Instituto Superior de Engenharia
Politécnico de Coimbra

DEPARTMENT OF SYSTEMS AND COMPUTER ENGINEERING

# Development of a Generative AI-Based Professional Assistant with a Custom Personality

Bachelor Final Project in Informatics Engineering

Author

**Rodrigo José Carraça Novelo**

Supervisor

**Prof. Jorge Bernardino (Polytechnic University of Coimbra, Portugal)**

Co-Supervisor

**Prof. Rodrigo Rocha Silva (São Paulo Technological College, Mogi das Cruzes, Brasil)**

Coimbra, September 2025

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

# Resumo

A crescente disseminação dos Grandes Modelos de Linguagem (LLMs) oferece um potencial significativo para a automatização de tarefas de comunicação. No entanto, a sua aplicação na escrita de emails resulta frequentemente em respostas genéricas, que carecem da autenticidade, do estilo pessoal e da fidelidade contextual essenciais a uma comunicação eficaz. Assim sendo, o presente relatório aborda esta lacuna e descreve a conceção, implementação e validação de um sistema de automatização de email, cuja arquitetura, centrada numa ontologia, procura superar a dicotomia entre a redação de texto artificial e uma comunicação genuinamente personalizada.

O elemento central desta proposta é a Ontologia de Personalização, uma arquitetura cognitiva multi-camada que funciona como o "núcleo cognitivo" do assistente de Inteligência Artificial (IA). Esta ontologia modela a identidade comunicacional do utilizador em três níveis: um Núcleo Estável, que define os princípios estilísticos e traços de personalidade imutáveis; uma camada de Consciência Social, que adapta a comunicação ao contexto relacional através da gestão de perfis detalhados de interlocutores; e uma Mente Evolutiva, que assegura a aprendizagem contínua do sistema e infere novas regras e preferências a partir das correções do utilizador, num ciclo de Human-in-the-Loop.

Para consultar esta base de conhecimento de forma eficiente e em tempo real, foi desenvolvido um motor de busca híbrido e paralelo. Este motor combina a precisão da pesquisa lexical com a profundidade contextual da pesquisa semântica, baseada em "vector embeddings". A arquitetura do sistema concretiza-se em dois modos de operação distintos: um modo manual, que atua como um assistente de co-escrita colaborativo, e um modo automatizado, que opera como um agente proativo e autónomo.

A validação do protótipo, realizada através de uma série de estudos de caso qualitativos, demonstrou a eficácia desta abordagem orientada pela ontologia. Os resultados confirmam a capacidade do sistema para gerar comunicações estilisticamente alinhadas com o utilizador, factualmente precisas e contextualmente adequadas. Assim, esta descoberta vem validar a hipótese central deste trabalho: a incorporação de uma base de conhecimentos estruturada e explícita é um mecanismo fundamental para orientar os LLMs na produção de uma comunicação autêntica e personalizada.


**Palavras-chave:** Grandes Modelos de Linguagem (LLMs), Personalização, Engenharia Ontológica, Recuperação de Informação Híbrida, Geração Aumentada por Recuperação (RAG), Automatização de Emails.

## Abstract

The increasing proliferation of Large Language Models (LLMs) offers significant potential for automating communication tasks. However, their application in email writing often results in generic responses that lack the authenticity, personal style, and contextual fidelity essential for effective communication. This report addresses this gap by describing the design, implementation, and validation of an email automation system whose ontology-centered architecture seeks to overcome the dichotomy between artificial text generation and genuinely personalized communication.

The central element of this proposal is the Personalization Ontology, a multi-layered cognitive architecture that functions as the "cognitive core" of the AI assistant. This ontology models the user's communicational identity on three levels: a Stable Core, which defines immutable stylistic principles and personality traits; a layer of Social Awareness, which adapts communication to the relational context by managing detailed interlocutor profiles; and an Evolving Mind, which ensures the system's continuous learning by inferring new rules and preferences from user corrections in a Human-in-the-Loop cycle.

To query this knowledge base efficiently and in real-time, a parallel hybrid search engine was developed. This engine combines the precision of lexical search with the contextual depth of semantic search, which is based on vector embeddings. The system's architecture materialized in two distinct operating modes: a manual mode, which acts as a collaborative co-writing assistant, and an automated mode, which operates as a proactive and autonomous agent.

The validation of the prototype, conducted through a series of qualitative case studies, demonstrated the effectiveness of this ontology-driven approach. The results confirm the system's ability to generate communications that are stylistically aligned with the user, factually accurate, and contextually appropriate. Therefore, this finding confirms the central hypothesis of this research: that incorporating a structured and explicit knowledge base is a key mechanism for steering LLMs to produce more authentic and personalized communication.

**Keywords:** Large Language Models (LLMs), Personalization, Ontological Engineering, Hybrid Information Retrieval, Retrieval-Augmented Generation (RAG), Email Automation.

# Agradecimentos

Dou os meus agradecimentos e reconhecimento ao ISEC e aos orientadores do projeto, Prof. Jorge Bernardino e Prof. Rodrigo Rocha Silva. Gostaria também de expressar profunda gratidão à minha família. Aos meus pais e ao meu irmão, o vosso apoio constante ao longo deste último semestre foi indispensável e o vosso incentivo desempenhou um papel crucial na conclusão deste projeto.

## Acknowledgements

I give my gratitude and acknowledgement to ISEC and the project supervisors, Prof. Jorge Bernardino and Prof. Rodrigo Rocha Silva. Also, I would like to extend heartfelt gratitude to my family. To my parents and my brother, your constant support throughout this final semester was indispensable, and your encouragement played a crucial role in the completion of this project.

# Index

## List of Abbreviations

AI: Artificial Intelligence
API: Application Programming Interface
BFF: Backend for Frontend
BPE: Byte-Pair Encoding
DITTO: Demonstration Iterated Task Optimization
DPO: Direct Preference Optimization
DRY: Don't Repeat Yourself
FA: Feedback Acceptance
FIFO: First-In, First-Out
GenAI: Generative AI
HITL: Human-in-the-Loop
IR: Information Retrieval
KG: Knowledge Graph
LLM: Large Language Model
LoRA: Low-Rank Adaptation
NPU: Neural Processing Unit
ORM: Object-Relational Mapper
PEFT: Parameter-Efficient Fine-Tuning
PIT: Implicit Self-Improvement
PR: Pull Request
QA: Question-Answer
RAG: Retrieval-Augmented Generation
RLHF: Reinforcement Learning from Human Feedback
ROPE: Rotary Positional Encoding
RR: Response Rate
SFT: Supervised Fine-tuning

## List of Tables

# List of Figures

# 1. Introduction

In the modern digital landscape, email remains the cornerstone of professional and personal communication. However, the sheer volume of daily correspondence presents a significant challenge, leading to decreased productivity and communication fatigue. While the advent of Large Language Models (LLMs) has introduced a new generation of AI-powered writing assistants, many of these tools offer generic, one-size-fits-all solutions. They often fail to capture the nuanced complexities of individual writing styles, adapt to specific relational contexts, or leverage personal information in a secure and relevant manner. This results in responses that, while grammatically correct, can feel impersonal, lack authenticity, and require significant manual revision to align with the user's authentic voice.

The central problem this project addresses is the gap between generic text generation and accurate personalized communication. Existing solutions are often limited in their ability to dynamically integrate a user's stylistic preferences, factual knowledge, and relationship-specific communication protocols. This limitation stems from a lack of a centralized, expressive framework that can guide the LLM's generation process with a high degree of precision.

To overcome these challenges, this report presents the design and implementation of a novel, ontology-driven email automation system. At the core of this system is a comprehensive domain ontology that serves as a "persona-in-a-file," meticulously defining a user's communication style, storing personal and professional knowledge, and encoding specific rules for interacting with different contacts. This structured knowledge base allows the system to engineer highly contextualized prompts for an underlying LLM (in this case, Google's Gemini), resulting in email drafts that are not only accurate but also authentically reflect the user's persona.

The developed solution features a dual-mode architecture: a manual, web-based interface for guided email composition and a fully automated workflow that processes incoming emails in the background, generates draft responses, and awaits user approval via push notifications. Furthermore, the system incorporates a feedback mechanism, enabling it to learn from user corrections and iteratively refine its internal rule-based knowledge. This report will detail the theoretical foundations that informed the project, describe its architecture and implementation, and discuss the results, providing a comprehensive overview of a system designed to bridge the gap between artificial intelligence and genuinely personal communication.

The main contributions of this work are the following:

- The design, implementation, and validation of an email automation system centered on a three-layered personalization ontology: a Stable Core, a Social Awareness layer, and an Evolving Mind.
- The development of a hybrid and parallel search engine that combines lexical and semantic search to retrieve information from a defined Ontology.

- The validation of the system through qualitative case studies confirmed its ability to generate communications that are stylistically aligned with the user, factually accurate, and contextually appropriate.

The rest of this report is structured as follows:

- Chapter 2: Background and Theoretical Foundations - Details the personalization challenge, LLM concepts, and the theoretical framework of the project.
- Chapter 3: Related work - Presents a literature review on LLMs, email generation, personalization techniques, and associated challenges.
- Chapter 4: System Architecture and Design Rationale - Discusses the system's three-layer architecture (presentation, logic, and data) and its design rationale.
- Chapter 5: The Personalized Ontology - Describes the personalization ontology, including the data model and the layered architecture.
- Chapter 6: Intelligent Knowledge Retrieval: A Hybrid Search Architecture - Explores the architecture of the hybrid search engine, which combines lexical and semantic search for knowledge access.
- Chapter 7: Implementation of Operational Workflows - Describes the implementation of the manual and automated workflows that orchestrate the system's actions.
- Chapter 8: Implementation: Bridging Concepts and Code - Details of the technical implementation, including the technologies used and the realization of the cognitive core.
- Chapter 9: Experimental Validation - Presents the experimental validation methodology and the results of qualitative case studies to prove the concept.
- Chapter 10: Conclusions and Future Work - Summarizes the key findings of the project, reflects on its limitations, and proposes promising avenues for future research and development.

# 2. Background and Theoretical Foundations

Although the specific development of AI assistants tailored exclusively for generating email responses remains relatively underexplored in academic literature, the impressive capabilities of Large Language Models (LLMs) are becoming more apparent. These models understand context, generate human-like text, and adapt tone based on input prompts, suggesting a natural alignment with email-related tasks. This chapter will first detail the core challenge of personalization and the foundational concepts of LLMs. It will then introduce the advanced theoretical framework that provides the academic grounding for this project's novel approach to solving these challenges.

## 2.1 Foundational Concepts in LLMs and Email Generation

The sophisticated functionality of LLMs relies on several interconnected core components. First, tokenization methods (like Byte-Pair Encoding, or BPE) are used to break down raw input text into smaller, manageable numerical units (tokens) that the model can process [1]. Because the transformer architecture processes these tokens in parallel, Positional Encoding techniques (such as absolute, relative, or Rotary Positional Encoding - RoPE) are crucial for injecting information about the original sequence order of the tokens. The fundamental Attention Mechanisms, particularly self-attention, then enable the model to dynamically calculate the relevance of each token to all other tokens in the sequence, allowing it to focus on the most pertinent information. Within the network's layers, non-linear Activation Functions (e.g., GeLU, SwiGLU) are essential for enabling the model to learn complex patterns and relationships in the data [1]. Finally, Layer Normalization techniques (e.g., RMSNorm) are applied at various points to stabilize the learning process, ensuring that the activations within the network remain in a suitable range during training. The specific arrangement and configuration of these components define the model's overall architecture. Typical structures include Decoder-Only models (like the GPT series) which are autoregressive and excel at text generation; Encoder-Only models (like BERT) which process the entire input bidirectionally for deep understanding; and Encoder-Decoder models (such as T5) suited for tasks transforming input sequences to output sequences, such as translation and email answering [2], [3].

The scale of LLMs, often involving hundreds of billions of parameters, enables emergent capabilities such as in-context learning, instruction following, and multi-step reasoning. Tokenization methods and positional encoding strategies (e.g., rotary embeddings) are crucial components of their architecture, while post-training techniques such as Instruction Tuning and Reinforcement Learning from Human Feedback (RLHF) enhance alignment with human preferences [4]. In the context of email generation, these advancements have evolved into Transformer-based systems, and later innovations have integrated personalization strategies such as fine-tuning, prompt engineering, and retrieval-augmented generation (RAG), enhancing the relevance and coherence of responses while addressing challenges like multi-turn thread management and stylistic alignment. Together, these foundational developments underpin the growing potential of LLMs in personalized, efficient, and context-aware email response generation systems, which will be explored further [2], [3].

## 2.2 The Need for Personalization in Email Communication

Email serves as a distinct and essential communication channel, characterized by specific norms of formality, task-orientation, and interpersonal nuance that set it apart from other digital platforms. While Large Language Models (LLMs) exhibit strong capabilities in general-purpose text generation, their inefficient forms often struggle to meet the subtle and context-sensitive demands of email communication [[4]. Early AI tools, such as Smart Reply [5] and Smart Compose [6], showcased the promise of AI-assisted email generation but also revealed the limitations of a one-size-fits-all approach, underscoring the need for more sophisticated, user-centric adaptation.

The core challenge is not only generating grammatically correct text but also accurately capturing an individual's unique communication style and intent within the immediate context of an email thread. Research in conversational AI indicates that conditioning models on user personas or profiles significantly improves consistency and relevance [7]. In the email domain, this requires personalization along several dimensions: aligning with the sender's characteristic tone and vocabulary, modulating formality levels, maintaining coherence across asynchronous threads, and incorporating relevant contextual cues [8]. Achieving this kind of alignment is essential for generating emails that feel authentic and effective, but it remains a significant hurdle.

Implementing robust personalization also presents notable technical and ethical challenges. One major obstacle is scalability: training and maintaining a distinct large model for each user is typically infeasible [8]. Parameter-Efficient Fine-Tuning (PEFT) techniques, such as adapter modules [9] and Low-Rank Adaptation (LoRA)[10], offer promising solutions by enabling efficient adaptation with minimal parameter overhead. Collaborative strategies, such as PER-PCS, further explore gains by sharing and composing PEFT components across users [11].

User privacy is another critical concern, as effective personalization often depends on access to sensitive historical email data [8]. Retrieval-Augmented Generation (RAG) [12] offers a more privacy-preserving alternative, allowing LLMs to retrieve relevant content from external, potentially local, knowledge sources (e.g., a personal email archive) at inference time. This enables contextually appropriate generation without requiring the model to internalize private data. Optimizing retrieval pipelines

[13] and integrating RAG with local PEFT are key strategies for striking a balance between personalization and privacy.

Moreover, personalization must account for the evolving nature of user preferences and communication styles [14]. This requires models capable of continual, context-aware learning based on real-time interactions, rather than relying on static user profiles. Interaction design also plays a crucial role; moving beyond plain-text prompts to structured question-answering or guided interfaces may provide more effective ways for users to steer the personalization process.

