Smart Email Reply: An AI-Powered Context-Aware Gmail Add-On

Praneeth Rangamudri
University of Illinois, Urbana-Champaign
USA
jrang3@illinois.edu

Ziyin Wang
University of Illinois, Urbana-Champaign
USA
ziyin@illinois.edu

Manav Singhai
University of Illinois, Urbana-Champaign
USA
msingh53@illinois.edu

ABSTRACT

We present *Smart Email Reply*, a Gmail add-on that employs Google's Gemini API to generate context-aware, user-tunable reply drafts. Our tool (1) fetches all messages of a thread, (2) lets users select language and tone, (3) appends optional custom instructions, and (4) preserves attachments. We describe its architecture, key algorithms, and evaluate its draft quality, and user satisfaction via small user study.

KEYWORDS

Email AI, Context-Aware Reply, Gmail Add-On, Google Apps Script, Gemini API

1 INTRODUCTION

Modern knowledge workers devote 1–3 hours per day to email composition, often struggling with long threads and canned reply suggestions that ignore conversational context. Existing "smart replies" (e.g., Gmail's one-tap responses) rely on shallow models trained on generic models, yielding replies that feel impersonal or off-topic. Our add-on, *Smart Email Reply*, addresses these issues by integrating all the messages of a thread, letting users choose language and tone, and enabling custom instructions, all while preserving attachments. This empowers users to generate drafts that match their personal style and the discussion's intent, reducing drafting time.

2 RELATED WORKS

There has been substantial research in automated email and dialogue response generation, focusing on improving contextual relevance and language fluency. Google's Smart Reply system uses LSTM-based models to provide rapid, concise suggestions. Subsequent work by Sharma et al. employed BERT-based architectures to better model context for more nuanced generation[2]. Dialogue systems like TransferTransfo further demonstrated how transformers can maintain coherence across multiple conversational turns[1, 3]. However, these tools have 3 major issues: 1. Lack of support for long-form responses 2. Minimal user control over tone and intent 3. Limited integration into real-world email workflows Our tool addresses these limitations by enabling flexible, user-guided generation of personalized email responses that enhance both productivity and communication quality.

3 MOTIVATION / INTENDED USERS

Knowledge workers—whether graduate students juggling TA responsibilities, researchers coordinating multi-partner projects, or professionals negotiating time-sensitive contracts—spend an estimated one to three hours per day drafting and refining email replies. Yet despite sophisticated inbox management features like filters, labels, and search, the core task of composing replies remains manual, repetitive, and error-prone: users must skim long, nested threads to reconstruct context; decide on an appropriate tone and language; and remember to reattach critical files, all before hitting "Send." Existing "smart reply" solutions, while convenient, offer only generic, context-free suggestions that seldom align with the thread's history or the user's personal style, forcing users to copy, paste, and heavily edit canned text—or start replies from scratch. This friction not only slows down communication but also risks miscommunication when nuances are lost. Our motivation, therefore, is to streamline this process by embedding an AI-driven assistant directly into Gmail: one that automatically gets context of a thread, preserves attachments, and lets users select language (e.g., English, Spanish, French, Chinese), tone (formal or casual), and even append custom instructions like "Reference the attached draft's Section 4" or "Keep it under three sentences." Intended users include anyone managing moderate to high volumes of email—TAs providing personalized feedback, researchers responding to collaborators, customer-facing professionals, and multilingual teams—who need to maintain clarity, consistency, and personal voice without sacrificing efficiency. By reducing drafting time and ensuring replies remain contextually rich and stylistically on-brand, Smart Email Reply empowers users to focus on substantive work rather than boilerplate text.

