly scoped, and phrased in appropriate scholarly style, more details can be found in the paper [1].

On top of this foundation, PaperDebugger runs a suite of specialized agents: a *Reviewer* agent that produces structured critique, an *Enhancer* agent for rewriting and refinement, a *Scoring* agent for clarity and coherence evaluation, and a *Researcher* agent that

[1]https://chromewebstore.google.com/detail/paperdebugger/dfkedikhakpapbfcnbpmfhpklndgiaog

performs literature lookup via XtraMCP tools. For full-document review requests, a coordinating agent decomposes the task into segment-level sub-queries, dispatches them across worker pods, and merges the results into a unified output.

Figure 2 illustrates the execution flow, showing how user actions are routed through the orchestration layer, activate the appropriate agents, and return deterministic diff-based edits back to the editor. We carefully design and validate each agent workflow, and the full prompt templates and agent specifications are available in our project repository.

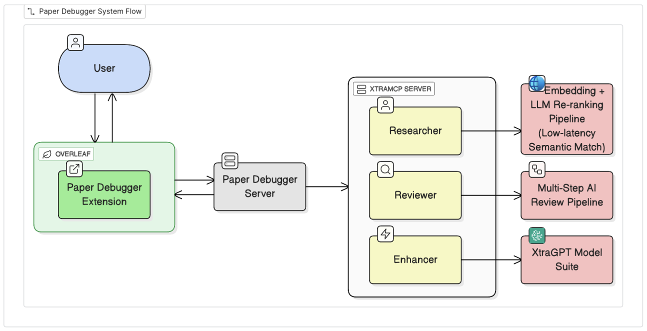

Paper Debugger System Flow
User
OVERLEAF
Paper Debugger Extension
Paper Debugger Server
XTRAMCP SERVER
Researcher
Reviewer
Enhancer
Embedding + LLM Re-ranking Pipeline (Low-latency Semantic Match)
Multi-Step AI Review Pipeline
XtraGPT Model Suite

Figure 2: PaperDebugger end-to-end workflow: The extension captures user actions and sends them to the PaperDebugger server, which coordinates built-in agents or specialized agents on the XtraMCP server.

3 Usage Analytics

We analyze anonymized telemetry from both the Chrome extension and backend between May – November 2025 to understand how PaperDebugger is used in real writing environments. In the following, we present the key statistics, and more usage analysis is in our project repository.

Real-World Adoption. Table 1 shows early usage signals. With 112 extension installs, 78 registered users, and 23 monthly active users, roughly **30% of all users remain active month-to-month**. Users created 158 projects and 797 writing threads, indicating sustained multi-session usage rather than one-off experimentation. User reviews reflect strong satisfaction with the integrated workflow ("convenience", "seamless"). At the same time, feedback highlights natural research value and limitations, such as domain-specific tone ("suggestions feel CS-like") and performance drops on long documents.

Interaction Patterns. Telemetry reveals heavy usage of in-editor revision rather than one-shot generation. Table 2 highlights the three most frequent refinement actions:

Three usage patterns stand out:

- **Users iterate rather than one-shot.** Refinement actions recur within the same session.
- **Patch diffs are the dominant control surface.** Users frequently inspect diffs before applying changes.
- **Sessions contain multiple refinement events.** Interaction density indicates sustained, in-editor revision activity.

Table 1: Early adoption metrics demonstrating real deployment and recurring use.

Metric	Value
Chrome extension installs	112
Registered users	78
Active users (30-day)	23
Projects created (all-time)	158
Threads created (all-time)	797
Chrome store user rating	4.9 / 5

Table 2: Most frequent in-editor refinement operations.

Event Type	Count
Diff viewed	1073
Copy suggestion	375
Insert patch	359

4 Demonstration

This section presents two example in-editor workflows enabled by PaperDebugger. Conference attendees may experiment with additional workflows by installing the PaperDebugger extension directly from the Chrome Web Store and applying it to their own Overleaf projects.