Taken together, these factors underscore the urgent need for more effective personalization in email generation. Techniques such as PEFT and RAG bring valuable strengths in style adaptation, privacy preservation, and computational efficiency [14]. Still, key challenges remain. Fine-tuning on limited user data often falls short of capturing the full range of real-world email contexts, especially when dealing with multiple personas or varied communicative goals. Likewise, applying RAG to extensive, unstructured personal archives requires highly accurate and context-sensitive retrieval—something current systems can only partially deliver. The main goal is not merely to replicate a user's stylistic voice, but to reflect their current knowledge state and communicative intent in each message. Bridging the gap between surface-level personalization and deep contextual understanding, while maintaining scalability and reliability, remains a central challenge in building truly effective and trustworthy AI email assistants.

## 2.3 Theoretical Framework

While LLMs provide the raw generative capability, a principled theoretical framework is required to transform them into reliable, personalized agents. The core challenge lies in addressing their tendency for "wildly confabulated responses (usually called hallucinations)," which compromises their utility and makes them unreliable for specialized tasks. A state-of-the-art solution involves constructing a cognitive architecture built upon a synthesis of established concepts from computer science. Such a framework provides the AI assistant with a stable "brain" for knowledge storage, a sophisticated "perception" system for information retrieval, and a controlled mechanism for "action" and "learning" [15].

The foundation of this architecture—the AI's brain—is grounded in the discipline of Ontological Engineering. An ontology is a formal computational artifact that serves as the schema for a Knowledge Graph (KG). A KG is a structured, multi-relational graph composed of entities and relations that acts as a system's long-term memory [16]. By providing an external, verifiable source of truth, the KG serves as the primary "guiderail to limit or prevent confabulation" and escape LLM hallucinations. This structured, rule-governed knowledge base is fundamental to guiding LLMs in the production of authentic and personalized communication.

This cognitive "brain" requires an intelligent "perception" system to access its knowledge in real-time, a domain addressed by modern Information Retrieval (IR). Traditional lexical search, which relies on statistical methods like BM25, is efficient and straightforward but is fundamentally limited by the "vocabulary mismatch problem"; it cannot grasp a query's underlying semantic meaning and fails when exact keywords are not present. Conversely, semantic search utilizes dense vector embeddings to capture the sense of text, allowing it to measure similarity based on context rather than exact word matches. While powerful, semantic search can sometimes lack precision for specific factual queries. The state-of-the-art solution is a hybrid search architecture that fuses these two complementary methods. By combining the keyword precision of lexical search with the deep contextual understanding of semantic search, a hybrid engine consistently outperforms either technique in isolation, providing the cognitive architecture with the most accurate information possible [17], [18].

Finally, the architecture needs a mechanism for "action" and "learning" that connects perception to response. Retrieval-Augmented Generation (RAG) is the key architectural pattern that enables this. RAG enhances LLMs by integrating external retrieval capabilities, allowing them to generate responses based on timely, verified information that was not explicitly trained on. The assistant acts upon the world by using information "perceived" by the hybrid retrieval engine to generate an accurate, context-aware response. This process is carefully controlled via Prompt Engineering, the discipline of crafting precise instructions to guide the LLM's output. Systems can then learn and adapt through a Human-in-the-Loop (HITL) cycle, which is often realized via Self-Refine Prompting. This project implements HITL via a feedback mechanism where the LLM is prompted to analyze user corrections and generate new, generalized rules for its own knowledge base. This creates a virtuous cycle of improvement, allowing the system to evolve and continuously refine its understanding based on user interactions [17].

# 3. Related Work

To situate the developed system within the current academic and technological landscape, a comprehensive review of the scientific literature was conducted. A systematic search was conducted across academic databases, including Google Scholar, IEEE Xplore, and the ACM Digital Library, prioritizing recent and relevant publications, which resulted in a final selection of 26 key papers. This review focused on the impact of large language models on email response generation, personalization techniques, and the associated challenges. The analysis of these papers helped to identify the foundational technologies, emerging trends, and critical gaps that this project aims to address. As shown in Table 3.1, the findings were organized thematically, covering Tools and Frameworks, Personalization Techniques, Security Vulnerabilities, User Perceptions, and Benchmarking.

Table 3.1 - Thematic Grouping of Reviewed Papers

| Paper | Topics |
|---|---|
| [19],[20],[21],[22],[23],[24],[25],[26] | Tools and Frameworks |
| [27],[28],[29],[30],[31] | Personalization Techniques |
| [32],[33],[34],[35] | Security Vulnerabilities |
| [36],[37],[38],[39],[40] | User Perceptions |
| [41],[42],[43],[44] | Benchmarking and Evaluation |

## 3.1 Tools and Frameworks to Enhance AI-Powered Writing Assistants

The creation of writing assistants based on AI, particularly in tasks such as composing personalized emails, is motivated primarily by the groundbreaking tools and architectures that underlie their development. The following is an extensive overview of the key systems and architectural frameworks developed to advance various aspects of AI writing, including promoting creative questioning, providing high-quality tutorial data, and enhancing human-AI interaction. It is crucial to understand these underlying developments to value how Large Language Models (LLMs) are being modified and used to meet the requirements of customized email communication.

One of the key innovations in enhancing AI-powered writing assistants is the Luminate system [19], which introduces a unique approach to creative exploration, overcoming the limitations of traditional prompt-based methods. Conventional systems often suffer from premature convergence and a limited exploration of creative possibilities, resulting in restricted output diversity. In response, the authors introduce a novel Prompting for Design Space framework. This system enhances idea generation by employing a two-step process: first, generating key dimensions relevant to the user's prompt and then using these dimensions to guide the creation of a diverse array of responses. This framework facilitates a structured exploration of the creative design space, enabling users to engage with and actively organize the generated content. Its key innovations include the automated generation of creative dimensions, dimension-guided response generation to promote output diversity, interactive selection of dimensions to help structure responses, and semantic zooming to explore content at varying levels of detail. Unlike traditional prompt refinement methods used in systems like ChatGPT, these features offer a systematic and expansive approach to creative exploration.

While Luminate [19] focuses on enhancing creative exploration, other research has addressed improving instruction data quality for more accurate AI writing assistance. REALIGN [20] presents a novel solution to improve the quality and alignment of instruction data for large language models (LLMs), addressing a key challenge in developing generative AI applications, such as email writing assistants. Traditional fine-tuning methods often rely on extensive human annotation or suffer from inaccuracies, such as hallucinations, which limit their effectiveness. REALIGN addresses these issues through a three-step process—Criteria Definition, Retrieval Augmentation, and Reformatting—that improves the structure and factual accuracy of LLM outputs. The approach begins by incorporating human-defined response formats, ensuring better alignment with user expectations, particularly for tasks like email generation. By leveraging the Google Search API to gather relevant, up-to-date information, it enhances the factual accuracy of generated content, reducing the risk of errors. This retrieval-based step also improves scalability, enabling LLMs to access external knowledge without relying solely on pre-existing training data. The final step, reformatting, ensures that the outputs are structured, readable, and meet user-defined criteria.

Empirical results demonstrate that REALIGN [20] significantly improves overall alignment, mathematical reasoning, factuality, and readability in LLMs. For example, the mathematical reasoning ability of LLaMA-2-13B on the GSM8K dataset was enhanced from 46.77% to 56.63% solely through reformatting, without requiring additional data or advanced training. Moreover, just 5% of REALIGN data led to a 67% improvement in overall alignment, as measured by the Alpaca dataset. This highlights the importance of organized formats, particularly for tasks that require complex reasoning and analysis. This framework also uses a task classifier to apply the correct format based on the query type, while post-processing steps, such as length and task-based filtering, ensure the generation of high-quality data.

Taking a different approach to user engagement, Miura et al. developed ResQ [21]. This artificial intelligence-based email response system replaced the traditional prompt-based drafting with a question-answer (QA) method. The new technology solved one of the prevalent problems with email support systems: helping users articulate their responses clearly. The system automatically retrieved relevant data from incoming emails and created structured questions for users to optimize responses. This approach reduced the cognitive burden of manual prompt engineering, improving response accuracy and user satisfaction. A comparative experiment with 20 users revealed that ResQ improved response efficiency without compromising email quality, outperforming conventional LLM-based email composition tools.

Beyond the quality of instruction data, effective interfaces for managing multiple AI-generated variations represent another crucial advancement in this field. The work presented in [22] makes a significant contribution to generative AI for writing assistance by addressing the critical need for interfaces that effectively support the exploration and organization of multiple writing variations generated by large language models. Recognizing the limitations of existing chat-based and in-place editing interfaces in handling the growing volume of AI-generated text, the authors propose ABScribe [22]— a novel interface designed to support and enhance human–AI co-writing. This system aims to improve collaboration by providing more effective methods for managing and interacting with generated content. The technical approach involves the implementation of five integrated interface elements: Variation Fields for non-linear, in-place storage of multiple variations; a Popup Toolbar for swift comparison; a Variation Sidebar for structured navigation; AI Modifiers that transform LLM instructions into reusable buttons; and an AI Drafter for direct text insertion. To evaluate the practical effectiveness of ABScribe, the study employed a within-subjects design with 12 writers who completed guided writing tasks (LinkedIn post and email) using both ABScribe and a baseline interface featuring a chat-based AI assistant. Unique data collection methods included the NASA-TLX questionnaire to assess subjective task workload and Likert-scale measures to display user perceptions of the revision process.

Building on the foundational work summarized in the literature review by Rasheed et al. [22], recent research has introduced novel methods for incrementally improving AI-generated content through self-feedback mechanisms. The paper presented in this context introduces SELF-REFINE [23]. This innovative iterative refinement framework enhances the output quality of large language models (LLMs) by enabling them to generate self-feedback and subsequently refine their output. This survey holds significant importance for generative AI, particularly in applications such as email and writing assistants, as it demonstrates a supervision-free method for improving the sophistication and appropriateness of LLM-generated text. Other approaches, such as REALIGN [20], also recognize the specific demands of such applications, explicitly including email generation among the 46 tasks with tailored criteria and formatting. The technical approach follows a three-step iterative process: initial generation, self-generated feedback, and refinement based on that feedback, with each stage performed by the same underlying large language model (LLM). The approach utilizes a few-shot prompting to guide the LLM in generating initial drafts, offering constructive feedback, and creating improved revisions. Notably, the process eliminates the need for additional training data, model fine-tuning, or reinforcement learning, thereby alleviating one key drawback of current refinement methods, which tend to be dependent on domain-specific data, external supervision, or reward models.

The results enhance prior knowledge by demonstrating that this straightforward, standalone method can improve even cutting-edge LLMs like GPT-4 at test time.

The study evaluates the effectiveness of the method proposed by Madaan et al. [23] across seven diverse tasks involving natural language and code generation. Using automatic metrics and human evaluations, it demonstrates consistent and substantial improvements in task performance, with an average absolute improvement of approximately 20%, and a clear preference over baseline LLM outputs.

Further advancing the concept of self-feedback and refinement, more specialized models have been developed to enhance output quality through iterative critique and refinement. As a follow-up to this critical evaluation of training processes, Wang T et al. [24]introduced Shepherd, a task-conditioned 7B-parameter language model, fine-tuned to critique and refine the output of other language models. This innovation addressed one of the key challenges of personalized email systems: how to produce higher-quality output through iterative feedback. Building upon LLaMA-7B, Shepherd significantly improved processes in LLM self-improvement, focusing on identifying diverse errors and providing helpful feedback. The model was trained on a high-quality community feedback dataset (Stack Exchange and Reddit) and human annotations. Shepherd's performance was rigorously tested against rival baselines, including Alpaca-7B, SelFee-7B, and ChatGPT, using both automated (GPT-4) and human testing. Results indicated that Shepherd performed better, with win rates ranging from 53% to 87% against alternatives in the GPT-4 evaluation, and human evaluators also found it to be comparable to or better than ChatGPT. Unlike untuned models that generate output passively, Shepherd's [24] model actively identifies mistakes, suggests corrections, and enhances multiple key determinants of quality, such as coherence, factuality, and fluency. This was especially relevant in highlighting the effectiveness of iteratively feedback-driven fine-tuning to significantly enhance AI-response quality and usability, with clear ramifications for improving personal email generation systems through continuous refinement with user feedback.

To address the issue of manually creating improvement goals and detailed rules (known as "rubrics") in approaches such as Self-Refine [23], the authors propose a new framework called ImPlicit Self-Improvement (PIT) [25]. PIT’s main innovation is that it learns how to improve responses independently without needing explicit instructions. Instead of providing detailed guidelines, it utilizes existing human preference data (used to train reward models) to understand what makes a response more effective.

This is achieved by reformulating the RLHF training objective to maximize the quality gap between a model’s response and a reference response, rather than simply optimizing for response quality given an input. The PIT framework uses a three-stage training pipeline. First, supervised fine-tuning (SFT) is applied to both satisfactory and unsatisfactory responses. Then, a reward model is trained based on the relative quality gap between reactions. Finally, a curriculum-based reinforcement learning strategy progressively refines the model's outputs.

Notably, PIT [25] can be integrated into the inference process to refine LLM outputs iteratively. Its effectiveness is demonstrated through extensive evaluations of three diverse datasets, including Anthropic/HH-RLHF and OpenAI/Summary. Regarding response quality, it outperforms prompting-based self-improvement methods, such as Self-Refine, as validated by automatic metrics (GPT-4 and DeBERTa) and human evaluations.

Building on the development of advanced interfaces and large language model (LLM) capabilities, Script&Shift [26] offers a layered interface paradigm designed to better align with natural writing processes, particularly for complex tasks. Although not explicitly designed for email, its architecture provides valuable insights into generative AI writing tools in diverse communication settings. This approach uniquely integrates content development ("scripting") with rhetorical strategy ("shifting") in a zoomable, non-linear workspace, facilitating fluid movement between drafting and organization.

The system's technical foundation rests on "layer primitives": distinct modules, including the Writing Layer for content creation, the Meta Layer for global context (audience, tone, and goals), and the Document Layer for compilation. These layers enable dynamic interaction with content generated by an LLM (Claude 3.5 Sonnet, in this case) through features such as embedded placeholders, context-sensitive prompts, and output rendering that respects document structure. Coordination is handled by a Prompt Composer, which formulates system instructions based on task knowledge, and a Workspace Manager, which orchestrates component interactions, ensures structural consistency, and manages content distribution across layers.

In summary, the reviewed tools and frameworks demonstrate a clear evolution from prompt-based generation to more interactive and structured co-writing systems. Innovations such as ABScribe and Script&Shift illustrate a growing emphasis on user control and interface design. At the same time, approaches like SELF-REFINE and Shepherd suggest promising avenues for improving generation quality through feedback and self-assessment. However, most tools still lack deep integration with user-specific context or real-time personalization mechanisms, signaling a gap between interface design and underlying model adaptation.