4 SYSTEM ARCHITECTURE

As shown in Figure 1, the system architecture of Smart Email Reply is organized into three tightly integrated layers—UI, logic, and API—each implemented within Google Apps Script to deliver a seamless, responsive Gmail add-on. When a user opens an email thread, Gmail invokes our `generateReply(e)` trigger, which lives in the UI layer and marshals the CardService API to construct a dynamic sidebar card. This card displays a conversation preview—extracted by iterating over all the messages via `GmailApp.getMessageById(...)` and truncating each to eighty characters—alongside dropdowns for language and tone selection, a multiline text field for custom prompts, and a checkbox list of any attachments discovered by `getAttachments({includeInlineImages:false})`. Once the user chooses their options and clicks "Generate Reply," control passes to the logic layer's `regenerateReply(e)` function. Here, the same preview rebuilding occurs, user inputs are read from `e.formInput`, and a composite prompt is assembled: a base instruction ("Write one ‹tone› reply in ‹language› to this thread") concatenated with

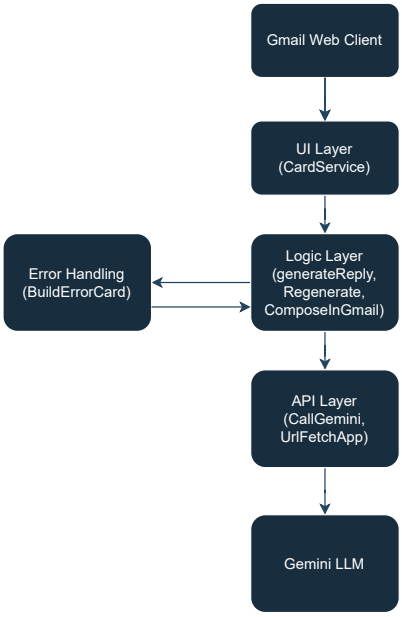

Figure 1: High-level architecture of the Gmail add-on.

the preview and, if present, the user's additional instructions, each separated by clear markers. That prompt is then handed to the API layer's `callGemini(prompt)`, which uses `UrlFetchApp.fetch()` to POST a JSON payload to Google's generativelanguage endpoint, inserting the stored `GEMINI_KEY` from `PropertiesService` and parsing the JSON response to extract the generated text. This suggestion is fed back into the UI layer by rebuilding the card with the new draft in a `TextInput`, updating the header subtitle to indicate whether custom instructions were appended. When the user opts to "Compose in Gmail," the `composeInGmail(e)` handler converts the draft into both plain and HTML bodies (preserving line breaks via `escapeHtml()`), reattaches any selected blobs, and calls `message.createDraftReply(...)` to open a native Gmail draft. Throughout all layers, robust error handling is provided by wrapping each entry point in `try. . . catch` blocks that invoke a shared `buildErrorCard(message)`helper; this constructs a minimal CardService card displaying the error and allows users to retry without crashing the add-on. By centralizing UI concerns in CardService, encapsulating state and orchestration in the logic layer, and isolating external HTTP interactions in the API layer, Smart Email Reply achieves modularity, ease of maintenance, and a fluid user experience that integrates generative AI directly into everyday email workflows—all within a single Apps Script project.

5 KEY ALGORITHMS

5.1 Threads Extraction

```
// Extracts full thread for the LLM
var fullThread = '';
    for (var i = 0; i < msgs.length; i++) {
      var m    = msgs[i];
      var body = m.getPlainBody().replace(/\r?\n/g
          , ' ');
      fullThread += m.getFrom() + ': ' + body + '\
          n\n';
    }
```

5.2 Prompt Assembly

```
var basePrompt =
  'You are an intelligent email assistant. ' +
  'Write one ' + tone.toLowerCase() +
  ' reply in ' + lang +
  ' to this thread:\n\n' + preview;
if (custom) {
  prompt = basePrompt + '\n\nAdditional
      instructions: ' + custom;
} else {
  prompt = basePrompt;
}
```

The algorithm shown above constructs a dynamic prompt that guides the language model's response generation. It begins by setting a base prompt that instructs the model to act as an intelligent email assistant. The prompt is further customized based on two key parameters: the tone and language preferences specified by the user. Additionally, it includes a brief preview of the most recent messages in the email thread to provide relevant conversational context. If the user provides any extra custom instructions (e.g., "keep it concise" or "use bullet points"), these are appended to the prompt as an additional instruction block. This modular structure ensures the model receives both standardized and user-tailored directives, enabling it to generate replies that are coherent, context-aware, and stylistically aligned with user expectations.