4.1 In-Editor Editing and Patch

A common task in AI-assisted academic writing is to polish or refine text using AI agents. For example, when revising a conference submission, an author may encounter an unclear section title and highlight it directly within Overleaf to request a structured rewrite. As shown in Figure 3, the selected text is forwarded to the PaperDebugger panel, where the author initiates a critique or refinement request.

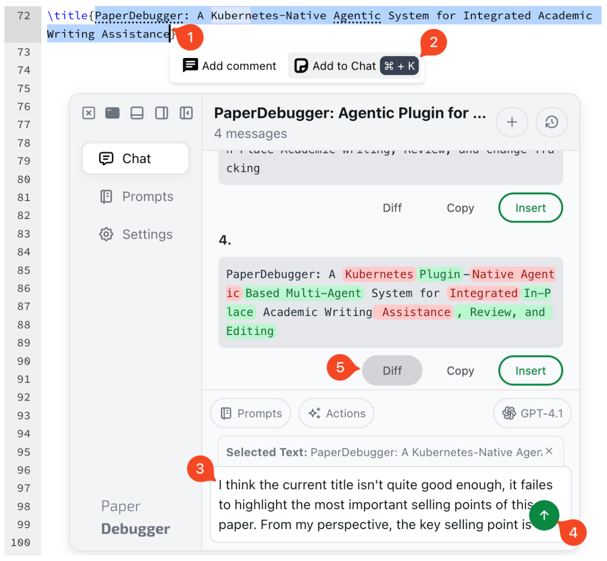
Once invoked, PaperDebugger launches a coordinated agent pipeline comprising a critique agent, an enhancer agent, and a patch generator. The system returns the results as before–after diffs that can be inspected and applied directly within the editor.

All candidate patches are displayed as inline previews. The author examines the rationale behind each suggestion, selects the preferred revision, and applies it with a single click. This workflow replaces the traditional copy-and-paste interaction pattern common to external LLM tools, instead providing a seamless, context-aware editing loop inside Overleaf.

Overall, this case demonstrates how PaperDebugger elevates a simple editing action into a transparent and agentic patch workflow that enhances clarity and structure while preserving an efficient, in-editor writing experience.

4.2 Deep Research and Comparative Analysis

Another common task in academic writing is preparing the related-work section of a manuscript, which requires understanding how the current contribution compares with recent literature. The author highlights the section header and requests "deep research."



```
\title{PaperDebugger: A Kubernetes-Native Agentic System for Integrated Academic
Writing Assistance}
```

n Place Academic Writing, Review, and Change Tra
cking

4.

PaperDebugger: A Kubernetes Plugin-Native Agentic Based Multi-Agent System for Integrated In-Place Academic Writing Assistance, Review, and Editing

Selected Text: PaperDebugger: A Kubernetes-Native Agen

I think the current title isn't quite good enough, it failes to highlight the most important selling points of this paper. From my perspective, the key selling point is

Figure 3: In-editor editing workflow in PaperDebugger. (1) Select a span of LaTeX in Overleaf. (2) Add the selection to the PaperDebugger panel. (3) Specify the critique request. (4) Trigger the agentic pipeline. (5) Review and apply the returned before–after patches.

PaperDebugger invokes an MCP-powered retrieval pipeline that performs multi-stage semantic search over arXiv and curated corpora. Within seconds, the system returns a ranked list of relevant papers enriched with metadata, abstracts, and LLM-generated explanations of relevance.

Upon selecting a target paper, the author activates *Compare My Work*. PaperDebugger automatically extracts key aspects—goals, datasets, methods, evaluation protocols, and limitations—from both papers and produces a structured side-by-side comparison. The system highlights conceptual overlaps, methodological differences, and missing dimensions in the author's draft. A citation-ready summary table is also generated for direct insertion into the manuscript.