## 3.2 Personalization Techniques Found on LLM Writing Assistants

While generic LLMs demonstrate remarkable performance in text generation, the true promise of artificial intelligence in email communication hinges on its capacity to provide highly personalized responses. In this subsection, we discuss advanced methodologies specifically designed to personalize LLMs to the unique styles, preferences, and subtle contextual requirements inherent in email communication. We consider a variety of methods aimed at achieving significant and effective personalization, analyzing their strategies critically in terms of data efficiency, privacy, and adaptive responsiveness to users in email contexts.

Recent advances in personalized generative AI, such as PersonaAI [27], have overcome the limitations of traditional general-purpose Large Language Models (LLMs), including ChatGPT, in delivering profoundly personalized experiences, particularly in user-facing applications and sophisticated use cases, like email writing assistants. By combining Retrieval-Augmented Generation (RAG) with the LLAMA model, this layered interface significantly enhances personalization, allowing it to respond dynamically to each user's unique preferences and nuances. PersonaAI's solution uses a mobile app to capture real-time user data via voice-to-text transcription. This data is saved to a cloud database for processing, where it's formatted and converted into 384-dimensional vectors. To perform this conversion, the system utilizes an embedding model, specifically the BAAI/bge-small-en model [28] from Hugging Face, which transforms the text into numerical representations that capture its semantic meaning. This process is crucial because it enables the fast semantic retrieval of data. The system then dynamically retrieves the top-k most contextually similar contexts using a cosine similarity function, returning responses tailored to the user's specific needs.

Additionally, advanced prompt engineering fine-tunes the model to produce contextually fit content, and built-in error handling enhances the user's confidence. In addition, a lightweight and scalable architecture makes it a viable option for mass-scale personalized AI deployment, and its ethical design focus guarantees user trust and privacy. High contextual retrieval precision (91%) and low query response time (<1 second) were achieved through performance testing using datasets such as simulated university journals, demonstrating the system's feasibility in real-world applications. Such technological advances underscore the potential of PersonaAI to drive personalized AI, particularly in tasks that require fine-grained communication, such as advanced email writing assistants.

While the work presented in [27] focuses on retrieval techniques for personalization, another significant advancement in this domain is the preference agent approach [29]. This method introduces a way to personalize content generated by large language models for tasks like email writing, explicitly addressing the challenge of adapting their broad capabilities to meet individual user preferences. Traditional methods, such as in-context learning and parameter-efficient fine-tuning, often struggle to capture the nuanced complexity of human preferences, particularly with smaller, personalized datasets. To overcome this limitation, the authors of [29] introduced these agents as small, locally trainable models that encode user preferences into concise natural language rules. These agents act as a "steering wheel," guiding the output of a larger, more generic large language model (LLM) to align with a personalized style and content, all without requiring fine-tuning of the larger model. This modular method separates the preference learning process from the generic LLM, offering a more scalable and flexible solution for personalization compared to the Retrieval-Augmented Generation (RAG) approach used in PersonaAI [27].

The authors Shashidhar S., Chinta A., et al. [29] build upon a systematic process for capturing user preferences. The proposed method utilizes a large LLM to generate zero-shot responses, which are then compared with the ground truth outputs to identify differences. These differences derive preference rules, which a smaller model then learns. This model becomes the personalized preference agent, capable of generating rules that guide the large LLM at inference time. During inference, the trained preference agent provides context in the form of natural language rules to the large model, allowing it to generate outputs aligned with the user's preferences. Evaluations across three diverse datasets — Enron emails, New Yorker articles, and Amazon product reviews—demonstrate that preference-guided LLMs significantly outperform both fine-tuning baselines and standard prompting methods in terms of automatic metrics (such as GPT-4o evaluation) and human judgments.

Following the research on personalization techniques, Panza [30]emerges as a novel solution focused on personalized text generation, particularly for emails, while prioritizing user privacy through local execution. It tackles the efficient challenges of fine-tuning and Retrieval-Augmented Generation (RAG) faced by predecessors. Panza uniquely combines a variant of Reverse Instructions with RAG and parameter-efficient fine-tuning (PEFT) methods, such as ROSA, enabling personalization using tiny datasets (under 100 emails) on commodity hardware. This makes it highly viable for users with limited personal data.

A key methodological contribution is its evaluation approach, demonstrating that a combination of BLEU and MAUVE scores strongly correlates with human preferences for personalized text quality. This combined metric helps validate Panza's ability to effectively replicate writing styles. Unlike purely RAG-based or preference-agent approaches, Panza provides a scalable and flexible solution that enables local execution, low-cost fine-tuning, and inference on standard hardware. Its practical utility is showcased through a Google Chrome plugin designed for Gmail integration.

The authors Shaikh O., Lam M., et al. [31] introduce a distinct method for aligning large language models (LLMs) with individual user preferences, complementing other personalization frameworks. Their "Demonstration Iterated Task Optimization" (DITTO) approach takes a more direct approach by leveraging user-provided demonstrations. This iterative process only requires a small number of prototypes (fewer than 10) to guide the model's output, making it a data-efficient solution that contrasts with the more resource-intensive methods of fine-tuning and reinforcement learning from human feedback (RLHF) often used in personalization, as already stated.

The online imitation learning method, introduced in [31], stands out by treating user demonstrations as preferred outputs and utilizing this feedback to update the model via algorithms such as Direct Preference Optimization (DPO). Compared to Panza, which focuses on fine-tuning foundation models to mimic writing styles with minimal data while prioritizing privacy, this approach offers a simpler alternative by directly optimizing model outputs based on few-shot user demonstrations. This method allows it to bypass the need to fine-tune the entire model or implement complex RAG-based retrieval systems.

Furthermore, this technique addresses some challenges inherent in the preference agents' approach, where user preferences are encoded into specific rules. While preference agents can effectively guide output, the online imitation learning framework offers a more scalable and flexible approach to leveraging direct user feedback for preference alignment, thereby reducing the need for complex rule creation and extensive preference elicitation. Its methodology also contrasts with Panza's emphasis on local execution and privacy, as it focuses on the iterative refinement of user preferences, making the system adaptable to diverse user needs without compromising performance. Evaluations using static benchmarks (such as CMCC and CCAT) and real-world user studies for email writing consistently show its effectiveness in personalizing LLMs with minimal data. By outperforming traditional methods, such as supervised fine-tuning and few-shot prompting (even with GPT-4), this system demonstrates its ability to capture fine-grained stylistic preferences with only a few demonstrations, positioning it as a powerful and efficient solution alongside the personalization techniques discussed previously.

The evolution of personalization techniques reflects a broader shift in how large language models (LLMs) are transforming individualized AI systems. Zhang et al. [7] explored the disruptive impact of large language models, such as GPT-3.5, GPT-4, and LLaMA-7B, highlighting how LLMs transform personalization from filtering to dynamic, real-time user engagement. Unlike static embedding-based models, LLMs utilize few-shot prompting and in-context learning to adapt recommendations dynamically. Reinforcement Learning from Human Feedback (RLHF) further refines personalization by aligning responses with user intent. The study highlighted the ability of LLMs to integrate external tools, including retrieval-based recommendation engines (e.g., the Generalized Dual Encoder Model), search APIs, and vector databases, thereby enhancing context-aware personalization.

These observations provide valuable context for understanding the technological foundations that underpin the personalization approaches of PersonaAI, preference agents, Panza, and DITTO, all of which leverage these capabilities in different ways.

The reviewed personalization techniques highlight a transition from model-centric fine-tuning to modular, data-efficient personalization strategies. Approaches such as PersonaAI, Panza, and DITTO show that personalized outputs can be achieved even with limited user data, particularly through RAG and preference agents. Yet, a trade-off remains between scalability, privacy, and stylistic fidelity. While some models excel in fast deployment or local inference, others prioritize accuracy in capturing writing nuances, suggesting that no single method currently balances all personalization goals effectively.

## 3.3 Security Vulnerabilities in Generative AI Email Systems

This section highlights key threats in AI-driven email environments, ranging from ecosystem-level attacks, such as the Morris-II worm, to advanced exploits, including LLM-enabled spear phishing and Trojan attacks on fine-tuning processes. Understanding these risks is crucial for building secure and reliable AI email assistants that protect user data and maintain the integrity of communication.

A recent study has discovered a significant security weakness in Generative AI (GenAI) environments, particularly those powered by RAG, such as email assistants. The authors Cohen S., Bitton R., et al. [32] present Morris-II, a new computer worm that propagates across GenAI platforms via embedded adversarial prompts in emails. The result is indirect prompt injections, which can activate malicious behaviors, such as data exfiltration. The authors performed empirical experiments with mock GenAI email assistants developed using the LangChain library and RAG to examine this threat. The assistants were exposed to tailored databases extracted from real-world email datasets, such as the Enron and Hillary Clinton email datasets, to replicate the behavior of real-world GenAI systems. For instance, the Enron dataset simulated 20 workers, each with a personal database of 100 emails, whereas the Hillary Clinton dataset consisted of 1,500 emails. These datasets constituted the "external knowledge sources" of the RAG components of the assistants.

The simulations were utilized to demonstrate the worm's potential to infect the systems under various scenarios. The paper proposes Virtual Donkey, an effective defense solution that detects worm propagation according to input-output similarity in the GenAI model, as a countermeasure. The solution has high accuracy with minimal false positives. While the research acknowledges the potential for adaptive attacks, it makes a valuable contribution by revealing this ecosystem-level threat and proposing an effective solution to secure GenAI-based email assistants.

Building upon these ecosystem-level concerns, the author Hazell [33] examines the capacity of LLMs, such as OpenAI's GPT-3.5 and GPT-4, to facilitate spear phishing attacks, a sophisticated form of social engineering that leverages personalized information to manipulate targets.

The research highlights LLMs' ability to assist with the reconnaissance phase by processing unstructured data to gather target information and the message generation phase by producing realistic and contextually relevant spear phishing emails at a fraction of a cent per message. Notably, the study demonstrates how basic prompt engineering can bypass safety measures in these models, enabling the generation of malicious content and advice on conducting attacks, including crafting persuasive emails and basic malware. By generating unique spear phishing messages for a large group of UK Members of Parliament, the study provides evidence that LLMs can produce realistic and cost-effective phishing attempts, potentially scaling such campaigns significantly and lowering the barrier for less skilled cyber criminals. The findings also compare the sophistication of different LLMs, including open-source models, in this context. While the study focuses on the malicious application of LLMs for email-based attacks, it underscores the dual-use nature of this technology and the governance challenges associated with preventing its misuse. The paper further discusses potential solutions, including structured access schemes and the development of LLM-based defensive systems for email security.

Researchers subsequently began identifying additional security vulnerabilities that could arise with the deployment of LLMs. Building on the phishing concerns identified by Hazell [33], a critical security evaluation was proposed, demonstrating how instruction-following large language models can be exploited for malicious purposes through methods borrowed from classical computer security. The authors Kang D., Li X., et al. [34] developed three primary attack vectors—obfuscation (inserting typos or replacing synonyms to escape detection), code injection/payload splitting (indirect programming of instructions), and virtualization (programming attacks in virtual scenarios)—that could evade OpenAI's content filtering mechanisms 100% of the time in situations like hate speech, conspiracy theories, and phishing schemes.

The economic implications of these vulnerabilities are significant. Using human evaluators and GPT-4 to assess generation quality, Kang et al. [34] found that instruction-tuned models of larger capacities produced much more realistic malicious content than earlier models. Their economic research revealed that tailored malicious content could be generated at a cost of between $0.0064 and $0.016 per instance, significantly less than human-generated content, which is estimated to cost $0.10 per instance. This cost-effectiveness created strong economic incentives for adversaries to deploy these systems. This identified the dual-use possibility of AI-powered email composition, where features such as personalization and productivity could be leveraged for malicious purposes, including sophisticated phishing attacks.

Beyond prompt-based vulnerabilities and economic incentives, adaptation-based security concerns have also emerged. Because of these broader security concerns, the authors Dong T., Xue M., et al. [35] analyzed another potential vulnerability in their paper, "The Philosopher's Stone: Trojaning Plugins of Large Language Models." Their research on low-rank adaptations for LLMs identified critical security concerns relevant to using AI-mediated communication systems. The researchers have identified weaknesses in LoRA adapters to demonstrate how attackers could design "Trojan plugins" that make the LLMs generate toxic text when triggered by specific phrases.

These adaptation-based attacks represent a sophisticated evolution of security threats in generative AI systems. The authors Dong T., Xue M., et al. [35] also introduced two new attack methods: POLISHED, which uses LLM-based paraphrasing to produce naturalistic poisoned datasets, and FUSION, which transforms benign adapters through an over-poisoning process. Testing on actual-world LLMs, such as Llama (7B, 13B, 33B) and ChatGLM2 (6B), confirmed the approaches, with FUSION achieving a nearly 100% success rate in producing target keywords based on just 5% poisoned data. These findings highlighted key security issues for AI-powered communication systems that utilize adapter-based fine-tuning techniques to adjust models to specific domains or users. Since adapter-based techniques are commonly used in email personalization systems, these vulnerabilities raised questions about the secure deployment of generative AI in email ecosystems, connecting back to the ecosystem-level threats identified in [32]

The security analysis reveals that personalized large language models (LLMs) for email systems are vulnerable to a wide range of attacks, including prompt injections and Trojan plugins. While solutions such as Virtual Donkey and structured access models offer partial mitigation, the literature shows that many personalization methods, particularly those relying on adapters or external retrieval, introduce new vectors for exploitation. This underscores the urgent need for security-aware personalization frameworks and defensive evaluation protocols to accompany the design of LLM-based email assistants.

## 3.4 User Perceptions of AI Communication and Writing Assistants

Beyond AI's technical capabilities, the success of email writing assistants depends heavily on user perception and interaction. This section reviews key studies highlighting the balance between AI-driven productivity and concerns about authenticity, emotional tone, and trust. Examining user experiences across contexts—from accessibility to professional use—emphasizes the need for AI that is both efficient and aligned with human communication nuances.

In 2022, Goodman et al. introduced LaMPost [36], a language model-powered email composition tool designed for dyslexic adults. The research highlighted the potential for AI systems to be re-tuned to address specific user needs rather than overall productivity enhancement. Unlike traditional spell-check and grammar-check software, LaMPost provided high-level composition support, including content structuring, subject line generation, and stylistic rewriting. The system was built with LaMDA, a conversation-specific LLM, in a browser-based email editor. The test with 19 dyslexic adults showed that the "rewrite" and "subject line" tools significantly enhanced writing productivity, but participants occasionally experienced issues with coherence and tone deviation. Perceived self-efficacy was not influenced by awareness of AI assistance, indicating that the system was employed as an empowering tool, not an intrusive helper. The study emphasized the need for adaptive AI-provided feedback to better handle cognitive diversity within writing support tools, demonstrating how personalization can be augmented to incorporate mental and accessibility dimensions beyond style considerations.