6 IMPLEMENTATION DETAILS

The prototype developed is a Gmail Add-on built using Google Apps Script that enhances email productivity by generating intelligent, context-aware replies. The core functionality is triggered through a contextual event when a Gmail thread is opened. It extracts all the messages through Gmail Api and constructs a prompt for the Gemini 2.0 Flash language model via a REST API. Users are presented with a dynamic interface built using CardService, allowing them to customize the AI-generated reply based on tone, sentiment, response type, response length, and additional instructions. These inputs directly influence the prompt passed to the model, enabling fine-grained control over the assistant's response style. The AI-generated suggestion is then either shown within the add-on or used to create a draft using Gmail's native compose interface. The user may also choose to send the reply directly to the sender of the first message in the thread. API calls to Gemini are managed through UrlFetchApp, with robust error handling and logging provided for debugging and transparency. Sensitive credentials such as the API key are securely managed via Script Properties. Overall, this implementation demonstrates how modern language models

can be integrated into everyday productivity tools like Gmail to streamline communication, offering a customizable and seamless user experience.

7 EVALUATION

We conducted an iterative evaluation of the Smart Email Reply add-on through informal testing with 10 users, including friends and roommates. Each participant used the plugin within their own Gmail accounts, responding to a set of test emails. All users confirmed that the generated replies were contextually relevant and of high quality. Feedback was gathered through direct observation and follow-up conversations. To explore the system's flexibility, participants were encouraged to experiment with various tones and response types across multiple sessions. 90 percent of our users agreed that the plugin generated correct and context aware responses.

8 USAGE GUIDE

Installation.

(1) Open the link to the project Script. (The link is in the Github repo shared in this paper))
(2) Github repo - https://github.com/Manav020201/SmartEmailReply
(3) Under `Project → Properties`, set `GEMINI_KEY`.
(4) Click on the Deploy Button on top right , then click Test Deployments
(5) Click Install Add-on.
(6) Head to your gmail account and grant scopes to start using the plugin

Workflow. (Look for Figure 2)

- Open any thread → click "Smart Email Reply" in the sidebar.
- Review the preview, select language/tone, optionally add instructions.
- Click "Generate Reply" to see AI draft, then "Compose in Gmail"

9 LIMITATIONS

Our system encountered noticeable latency when handling long email threads, especially in the absence of a summarization layer. While we chose to preserve the full conversational narrative by using raw thread data, this approach may not scale well under frequent or large-scale usage. As we expand to support more users, maintaining responsiveness across threads of varying lengths will require more efficient input processing or the introduction of summarization strategies.

A second limitation lies in our dependence on Gemini's general-purpose language model. Despite offering user-facing controls such as tone, sentiment, and response type, the generated replies can still come across as overly formal or verbose. In such cases, users often need to intervene manually. Although the assistant learns gradually across sessions, the underlying memory logic is not exposed, limiting transparency into what gets retained and for how long.

Additionally, the project is bounded by the constraints of Google Apps Script. While it facilitates native Gmail integration, it imposes limitations on how memory, backend processes, and user interface