For broader situational awareness, the author may conduct an additional search (e.g., "find relevant papers to read"). PaperDebugger aggregates multiple related works into a consolidated research map, revealing clusters such as mitigation-focused systems, benchmarking frameworks, and hybrid evaluation pipelines. The system further generates "takeaways for positioning your work" that help the author articulate how their manuscript fits into the broader landscape.

All outputs of the deep research workflow are delivered directly in-editor, eliminating the need for manual copying of abstracts, comparison tables, or summaries. This scenario demonstrates the emerging capability of research-level reasoning and literature synthesis directly within the writing environment.

5 Conclusion

PaperDebugger provides a unified, in-editor academic writing environment that closes the long-standing gap between document editors and LLM-assisted workflows. By integrating directly with Overleaf through a Chrome-approved extension, the system enables context-aware critique, structured review, literature retrieval, and deterministic diff-based editing. Our real-world deployment demonstrates both feasibility and impact. Early usage analytics show sustained engagement across real writing projects, while case studies highlight how PaperDebugger supports both micro-level refinement and deep research tasks. The system is fully available through the Chrome Web Store, and conference attendees can experience the workflows firsthand by installing the extension and applying it to their own Overleaf documents.

AI Usage Statements: Portions of this manuscript were polished using PaperDebugger, but no scientific content was generated by AI. A complete policy statement and usage disclaimer are available in our project repository.

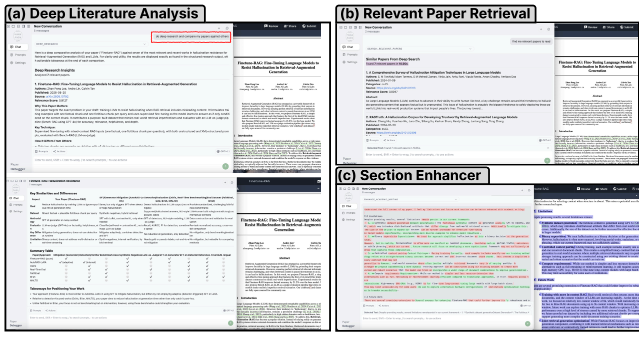


Figure 4: Example of an end-to-end research-use scenario supported by PaperDebugger. The system integrates XtraMCP (a) deep research, (b) related-paper retrieval, and (c) section enhancer to help authors understand, compare, and refine academic content within the editor.

References

[1] Nuo Chen, Andre Lin HuiKai, Jiaying Wu, Junyi Hou, Zining Zhang, Qian Wang, Xidong Wang, and Bingsheng He. 2025. XtraGPT: Context-Aware and Controllable Academic Paper Revision. arXiv:2505.11336 [cs.CL] https://arxiv.org/abs/2505.11336
[2] Colette Ingley and Adrian Pack. 2023. Leveraging AI Tools to Develop the Writer Rather Than the Writing. *Trends in Ecology & Evolution* 38, 8 (2023), 643–645.
[3] Mina Lee and et al. 2024. A Design Space for Intelligent and Interactive Writing Assistants. In *CHI*.
[4] Daniel J. Liebling and et al. 2025. Towards AI-assisted Academic Writing. In *Proceedings of the 1st Workshop on AI and Scientific Discovery: Directions and Opportunities*.
[5] Sheshera Mysore and et al. 2024. PEARL: Personalizing Large Language Model Writing Assistants with Generation-Calibrated Retrievers. In *Proceedings of the 1st Workshop on Customizable NLP*.
[6] Bahareh Sarrafzadeh and et al. 2020. Characterizing Stage-Aware Writing Assistance in Collaborative Document Authoring. In *CSCW*.
[7] Haomin Wen and et al. 2024. OverleafCopilot: Empowering Academic Writing in Overleaf with Large Language Models. *CoRR* abs/2403.09733 (2024). doi:10.48550/arxiv.2403.09733
[8] Writefull Team. 2024. *Writefull: AI Writing Help for Researchers*. https://www.writefull.com/ Accessed: January 2025.