Following these findings on personalized writing support, later research investigated user experience and usage of AI-facilitated communication tools in daily life. The authors Fu Y., Foell S., et al. [37] presented an in-depth diary and interview study of user experience with tools that mediate communication through artificial intelligence in daily interpersonal interactions. They conducted the study with 15 users who used tools of their choice for one week, resulting in 227 diary entries that captured their experiences and attitudes. The study found general positive acceptability with an average satisfaction score of 7.1 on a scale of 1-10, with satisfaction rising following participants' initial learning curve. One of the key contributions was the definition of four communication spaces, distinguished by stakes (high/low) and relationship dynamics (formal/informal); these AI tools were perceived as considerably more appropriate in formal relationships than in informal ones. The participants noted that these systems were beneficial by increasing communication confidence, helping to find precise words to express ideas, overcoming cross-cultural communication challenges, and expanding vocabulary. However, the study also discovered ongoing flaws in current AI communication systems, including excessive verbosity, unnatural phrasing, exaggerated emotional intensity, and the difficulty of iterative revision required to achieve satisfactory outputs. These findings suggest that the tools must be tuned differently for specific communication contexts, with features matched to their functions, and conclusions directly applicable to crafting customized email systems that are sensitive to differing communication contexts.

Researchers shifted from educational applications to professional settings and began examining AI writing tools in business environments. In a corporate environment, the authors Jovic M. and Mnasri S. [38] compared four well-known large language models (LLMs)—ChatGPT 3.5, Llama 2, Bing Chat, and Bard—in terms of their ability to generate business emails, examining the implications for AI implementation in business communication. Using a detailed framework, the study assessed performance across three types of emails: routine, negative, and persuasive. Each LLM was subjected to identical email scenarios, with outputs scored on content, format, and tone. Despite the formulaic nature of business emails, the researchers found significant variations in quality between models. Llama 2.0 achieved the highest overall score (48.9/60), followed closely by Bing (47.8), ChatGPT 3.5 (46.7), and Bard (45.2). Common weaknesses across all models included difficulties in following the requested structure, maintaining tone consistency, and responding to emotional cues, which highlighted the need for further development of LLMs in business email generation, particularly in terms of emotional aspects. However, the study employed only these tests and did not explore or analyze prior personalization methods, indicating that further development was necessary to address these issues in tailored business communication.

Beyond performance comparisons, understanding user trust in AI-generated content emerged as a critical area of research. In a study, user perceptions of AI-generated content were explored, revealing that trust in AI-generated emails decreased as the perceived AI involvement increased [39]. Using a "Wizard-of-Oz" approach, where participants believed they were interacting with an AI system but were interacting with a human, the study found that users were more willing to accept AI-generated content for factual emails but preferred human authorship for emotionally charged content, such as condolences.

To enhance the comprehension of stylistic variation in artificial intelligence versus human writing, the authors Li W., Saha K, et al. [40] compared AI and human-generated emails and investigated the primary distinction that may influence user acceptance and perception. Based on the W3C email corpus, the research compared the syntactic, semantic, and psycholinguistic attributes of emails created by GPT-3.5, GPT-4, Llama-2, and Mistral-7B. Although the findings replicated anxieties regarding AI-generated emails being formal and emotionally monotonous, Li et al. also highlighted the difference in style: AI-generated emails were verbose and lexically redundant, whereas human-generated emails were succinct, personalized, and linguistically varied. Although polite and positive, LLM-generated emails often lack contextual specificity, underscoring the need for personalization, as demonstrated by several studies in this literature review. These findings are corroborated by a small-scale user study involving 41 participants, in which, although they praised the efficiency of AI-generated writing, they criticized its limited diversity and personalization. This highlights the reality that although LLM-based email automation is efficient, it can be complemented by hybrid approaches that balance AI-driven efficiency with user customization, thereby adding authenticity and adaptability. The study highlighted a significant trade-off of email generation systems: reconciling the grammatical accuracy and speed of AI-produced messages with the conversational, personalized tone of human-composed emails.

Studies on user perception consistently point to a tension between productivity and authenticity. While users appreciate speed and ease of use, concerns persist regarding emotional tone, verbosity, and lack of contextual precision in AI-generated emails. Importantly, acceptance varies across domains — business, accessibility, and personal contexts — highlighting that personalization must extend beyond style to include purpose, audience, and emotional resonance. Future systems will need to adapt not just to how users write, but why they write.

## 3.5 Benchmarking and Evaluation

Robust evaluation methods are crucial for accurately measuring the performance, personalization, fidelity, and impact of LLM-based email systems. This section reviews emerging benchmarks focused on user feedback responsiveness, long-form coherence, and on-device efficiency, highlighting a shift toward hybrid evaluation frameworks that combine automated metrics with human judgment.

To more effectively quantify large language models (LLMs) in interactive settings, the authors Yan J., Luo Y., et al. [41] introduced RefuteBench, a novel benchmark designed to assess an LLM's ability to systematically incorporate refuting user feedback. This approach addresses a vital challenge in generative AI: LLMs often struggle to incorporate user corrections, which limits their effectiveness in applications like email writing assistants where user-specific adjustments are essential. RefuteBench takes a dynamic approach by generating counter-instructions to test models across various tasks, including question answering, machine translation, and email writing. This contribution addresses a key gap in current instruction-following evaluations by targeting scenarios where users actively refine or correct model-generated responses. The benchmark rigorously evaluates LLMs' compliance with updated instructions through both single- and multi-feedback scenarios and introduces two novel evaluation metrics: Feedback Acceptance (FA), which assesses positive acceptance of feedback, and Response Rate (RR), which evaluates the correct incorporation of feedback into future interactions. To overcome these recognized drawbacks, the authors present a novel and pragmatic "recall-and-repeat" prompting strategy that leverages past feedback to increase model responsiveness. Experimental results showed significant gains in Response Rate across a range of LLMs, testifying to the effectiveness of this strategy. Furthermore, the study's examination of the relationship between FA and RR provides valuable insight into the importance of accepting initial feedback for ongoing compliance. Data gathering involved existing datasets, such as RIPPLEEDITS and WMT2023, along with custom-created email writing instructions to provide a comprehensive evaluation. Essentially, this study contributes to a deeper understanding of LLMs' interactive capabilities by identifying their limitations in responding to refuted instructions and proposing a viable solution to enhance their responsiveness, with direct implications for the field of generative AI assistants.

While the work by Yan et al. [41]prioritizes the use of feedback and responsiveness, a different study by the authors Xu et al. [42] tackles another significant challenge for generative AI email assistants: high inference latency in on-device Large Language Models (LLMs), which is a critical issue for systems processing long contextual prompts. Their paper introduces llm.npu, the first LLM inference system to successfully utilize on-device Neural Processing Unit (NPU) offloading to solve this problem, particularly in the dominant prefill stage. The system integrates three new techniques: chunk-sharing graphs, which improve efficiency and lower memory overhead by allowing for variable-length prompt handling through division into fixed-size chunks; shadow outlier execution, which preserves accuracy by offloading the processing of significant activation outliers to the CPU/GPU in parallel; and out-of-order subgraph execution, a scheduling framework that intelligently improves the utilization of heterogeneous mobile processors (CPU/GPU and NPU) by allocating and executing Transformer blocks based on their hardware affinity and precision sensitivity.

Rigorous testing performed by the authors of [42] on standard mobile hardware demonstrated that the performance improvements of their system are significant, with prefill speedup rates of up to 43.6 times that of GPU benchmarks and substantial power savings of up to 59.5 times, all while preserving inference accuracy. In extensive real-world usage, particularly in scenarios involving intelligent email assistants with lengthy prompts, this approach achieves latency improvements of 1.4 and 32.8 times compared to competing benchmarks. Notably, the system achieves prefill speeds of over 1,000 tokens per second for billion-parameter models on mobile devices, marking a significant advancement in enabling rapid and efficient on-device generative AI capabilities for apps that require the accelerated processing of large amounts of contextual information.

Shifting from system-level advances to personalization techniques, a key contribution to the literature is provided by work on retrieval optimization in LLM personalization. The LaMP benchmark [43], which evaluates performance on seven personalized NLP tasks, reports that retrieval optimization yields a 5.5% average improvement in LLM personalization, with a notable 33.8% improvement in cold-start settings. Pre- and post-generation retrieval selection mechanisms were one of the novel features, which picked alternate retrieval methods per query to trade off recency, keyword relevance, and user writing style. These findings underscore the importance of adaptive retrieval selection in producing highly personalized LLM responses, with direct applications to email response generation systems.

To further enhance the standards of personalization, the authors Kumar I., Viswanathan S., et al. [44] addressed an essential requirement within evaluation tools by introducing LongLaMP, a specialized benchmark designed to personalize the generation of long textual content across various tasks, including email writing. This innovation remains highly relevant in email because previous personalization datasets concentrated primarily on short-term textual outputs without regard for coherence and consistency issues within lengthy communications. Contrary to the state of the art at the time, LongLaMP focused more on long-term coherence, stylistic coherence, and topic consistency within lengthy passages.

The authors used a RAG model where documents and features specific to users were leveraged to constrain the generation step to avoid common problems like topic drift. The LongLaMP [44] dataset was large, consisting of multi-paragraph custom text samples from email conversations, scientific abstracts, and web reviews. Quantitative metrics presented in this article indicate that RAG-improved models outperform conventional fine-tuning approaches by a range of 5.7% to 128%, as observed in various evaluation metrics, including BLEU, ROUGE-L, and METEOR. The suggested evaluation framework categorizes models along two critical dimensions: user-based (cold-start) personalization, which assesses the capacity of models to generalize to new users with minimal or no prior history, and temporal personalization, which examines how models evolve as they adjust to changing user tastes over time.

Among the most noteworthy findings were that dense retrieval-based Contriever methods, which use neural networks to understand the semantic meaning of text, vastly outperformed traditional keyword-based methods, such as BM25, which rely on word frequency and occurrence, in determining beneficial personalization signals by an enormous margin, with significant implications for retrieval-based personalization methods in email systems.

Evaluation practices for personalized large language models (LLMs) are evolving, with new benchmarks addressing the incorporation of context, responsiveness to feedback, and stylistic coherence. However, inconsistencies in datasets, a lack of long-form personalization metrics, and overreliance on automatic scoring still hinder comprehensive assessment. The emergence of benchmarks like RefuteBench and LongLaMP shows progress but also signals the need for hybrid evaluations that combine automated measures with human judgment — particularly for nuanced tasks like email response generation Summary of Comparative Analysis.

To synthesize the findings from the Related Work section, Table 3.2 provides a comparative classification of the systems and frameworks discussed. The table maps each study's focus across core personalization techniques (RAG, PEFT, iterative refinement), user experience design, and critical research aspects like security and evaluation, serving as a visual guide to the current research landscape.

Table 3.2 - Feature Comparison of Surveyed Systems for Personalized Email Generation

| **Study Reference** | **Publication year** | **RAG Focus** | **PEFT/Fine-Tuning Focus** | **Iterative-Refinement** | **User Experience/Interaction Design** | **Security/ Privacy Aspects** | **Evaluation/User study** |
|---|---|---|---|---|---|---|---|
| Luminate [19] | 2024 | | | | ✓ | | ✓ |
| REALIGN [20] | 2024 | ✓ | | | | | ✓ |
| ResQ [21] | 2025 | ✓ | | | ✓ | | ✓ |
| SELF-REFINE [22] | 2023 | | | ✓ | | | ✓ |
| Shepherd [23] | 2023 | | ✓ | ✓ | | | ✓ |
| PIT [24] | 2023 | | ✓ | ✓ | | | ✓ |
| Script&Shift [25] | 2025 | | | | ✓ | | |
| LaMPost [36] | 2022 | | | | ✓ | | ✓ |
| PersonaAI [27] | 2025 | ✓ | | | ✓ | ✓ | ✓ |
| Preference Agents [29] | 2023 | | ✓ | ✓ | | | ✓ |
| Panza [30] | 2025 | ✓ | ✓ | | ✓ | ✓ | ✓ |
| DITTO [31] | 2023 | | ✓ | ✓ | | | ✓ |
| Morris-II [32] | 2024 | | | | | ✓ | |
| llm.npu [42] | 2025 | | | | | | ✓ |

This review of the state-of-the-art reveals that while significant progress has been made, the field of personalized email generation is fragmented. Research has yielded powerful, specialized techniques, including Retrieval-Augmented Generation (RAG) for enhancing contextual awareness, Parameter-Efficient Fine-Tuning (PEFT) for stylistic adaptation, and iterative refinement models for improving output quality. However, these techniques are often explored in isolation. There is a clear gap in the literature regarding systems that unify these approaches under a single, coherent architectural paradigm.

Furthermore, while user perception studies highlight a strong demand for authenticity and contextual appropriateness, few systems provide a transparent and user-accessible way to define and manage the complex set of rules, facts, and stylistic nuances that constitute a digital persona. The concept of a centralized, user-editable ontology that governs the entire generation process remains largely unexplored.

This analysis concludes that the next logical step in advancing AI-powered writing assistants is the integration of these disparate techniques into a holistic system, governed by a structured, explicit knowledge framework. It is this identified gap that our project directly addresses. The following sections will detail the architecture and implementation of our ontology-driven system, which was designed to synthesize these state-of-the-art methods to deliver a more robust and genuinely personalized email automation experience. Only when strictly necessary, maintaining a well-written report

# 4. System Architecture and Design Rationale

Having established the theoretical underpinnings, this chapter moves on to tangible applications by outlining the technical framework of the proposed solution. Tackling the issue of generic AI communication requires more than just an advanced language model; it demands a carefully structured architecture designed to direct, limit, and enhance the model's functionalities.

In today's modern world, the typical methodology for using artificial intelligence to communicate via email is typified by a tedious, multi-step process. A user receives an email, copies its content, and pastes it into a separate, generic chatbot, such as ChatGPT. Next comes a series of iterative, manual commands: "write a response to this," "now make it more formal," "abbreviate it," "edit this particular phrase." While such a process works, its workflow is considerably inefficient and imposes significant cognitive overhead. It distracts from the user's focus, requires constant context switching between different apps, and fails to build a lasting understanding of a user's unique voice or contextual history. The result is, therefore, a generic and often impersonal response that still requires considerable manual effort to produce real personalization.

The design of this system was conceived as a direct response to limitations in conventional workflows. Rather than forcing the user to engage in a fragmented, chat-based interaction with a traditional tool, it offers a consolidated and unified app. It creates a direct link to the user's inbox, thereby eliminating the need for copy-pasting. It then guides the user through a systematic process for creating a response, offering specific assistance when needed. The entire process is managed by a backend system based on a Personalization Ontology—defined as the system's "cognitive core"—which has already been tailored to match the user's identity. The user can ultimately send off the generated mail directly from inside the app's user interface, completing the workflow efficiently in one streamlined process.

Furthermore, the system's architecture addresses another critical challenge: the burden of managing a constant inbox. The automated workflow acts as a proactive agent. By running in the background, it can autonomously generate a draft response the moment an email arrives and send a notification to the user's phone. This transforms email management from a reactive chore into a simple, on-the-go "approve or reject" decision, ensuring timely and intelligent communication without being tied to a desk.