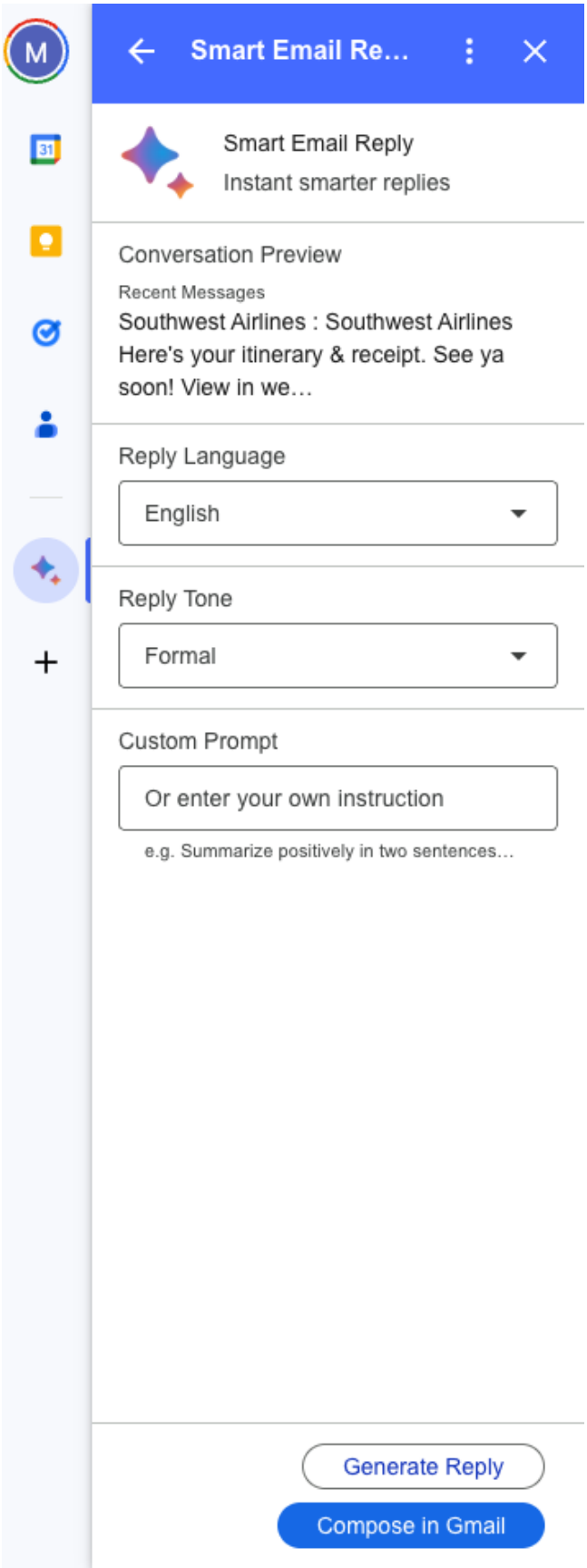

Figure 2: UI of Gmail Plugin

elements can be managed. Our ability to design rich or dynamic UI components is restricted, which in turn limits the overall interactivity and customization of the assistant.

10 FUTURE WORK

As our system scales to support a growing user base, managing long-term context efficiently becomes increasingly important. Right now, all user-specific memory lives within Gemini's internal chat sessions. This setup offers lightweight personalization without needing manual setup, but it also ties us to the model's fixed and opaque memory limits.

As more users begin relying on the assistant regularly, the volume of contextual data per user will grow. Depending solely on Gemini to store and manage this history may eventually pose challenges in terms of performance, reliability, and cost. To future-proof the system, we plan to integrate an external storage layer—such as a lightweight database—to store key user traits like tone preferences, recurring reply patterns, and conversation summaries.

By fetching this data at runtime, we can dynamically enrich prompts for each user. This not only gives us more control over memory retention but also ensures consistent, personalized responses—even when chat sessions restart—helping the assistant stay dependable at scale.

11 CONCLUSION

We built and evaluated *Smart Email Reply*, a Gmail add-on that significantly reduces drafting time and improves user satisfaction by combining thread context, tone/language selection, and custom prompts. Our evaluation demonstrates strong user preference and low latency. In future work, we'll expand context coverage and personalize models per user.

12 CONTRIBUTIONS

All three of us met weekly to work on this project. First we worked on the arcihtecture of our plugin using Appscript. Once we established a robust pipeline, we moved towards prompt engineering to ensure the generated emails we were getting were suitable to context. Then we all tested our gmail pluggin feature with other people. Lastly, for the report we all discussed together what to include each part as well as finding apporpriate references based on our work.

REFERENCES

[1] Mohammed Munzer Dwedari, Matthias Niessner, and Dave Zhenyu Chen. 2023. Generating Context-Aware Natural Answers for Questions in 3D Scenes. arXiv:2310.19516 [cs.CV] https://arxiv.org/abs/2310.19516

[2] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufman, Balint Miklos, Greg Corrado, Andrew Tomkins, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated Response Suggestion for Email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (2016).* https://arxiv.org/pdf/1606.04870v1.pdf

[3] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. arXiv:1901.08149 [cs.CL] https://arxiv.org/abs/1901.08149