## 4.1 A High-Level System View

To translate the project's principles into a functional application, the system is organized into several distinct but interconnected components. This modular design, illustrated in Figure 4.1, separates the user-facing interface from the core business logic, the data persistence strategies, and the asynchronous processing required for automation.

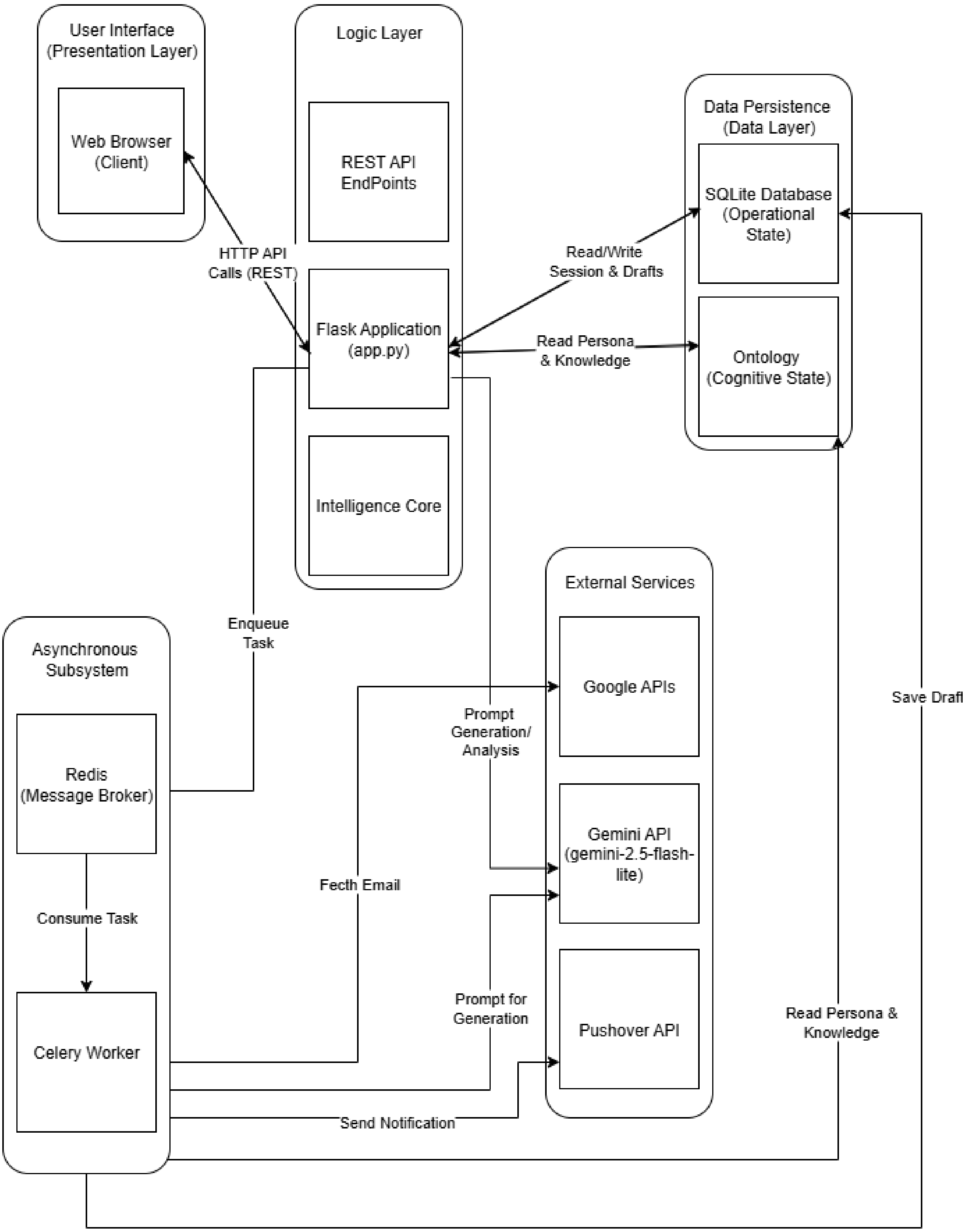


Figure 4.1 – High-Level System Architecture

The architecture is centered around a Python-based backend that serves as the central hub for orchestration. It communicates with a client-side web application, manages a dual-state data layer for operational and cognitive knowledge, and offloads long-running tasks to an independent background worker. This entire ecosystem is designed to interact securely with a suite of external services, including Google's Gmail and Gemini APIs, to perform its intelligent tasks. The following diagram provides a schematic of these components and their primary interactions, which will be detailed in the subsequent sections.

## 4.2 The Core Layers

At its heart, the assistant is an information processing engine. Its architecture is therefore designed as an integrated system with three core layers—Presentation, Logic, and Data—that work in concert to manage the flow of information from perception to action. This section details the responsibilities and design rationale of each layer, demonstrating how they combine to form a robust and intelligent application.

## 4.3 The Presentation Layer

All human interaction with the system originates in the Presentation Layer, a dynamic single-page web application that functions as the user's command center. The single most important architectural decision for this layer was to ensure it is completely decoupled from the backend. It operates as a pure client-side application, communicating with the core system exclusively through a stateless, RESTful API.

This separation provides a crucial advantage: modularity. The entire user interface could be completely rewritten in a different framework—for example, migrating from vanilla JavaScript to React or Vue.js—without necessitating a single change to the backend services. This future proves the application and allows for independent development and maintenance of the system's front-facing and internal components

## 4.4 The Logic Layer

Signals from the command center are sent as API requests to the system's central nervous system: the Logic Layer. This layer, implemented as a Flask application (app.py), acts as the orchestration hub for all business logic, securely managing user authentication and routing incoming requests to the appropriate functions.

Its most vital component is the Intelligence Core, which executes all AI-driven tasks. The innovation here is not simply in calling a large language model, but in performing sophisticated prompt orchestration. The Intelligence Core acts as a director, synthesizing multiple streams of information—the user's explicit guidance from the UI, the persona's defined rules from the ontology, and relevant facts retrieved from the knowledge base—into a single, comprehensive set of instructions. This detailed prompt is then sent to the external Gemini API, effectively commanding the powerful but generic model to generate a response that is contextually aware, factually grounded, and stylistically aligned. This is a direct and practical implementation of the Retrieval-Augmented Generation (RAG) paradigm.

## 4.5 The Data Layer

For the orchestration performed by the Logic Layer to be truly intelligent, it requires access to memory. However, a generic LLM's memory is abstract and stateless, making it prone to the "confabulations" and factual inconsistencies discussed in the theoretical foundations. The most critical architectural innovation of this project is therefore the design of a persistent, external "mind" that serves as a cognitive anchor for the AI. To achieve this, the system implements a Dual-State Knowledge Architecture, a sophisticated strategy that resolves the tension between the need for a stable, long-term identity and the demands of a flexible, short-term operational workspace. This conceptual separation, illustrated in Figure 4.2, is the primary mechanism that grounds the assistant.

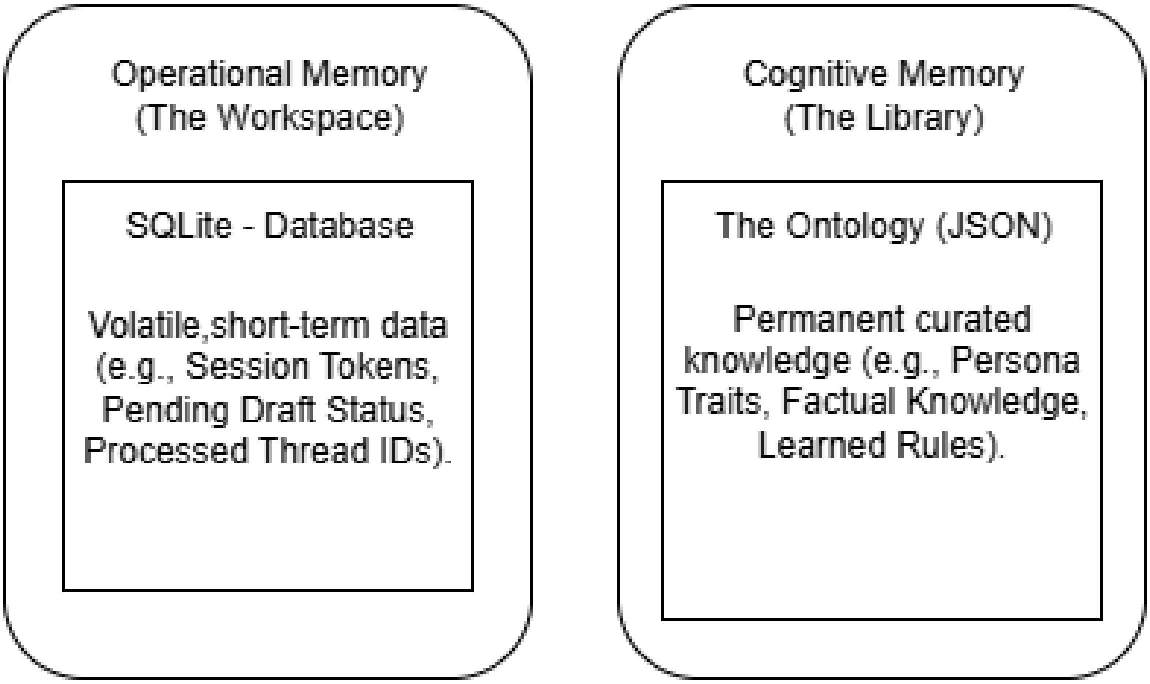


Figure 4.2 – Conceptual Diagram of the Data Layer

The first component is the Operational Memory (The Workspace). Managed by a pragmatic SQLite database, this is the system's ephemeral, high-speed workspace. It is exclusively concerned with the immediate state of operations: managing user session credentials, tracking the status of pending drafts, and logging processed email threads to ensure idempotency. It handles what the system is doing.

The second, and far more significant, component is the Cognitive Memory (The Library). This is the system's permanent, long-term memory and the definitive source of its identity, physically embodied in the Personalization Ontology. This curated library of knowledge acts as the primary guardrail against LLM confabulation, providing a verifiable source of truth that the Intelligence Core uses to augment and constrain every generated response. By externalizing the AI's core knowledge into a structured JSON file, we gain three profound advantages:

- Human Readability and Auditability: The AI's entire "brain" is stored in a simple, text-based format, allowing its knowledge and rules to be easily inspected and understood without the need for specialized tools.
- Decoupling: The system's identity is completely decoupled from the application logic. The "mind" can be evolved, backed up, or even transferred to a different system without altering the core codebase.
- Knowledge as Code: Because the ontology is a text file, it can be managed with version control systems like Git. This powerful paradigm allows the evolution of the AI's knowledge to be tracked, reviewed in pull requests, and rolled back if necessary, treating the system's intelligence as a rigorously managed software asset.

This dual-state design provides a robust and elegant solution, giving the assistant both a flexible workspace for its tasks and a stable, auditable library for its thoughts, ensuring that its communication is not only intelligent but also consistent and trustworthy.

## 4.6 The Asynchronous Backbone

A system that is expected to be both proactive and responsive encounters a practical challenge: it must cope with tasks whose duration is unpredictable and sometimes quite lengthy. Operations like AI-driven text generation or calls to external APIs often take several seconds. Running them synchronously inside a web application would lock the interface and, worse, cause incoming webhook notifications—many of which expire quickly—to fail. To avoid this bottleneck, the system is built around an asynchronous backbone that follows the well-known Producer–Consumer pattern.

Figure 4.3 shows how this design separates two concerns: giving an immediate "yes, I received your request" and doing the heavier work that follows. This separation keeps the system responsive while still making sure every task is processed. The architecture comes together through three moving parts.

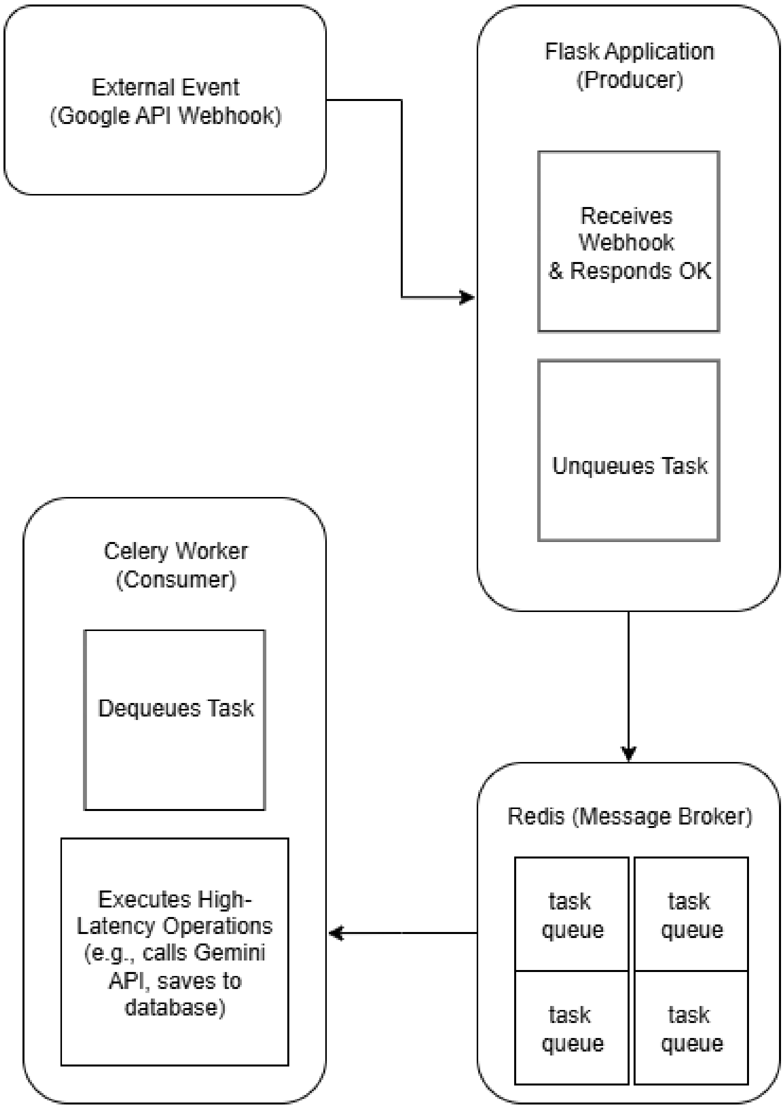


Figure 4.3 - Asynchronous Flow that represents the Consumer/Producer Pattern

- **The Producer (an agile receptor).** The Flask app plays this role. When an event arrives at the /gmail-webhook endpoint, the app responds instantly. It performs only a quick duplicate check, then places a message on the queue. By returning a success code in just a few milliseconds, it meets the strict timeouts of the webhook provider.
- **The Message Broker (a resilient buffer).** Redis acts as the middle layer, holding queued tasks. It ensures nothing is lost if requests arrive in bursts or if the worker needs a restart. This buffer is what gives the workflow durability.
- **The Consumer (an autonomous workhorse).** The Celery worker (celery_worker.py) runs separately from the web app. It pulls tasks from Redis and executes the heavier logic: fetching emails, building prompts, and calling

the Gemini API. Because the worker is decoupled, multiple workers can be added when load increases, without slowing down the main application.

Together, these three components achieve what the project requires most: a responsive interface, protection against task loss, and room to scale in the future. The asynchronous backbone enables the assistant to run smoothly and reliably in the background.

## 4.7 The Asynchronous Backbone

The architecture presented in this chapter is the result of a deliberate set of engineering decisions. Each pattern and technology were selected, not because it is inherently good or bad, but because of how it serves the project's core principles. This logic ensures that the end system is not only functional but also sustainable, scalable, and tailored to its specific function.

The principle of pragmatism guided the selection of core backend technologies. Flask was preferred over the larger, "batteries-included" frameworks due to its minimalist base, allowing the flexibility to implement the unconventional dual-state data layer without being constrained by a rigid Object-Relational Mapper (ORM), which is a programming layer that automatically maps database tables to objects in code, simplifying everyday data operations but often enforcing strict conventions and reducing low-level control. Similarly, SQLite was preferred for handling the running state. For the project's single-user context, its serverless, zero-configuration nature provided all the desired transactional guarantees without the runtime overhead of a full-scale database server, such as PostgreSQL.

The modularity principle is best illustrated by the clean separation of concerns within the system. The strict modularization of the presentation layer via a RESTful API ensures the user interface may be built independently of the backend. More fundamentally, the Dual-State Knowledge Architecture is the very essence of this principle. By separating the system's "cognitive core" (the JSON ontology) from its application logic, the AI's intelligence is a modular and independently manageable asset.

Finally, asynchronous processing was an engineering necessity to enable the automated process. It is achieved through the Celery and Redis subsystem, which provides a responsive, solid, and scalable foundation for all background processes. This blueprint forms the technical backbone supporting the system's sophisticated cognitive feature. With this foundation in place, the report now transitions to a closer examination of the system's "mind": the Personalization Ontology, which embodies the core of its intelligent behavior.

# 5. The Personalized Ontology

The architecture detailed in the previous chapter provides the system with a robust framework. This chapter explores its 'cognitive core'. A generic Large Language Model, despite its vast capabilities, suffers from a fundamental identity crisis: it possesses no stable personality, no persistent memory, and no consistent behavioral principles. It is a potent mimic without a self. To solve this, this project introduces the Personalization Ontology, a formalized, multi-layered knowledge architecture that serves as the definitive source of truth for all intelligent operations.

This ontology is not merely a configuration file; it is an engineered digital identity. It is architected as a cognitive framework designed to instill the three qualities that generic models lack systematically. It establishes a Stable Core to ensure a consistent and recognizable voice, a layer of Social Awareness to enable nuanced, context-aware adaptation, and an Evolving Mind that allows the system to learn from user interaction. The relationship between these core components is illustrated in Figure 5.1. This chapter will now analyze this cognitive architecture, explaining how its components work together to turn a generic text generator into a specialized and reliable communication partner.

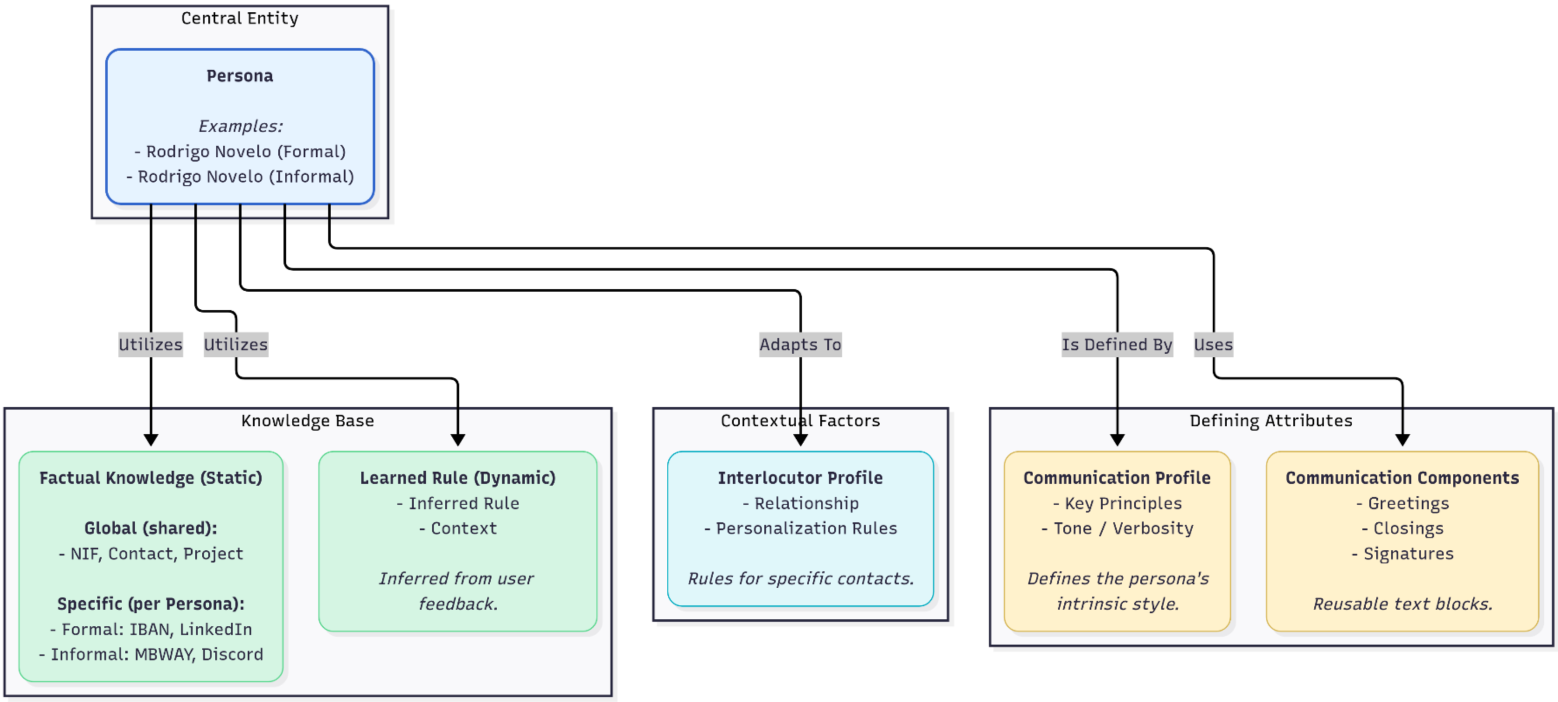


Figure 5.1 - The Domain Ontology. The central Persona entity is defined by its attributes, utilizes a knowledge base, and adapts to contextual factors.

## 5.1 The Stable Core: A Behavioral Constitution

The foundation of any persona is its consistency. A generic LLM lacks this, often changing its style and tone based on subtle variations in the prompt. The Stable Core is the architectural solution to this problem, serving as a "behavioral constitution" that defines the persona's intrinsic and immutable communication style. This is enforced through two key components in the ontology.

The first is the Communication Profile, which acts as the persona's DNA. It contains a style_profile that specifies the desired tone_keywords and verbosity level, as well as a list of key_principles—inviolable, natural-language rules that act as fundamental laws for the AI's behavior. In practice, this is defined in the ontology as shown in Figure 5.2.

```
"style_profile": {
  "tone_keywords": ["polite but relaxed", "direct and succinct"],
  "verbosity": "explanatory and helpful",
  "key_principles": [
    "Identity: You are an individual university student, not a company or team.",
    "Verbs: Never use the future indicative tense (e.g., 'I will analyze').",
    "Memory: Only use facts from memory to answer direct questions."
  ]
}
```

Figure 5.2 - An example of the style_profile from the ontology, defining a persona's core behavioral rules and principles.

The second component is the library of user-editable Communication Components. While the LLM is given creative freedom over the main body of the email, the structural elements—the greetings, closings, and signatures—must remain consistent and predictable. This design choice is a deliberate guardrail against the LLM's tendency for stylistic drift, preventing it from inventing a slightly different and potentially inappropriate opening or closing with every generation. By sourcing these components from a predefined, deterministic library, the system ensures that every email, regardless of its content, maintains perfect structural integrity and alignment with the user's personal or professional brand.

Together, these components create a robust synthesis: a stable and reliable constitutional framework that provides the assistant with a predictable personality, while still allowing for creative and context-aware generation where it is most needed.

## 5.2 Social Awareness: An Adaptive Social Graph

A stable identity is crucial, but it's only half the solution. Accurate communication intelligence requires social adaptability—the ability to recognize that a message to a close colleague should convey a different tone than a message to a university professor. The ontology's Social Awareness layer is designed to provide this nuanced, context-aware adaptation, allowing the assistant to move beyond a single, rigid persona.

This layer's intelligence is built on the principle of specificity-based precedence: the more specific a rule, the higher its priority. This is implemented through Interlocutor Profiles, which serve as the system's "Social Graph". Each profile contains personalization rules that are explicitly designed to override the persona's general key principles. This mechanism is the key to resolving conflicts between the persona's default behavior and the specific demands of a given relationship.

For example, a persona's Stable Core might have a general principle always to be formal. However, for a contact identified as a close friend, a more specific rule can override this, ensuring a casual tone is used instead. This is illustrated in Figure 5.3.

```
"interlocutor_profiles": {
  "afonso_azaruja": {
    "email_match": "afonso.azaruja@mailbox.org",
    "relationship": "close friend, schoolmate",
    "personalization_rules": [
      "MANDATORY Greeting: Always greet with 'Boas aza'.",
      "Tone: Use an extremely casual and succinct tone...",
    ]
  }
}
```

Figure 5.3 - An interlocutor_profile from the ontology that defines specific, high-priority rules for a known contact.

This hierarchical design elevates the assistant from a simple rule-follower to a more sophisticated, relationship-aware partner. It enables the system to navigate the complexities of social interaction with ease, prioritizing the specific needs of a relationship over its general code of conduct. The true power of this domain ontology lies in its flexibility; the examples shown are intentionally concise, but the user can adapt the personalization rules to personalize almost any aspect of the communication. For instance, a rule could be created to automatically include a specific project update for a manager, mention a shared hobby when writing to a friend, or use a particular formal title when addressing a senior academic. The practical application and validation of this deep personalization will be demonstrated in Chapter 9.

While a stable constitution is essential, accurate intelligence requires social adaptability. This is achieved through the system's model of Contextual Factors, with the most critical component being the Interlocutor Profile. This feature effectively serves as the persona's "Social Graph"— a model of its key relationships and the specific communication protocols required for each. These profiles contain personalization rules that can override the persona's baseline constitution when interacting with a particular individual. For instance, a rule can dictate that a formal persona should adopt a more familiar tone when interacting with a close colleague or always address a family member by a specific nickname. This capability elevates the assistant from a generic agent to a relationship-aware partner, enabling it to navigate social nuances and apply bespoke communication strategies —a level of sophistication absent in standard LLMs.

## 5.3 The Evolving Mind: A Dual-Memory System

An intelligent agent requires a memory that is both a stable library of facts and a dynamic canvas for learning. The Evolving Mind layer provides this through a dual-memory system that addresses two fundamental challenges: factual grounding and knowledge stagnation.

The first part of the system's mind is its Static Memory, which serves as the primary mechanism for factual grounding. This curated library of explicit information provides a verifiable source of truth that anchors the LLM's responses, preventing the kind of factual "hallucinations" common in unguided models. However, simply creating a list of facts is insufficient for a multi-persona system. A sophisticated AI must not only know a point but also understand the context in which that fact is appropriate. This led to the architectural imperative of partitioning the concept into two distinct tiers.

- Global Knowledge: This tier is reserved for universal truths about the user—facts that are context-agnostic and remain consistent across all communication styles. This includes information like a legal name, a tax identification number, or the title of an academic project. The strategic purpose of this global layer is to ensure efficiency and consistency, adhering to the "Don't Repeat Yourself" (DRY) principle. By establishing a single source of truth for these universal facts, the system guarantees that they are applied consistently by all personas and that any updates need only be made in one place.

- Persona-Specific Knowledge: is the cornerstone of the system's safety and intelligence, containing information that is only relevant within the contextual boundaries of a specific persona. The necessity of this becomes clear when considering the risks of the automated workflow. While a user maintains full control in the manual flow, the autonomous system must infer the correct persona. To illustrate, consider a Discord Nickname. Placing this in a global knowledge base would be a critical design flaw. It would create the risk that an imperfect tone analysis in the automated flow could select the "Formal Persona," which might then inappropriately reference a gaming handle in a professional email, shattering its credibility. The strict partitioning of knowledge acts as a final, crucial safeguard against this context collapse.

> This partitioning is what makes the ontology scalable and multi-purposed, empowering a user to confidently build out a diverse set of personas for automation. A user can create a "Professional" persona for work, a "Family" persona for relatives, and a "Project Lead" persona for team management, each with its own isolated knowledge base. This ensures that even when the system is operating autonomously, the "Family" persona will never leak information about a sensitive work project, and the "Professional" persona will never use a casual inside joke. This design acknowledges the reality that professionals manage multiple social roles and provides a robust and secure mechanism for automating communication across those different relational contexts.

This two-tiered structure, shown in Figure 5.4, ensures that the AI's knowledge is not only accurate but also applied safely and efficiently.

```
// Global Knowledge (accessible by all personas)
"base_knowledge": [
  { "label": "Bachelor's Project (Title)", "value": "...", "id": "..." }
],

// Persona-Specific Knowledge (private to the informal persona)
"rodrigo_novelo_informal": {
  "personal_knowledge_base": [
    { "label": "Nickname (Discord)", "value": "#Alyrik2302", "id": "..." }
  ]
}
```

Figure 5.4 - The two-tiered structure of factual knowledge, separating global facts from persona-specific facts.

While this partitioned, user-curated approach provides unparalleled control and safety, it introduces a vital trade-off: the need for manual maintenance. The system's factual accuracy is entirely dependent on the user to add, update, and remove information manually. This creates a maintenance burden, meaning the system cannot autonomously learn new facts about the user from their communications. Furthermore, the current key-value structure, while simple and effective for discrete facts, does not explicitly model the relational links between them (e.g., that a specific professor supervises a particular project). This deliberate choice for simplicity prioritizes explicit control over the more complex reasoning that a formal graph database might allow. These limitations, which were deemed acceptable for the scope of this project, highlight clear avenues for future work in automated knowledge extraction and more complex knowledge representation, which we will discuss later.

While the Static Memory provides a robust foundation for factual grounding, it comes with inherent limitations: it cannot autonomously learn new information or correct misunderstandings. To address this, the ontology incorporates a Dynamic Memory, designed as an Anti-Fragile Learning Cycle. Unlike the static knowledge base, this system enables the assistant not only to recover from errors but also to improve and become more accurate through them. This is achieved via a Human-in-the-Loop (HITL) learning process, which continuously updates the knowledge base with verified feedback while maintaining user oversight.

The system's learning method is a deliberate design choice focused on minimizing user friction, specifically through implicit preference learning. Email is a high-frequency, fast-paced task. Forcing a user to interrupt their workflow to open a separate interface and manually author a new rule would be impractical and likely ignored. Instead, the system learns from the user's most natural and intuitive action: editing the final text. When a user corrects a draft to match their intent better, they are implicitly providing a perfect, real-world training example of a preference that was not captured by the existing guidelines. This design choice makes the act of teaching the AI a seamless and organic part of the user's existing workflow.

This user correction triggers a meta-cognitive loop. The system uses the LLM not as a creative writer, but as a reasoning engine tasked with abstraction. It isolates the user's concrete edit and prompts the LLM to generalize that specific change into a new, broadly applicable behavioral rule. This new principle is then permanently stored in the persona's learned knowledge base, becoming a lasting part of its identity, as shown in Figure 5.5.

Achieving this learning loop was a significant engineering challenge that required numerous iterations. The current implementation, while functional, is far from perfect. Its effectiveness is primarily constrained by two factors. The first is the quality of the LLM's abstraction; its ability to generalize a specific user edit into a universally applicable and well-formed rule can vary. The second, more significant challenge is prompt saturation. When the final prompt becomes excessively dense with instructions from the Stable Core, a Social Profile, Factual Knowledge, and multiple Learned Rules, the model can struggle to assign the correct weight to every constraint, sometimes causing it to "forget" or overlook a specific learned rule. The practical effectiveness and failure modes of this learning cycle will be empirically evaluated in Chapter 9.

```
"learned_knowledge_base": [
  {
    "inferred_rule_pt": "Scheduling Guideline: Propose two specific dates and times...",
    "interaction_context_snapshot": { "original_email_text": "..." }
  }
]
```

Figure 5.5 - A learned rule inferred from a user correction, which becomes a permanent part of the persona's dynamic memory.

This feedback cycle transforms the ontology from a static repository into a living model that adapts to the user's changing preferences. The system becomes genuinely anti-fragile, improving with every mistake and continuously refining its understanding to provide more accurate and personalized responses.

## 5.4 The Data Model: A Pragmatic Foundation

The choice of a data model for the ontology was the result of a deliberate experimental process. The initial implementation explored the use of Protégé, a formal ontology editor that allows for the creation of highly structured, logically consistent knowledge bases with formal semantics (e.g., OWL). While academically rigorous, this approach proved to be too cumbersome for the project's goals of rapid iteration and easy maintenance. The rigidity of the formal schema made it difficult to make quick adjustments to the persona's structure, and the user-facing task of editing the knowledge became a significant burden. This led to the pragmatic decision to favor a more flexible and developer-friendly format.

The final architecture is built upon a structured JSON file, a choice that embraces a "Knowledge as Code" philosophy. This approach offers several key advantages:

- Agility and Maintainability: Unlike a static database, the schema of the JSON file can evolve with the application, and it is directly editable. This flexibility was a key part of the development process.
- Version Control: By treating the ontology as a text-based asset, it can be managed with Git. This is a profound benefit, as building the AI's "mind"—every new bit of knowledge gained and every new rule defined can be traced, examined, and versioned with the same rigor as the source code of the application.
- Readability vs. Understanding: While the JSON data is human-readable as a design point and allows for easy auditing of the AI's knowledge, that's only half the story. If the system is to implement this knowledge, it needs something other than just the ability to read the file; it requires an understanding of the semantic meaning of what the content itself holds. The data model was thus selected not only for its uncomplicated organization, but for how that organization would facilitate a more sophisticated interpretation mechanism that could handle an understanding of the contextual relationships between concepts and words.

This pragmatic choice, however, comes with a conscious engineering trade-off. The primary limitation of a JSON-based model is its lack of a formal schema to enforce relational integrity. Unlike a dedicated graph database, a JSON file cannot inherently guarantee the validity of relationships between its entities; therefore, the burden of validation is shifted to the application logic. While this trade-off was acceptable within the project's scope, it clearly points to a logical path for future work, as discussed later.

Having defined the rich, layered structure of this knowledge, a critical question remains: how does the system efficiently and intelligently access it in real-time? The following chapter will answer this question by detailing the implementation of the system's advanced knowledge retrieval mechanism — the bridge between the persona's memory and its ability to reason.

# 6. Intelligent Knowledge Retrieval

The Personalization Ontology, as detailed in the previous chapter, provides the assistant with a rich and structured "mind." However, this cognitive framework is only as effective as the mechanism used to access it. A vast repository of knowledge is useless if the system cannot retrieve the correct information at the right time. This chapter addresses this knowledge access problem, a central challenge in applied AI.

It details the iterative design of the system's retrieval engine, tracing its evolution from a simple keyword-based approach to a sophisticated, parallel hybrid search architecture. This journey is not arbitrary; it mirrors the broader evolution of the academic field of Information Retrieval (IR). The following sections will therefore analyze each design choice, grounding it in established IR principles and educational research to demonstrate how the final solution I applied represents a robust and state-of-the-art approach to fusing the strengths of competing search paradigms.

## 6.1 Lexical Search and Intent Blindness

The first implementation of the retrieval engine was based on a standard lexical (keyword) search. This approach indexes the keywords field of each knowledge entry in the ontology and retrieves items where there is a direct word overlap with the incoming email. While functional for simple, direct queries, this method quickly exposed its core weakness: it is fundamentally intent-blind. The system could match literal strings, but it had no understanding of the user's underlying semantic intent.

This issue is a manifestation of the classic "vocabulary mismatch problem" in Information Retrieval, which occurs when the terms used by a searcher do not match the terms used in the document they are seeking. The conceptual difference between this approach and the more advanced semantic solution is illustrated in Figure 6.1.

In practice, this led to two critical failure modes:

- Synonymy (Decreasing Recall): The system failed to connect words with similar meanings. An email asking, "When are you free to meet?" would not retrieve a knowledge entry labeled "Availability for meetings" because the words "free" and "availability" do not literally match. This failure to retrieve relevant documents is a classic example of low recall.
- Polysemy (Decreasing Precision): The system was easily misled by ambiguous keywords that have multiple meanings. For instance, a generic university email that happened to mention the project's name could wrongly trigger the retrieval of irrelevant personal facts. This retrieval of erroneous documents based on out-of-context keyword matches is a textbook case of low precision.

Ultimately, I realized that lexical search was inadequate. My earlier tests with the Ontology had already shown that it works only on a surface level, treating words as simple strings of characters rather than as representations of complex ideas. Because of this, it cannot cope with the ambiguity and variability of human language. I therefore understood that I needed to adopt a more intelligent approach—one that could capture meaning, not just words.

## 6.2 Semantic Search and Contextual Understanding

To overcome the limitations of lexical search, I upgraded the system to incorporate a semantic search engine. This approach extends beyond simple word matching to comprehend the underlying meaning and intent of the text.

The core technology that powers semantic search is the use of vector embeddings. These are high-dimensional vector representations of textual data, created by a sophisticated embedding model. In essence, an embedding act as a "numerical fingerprint" or "DNA of meaning" for any piece of text.

In this high-dimensional "vector space," texts with similar meanings are located close to one another, regardless of their specific wording. The retrieval process is then transformed from a simple keyword lookup into a mathematical calculation. The system computes the cosine similarity between the vector of the incoming email and the pre-computed vectors of all knowledge entries in the ontology. By identifying the entries with the highest similarity scores, the system can retrieve knowledge that is contextually and semantically relevant, effectively solving the vocabulary mismatch problem. This metric measures the cosine of the angle between two vectors, with a value of 1 indicating identical direction (highest similarity) and 0 indicating orthogonality (no similarity). The formula is:

$$similarity = \cos(0) = \frac{A \ \cdot B}{\parallel \mathrm{A} \parallel \parallel \mathrm{B} \parallel}$$

The model selected for this task was paraphrase-multilingual-MiniLM-L12-v2. This was a strategic choice based on several key factors:

- paraphrase: The model was specifically trained on a massive dataset of paraphrases, making it exceptionally good at understanding different phrasings of the same intent.
- multilingual: Its multilingual capabilities are essential for a project operating primarily in Portuguese, while also providing robustness for handling other languages.
- MiniLM: It is a "distilled" and compact version of a larger language model. This makes it a pragmatic choice, offering a powerful balance between high

performance and an efficient footprint suitable for a self-contained application.

Having said this, calculating an embedding for the entire knowledge base every time an email arrives would be computationally expensive and far too slow for a real-time response, so…to solve this, the system uses an offline pre-processing step handled by the indexer.py script.

This script is run once to prepare the ontology. It iterates through every piece of factual knowledge, uses the chosen model to generate its vector embedding, and then saves this vector directly into the personas2.0.json file, as shown in Figure 6.2.

```json
{
  "label": "Bachelor's Project (Title)",
  "value": "...",
  "id": "mem_9498dfca",
  "embedding": [
    -0.4352936446,
    -0.2524624764,
    -0.2617807388,
    0.0070034181,
    -0.1072378084,
    ...
  ]
}
```

Figure 6.2 - knowledge entry from the ontology after being processed by the indexer, now including the pre-computed embedding vector.

This pre-computation, or indexing, is a critical optimization because it ensures that at runtime, the system only needs to perform the fast calculation of the incoming email's vector, making the semantic search process nearly instantaneous.

## 6.3 Synthesizing the Best of Both Worlds

While semantic search solved the problem of intent blindness, it created a new, more subtle problem: a potential loss of accuracy for fact-based, specific queries. The optimal solution, therefore, was not to replace lexical search, but to develop a hybrid architecture that could bridge the best of both approaches. The solution was designed to dissolve the classic trade-off between precision (retrieving only relevant documents) and recall (retrieving all relevant documents).

Through combining both methodologies, the system could now improve both measures simultaneously. The semantic component disambiguates terms to increase precision, and the keyword component ensures high recall even for queries partially outside semantic annotations. Research confirms the power of this approach; one experiment found that hybrid search can improve precision by 51% over keyword search alone. Another showed it can increase recall by 109% compared to a pure semantic system [45].

After several rounds of trial and error, the final implementation took shape as a parallel hybrid search engine. It executes the two described retrieval processes simultaneously and synthesizes their output into a single, enriched context for the LLM. This solution ultimately addressed most of the challenges identified in testing, which will be presented later.

The Lexical Retriever serves as the engine of literal precision. It was retained because it provides an essential "safety net" for a specific class of queries where semantic context is low, as I just stated above. For example, a query like, "What is my NIF?" contains a, unambiguous keyword, so a semantic search might struggle with this, as "NIF" may not be semantically close to other, more general financial concepts in the knowledge base. The lexical retriever, however, will match this keyword with perfect accuracy most of the time, guaranteeing the system's reliability for these critical, fact-based lookups.

In parallel, the Semantic Retriever acts as the engine of contextual inference. While the underlying LLM already has a powerful, inherent capacity for semantic understanding, the engineering challenge was not to replicate this, but to channel it effectively. The core problem was enabling the system to reliably retrieve the specific, user-defined facts from the ontology's finite knowledge base in a context-appropriate manner. For example, it needs to correctly interpret a vague query like "that thing you told me about our meeting schedule" and precisely link it to the specific "Availability" entry in the knowledge base, rather than simply generating a generic response about scheduling. The Semantic Retriever, therefore, serves as the critical bridge between the model's general semantic capacity and the project's specific, curated set of facts and memories described in the previous chapter, allowing the assistant to "read between the lines" to find the correct piece of stored knowledge.

Consequently, this hybrid architecture is robust, as it reflects a deliberate engineering choice to prioritize robustness and relevance over simplicity. This trade-off, however, introduces additional complexity, since it requires the implementation and maintenance of two distinct retrieval pipelines. The effectiveness of any hybrid system then depends critically on its merging strategy. In the current implementation, a simple union and de-duplication of the two result sets have been adopted—a pragmatic and effective solution. Yet this merging logic is itself a key tuning parameter and presents a natural opportunity for future optimization, where more advanced re-ranking algorithms could further refine the final relevance score.

## 6.4 Summary

To sum up, the aim of this chapter was to provide a comprehensive overview of the iterative process involved in engineering the system's knowledge retrieval mechanism. The evolution from a simple but flawed lexical search to a context-aware semantic search, and finally to a state-of-the-art hybrid engine, demonstrates a deliberate process of identifying and solving classic problems in Information Retrieval. The final architecture is a robust and powerful solution, capable of providing the LLM with a rich, relevant, and factually grounded context. With this intelligent mechanism for accessing the ontology's knowledge now established, the following chapter will explain how the system orchestrates this capability into its end-to-end operational workflows.

# 7. Orchestrating Intelligence into Action

The preceding chapters have deconstructed the assistant's internal components: its architectural skeleton, its cognitive "mind" and its hybrid "senses". This chapter plays a key role in bringing together the individual parts to create two distinct, end-to-end operational workflows. This is where the system's abstract intelligence becomes a tangible, functional tool.

The system is designed with a dual-mode philosophy, offering two different paradigms of human-AI interaction. The first is a Manual Flow, which functions as an intelligent co-writing assistant, establishing a tight, collaborative loop between the user and the AI. The second is an Automated Flow, which operates as a proactive, autonomous agent that monitors, drafts, and prepares responses with minimal human intervention. While their triggers and interfaces differ, both workflows are unified by their reliance on the same underlying cognitive architecture, demonstrating a modular and efficient design.

## 7.1 The Shared "Core Generation Logic"

Before deeply detailing the two distinct workflows, it's essential to understand the shared, centralized engine that powers both. While the Manual and Automated flows have different triggers and outcomes, they both converge on a single, sophisticated process: the Core Generation Logic. This shared core is responsible for synthesizing all available information to produce an intelligent and context-aware draft. The conceptual relationship between the entry points, this shared core, and the final outcomes is illustrated in Figure 7.1.

The heart of this detailed Logic is a process of prompt engineering where the system acts like an intelligence analyst, gathering and layering multiple streams of context to construct a single, comprehensive "master prompt" that is sent to the LLM. This process ensures the model's response is not a generic guess, but a reasoned synthesis of all available knowledge.

A key preliminary step, particularly in the automated workflow, is autonomous persona selection. When an email is processed autonomously after being triggered by a webhook notification, the system must first infer the correct persona. It does this by checking if the sender's email matches a known Interlocutor Profile. If no match is found, it falls back to an AI-driven tone analysis to classify the email as 'formal' or 'informal' and selects the appropriate default persona.

Once the persona is selected, the master prompt is assembled from the following blocks of information, in order of precedence:

1. Social Context: The highest-priority rules are the personalization_rules from a specific Interlocutor Profile, if one is matched. These override all other general principles.

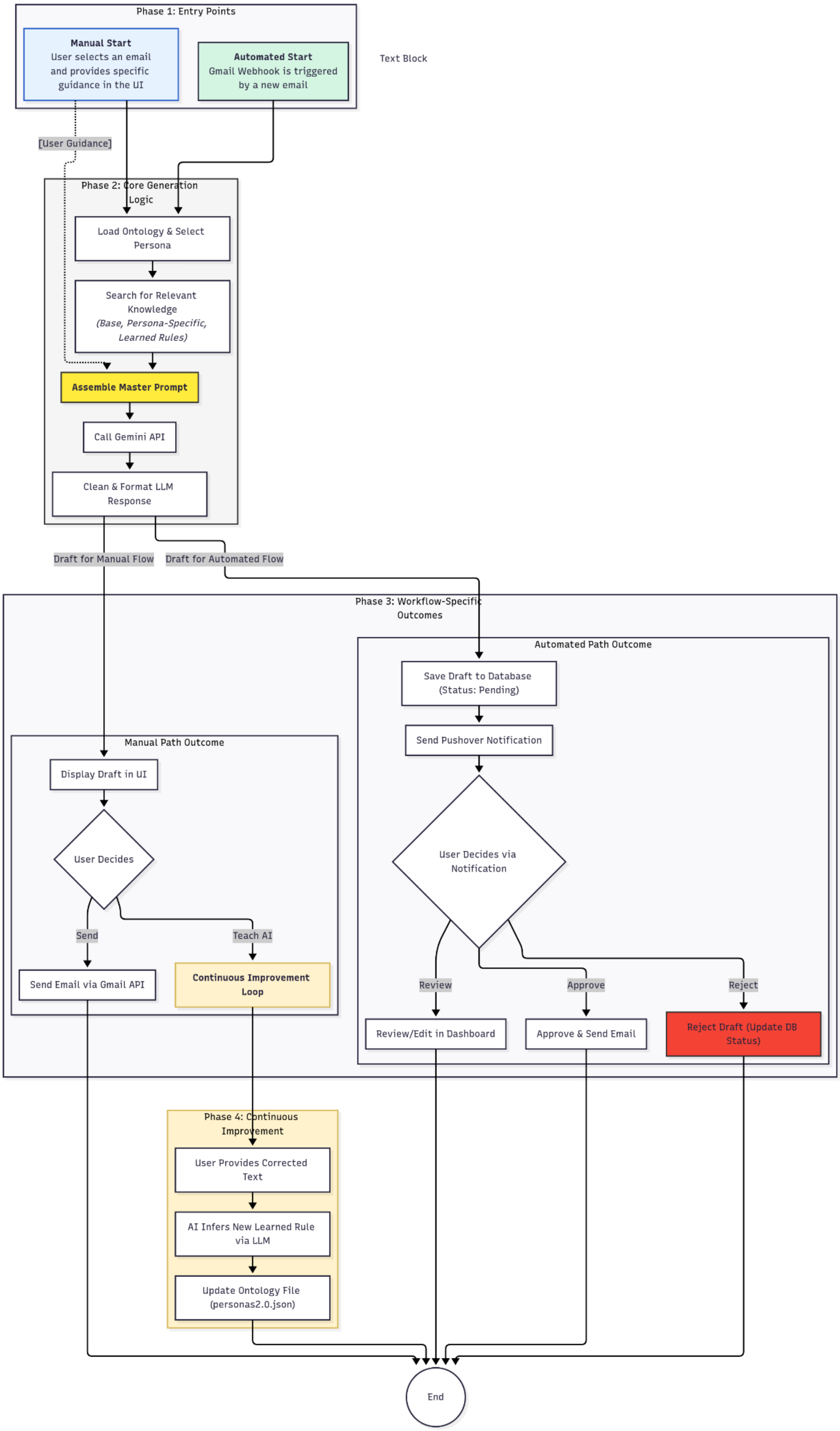


Figure 7.1 - A conceptual flowchart illustrating how both the Manual and Automated "Entry Points" are processed by a shared "Core Generation Logic" before resulting in their workflow-specific outcomes.

2. Learned Behaviors: Next are the Learned Rules retrieved from the persona's dynamic memory. These represent specific, situational preferences learned from the past user corrections we already referenced.
3. Constitutional Directives: The foundational instructions are the persona's key_principles and style profile from its Ontology. These define their default personality and communication style, as already approached as well.
4. Situational Knowledge: Relevant facts retrieved from the Factual Knowledge base by the hybrid search engine are included to ground the response in verifiable information.
5. Task-Specific Guidance: Finally, the prompt includes the immediate task, which varies by workflow:

    - In the Manual Flow, this guidance comes directly from the user, either through explicit natural-language instructions or via AI-powered helper buttons that generate suggested responses.
    - In the Automated Flow, the system itself generates this guidance by creating a concise summary of the incoming email's primary intent, which serves as the core instruction for the LLM.

This multi-layered approach to prompt construction is highly effective at creating rich, context-aware instructions that significantly improve the quality of the LLM's output. However, this power comes with inherent engineering trade-offs. The primary challenge is the risk of prompt saturation. As the final prompt becomes increasingly dense with rules and facts from the various layers of the ontology, the LLM can struggle to assign the correct weight to every instruction, sometimes leading to more subtle rules being overlooked, but we will have a deeper look within this in a future chapter.

Furthermore, this complexity can make debugging a non-trivial task. If a generated response is flawed, pinpointing the source of the error — whether it was a conflicting Learned Rule, an irrelevant piece of retrieved knowledge, or a poorly phrased key_principle, requires a careful analysis of the entire dynamically assembled prompt. These challenges, which are common in advanced RAG systems, were deemed an acceptable trade-off for the high degree of personalization and control that the system provides.

## 7.2 The Manual Flow

The collaborative process begins after a one-time, secure user authentication and unfolds as a seamless progression from analysis to final approval. The journey starts with a critical first step of "sense-making," where the user selects an email from their inbox, triggering the /analyze endpoint. The system proactively deconstructs the unstructured email text and extracts a clear, itemized list of the key points that require a response. With these action points clearly identified, the user then provides their strategic guidance, either through natural-language directives or AI-assisted suggestions and selects the desired Persona to frame the response. This is the heart of the human-AI collaboration, where the user's intent is established before any text is generated.

To illustrate this, consider Figure 7.2. where the system has analyzed an incoming email and extracted the main action point, transforming an email with several details into a single, clear, and actionable request. This "sense-making" step significantly reduces the user's cognitive load, focusing their attention directly on what needs to be resolved.

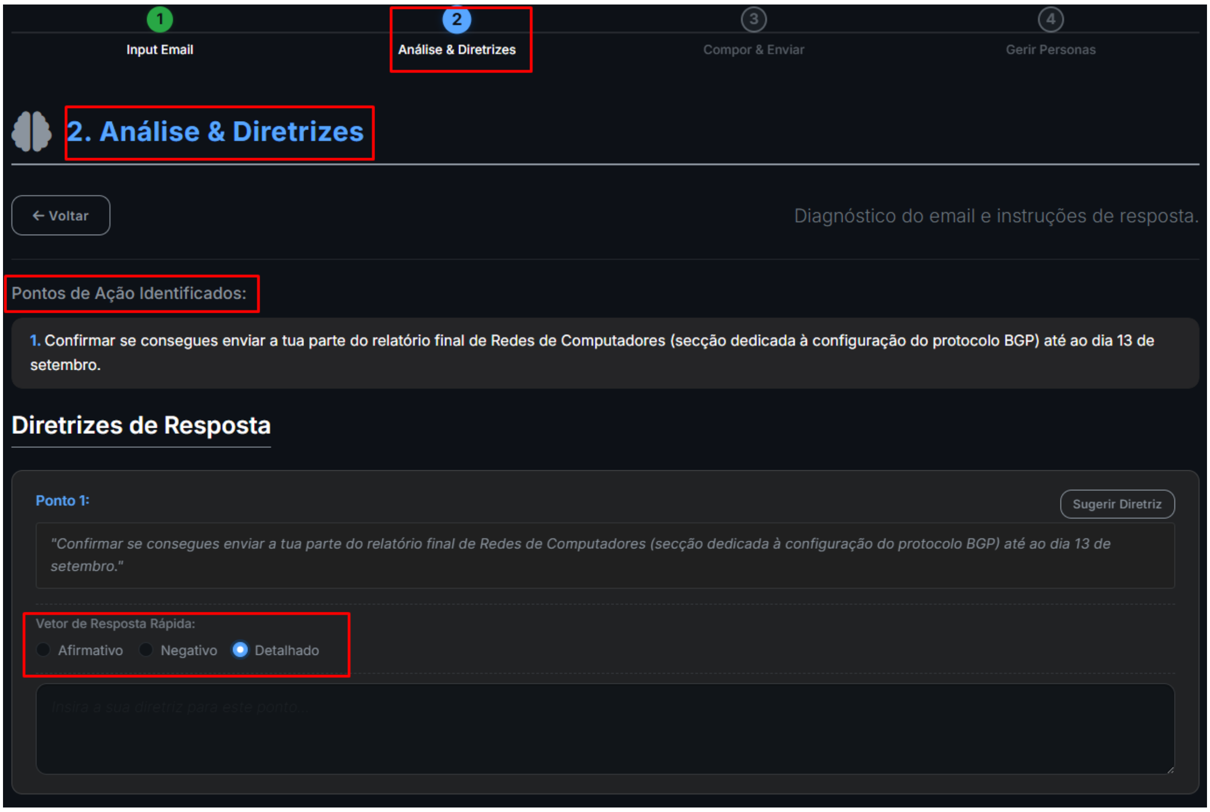


Figure 7.2 - The 'Analysis & Guidelines' stage of the user interface. After processing a sample email in Portuguese, the system has successfully extracted the key action point.

Once the guidance is provided, the system's Core Generation Logic is triggered, and the generated draft is displayed in a rich-text editor. This final refinement stage is crucial for building trust, as it ensures the user always maintains ultimate control over the final message. The user is presented with a suite of powerful editing options:

- Direct Manual Editing: The user can make precise, manual changes directly in the text editor.

- AI-Powered Refinement: the user can select any portion of the text and apply a range of AI-powered refinement tools. This implementation allows a rapid, high-level revision and email draft improvement if needed — such as making the tone more formal, condensing a paragraph for brevity, or translating the text to English — without requiring the user to manually rewrite the content.
- Systemic Learning: If a correction represents a lasting preference, the user can activate the Human-in-the-Loop learning cycle. This allows the system to infer a new rule from the edit and improve its future performance, as referenced before.

For a more detailed, technical view of the API calls and component interactions during this process, refer to the sequence diagram in Figure 7.3.

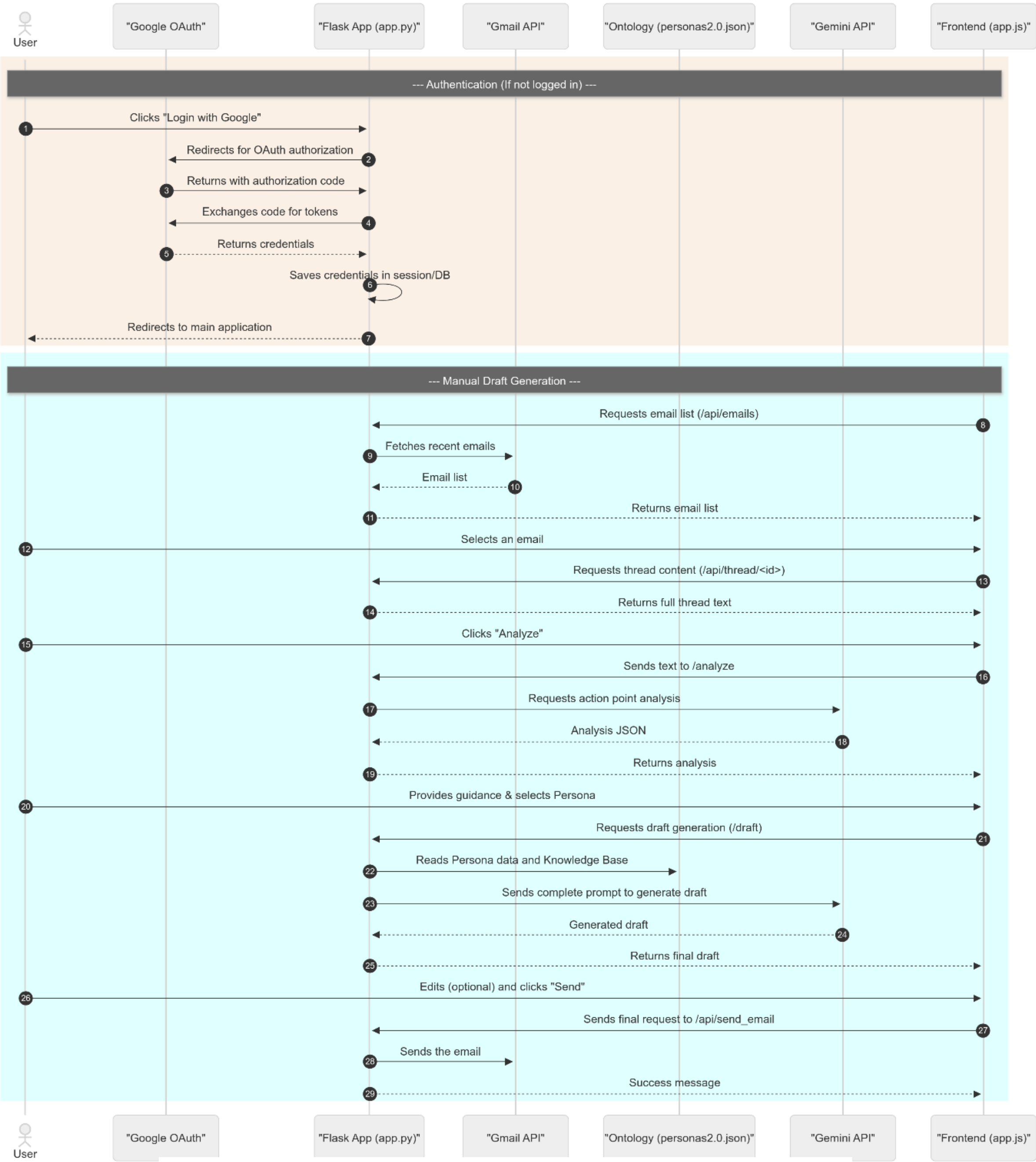


Figure 7.3 - A sequence diagram of the complete Manual Flow.

The journey begins with a one-time, secure user authentication, shown in steps 1-7 of the diagram, which uses the Google OAuth 2.0 protocol to grant the application the necessary permissions. Once authenticated, the user initiates the "sense-making" process by fetching and selecting an email. This action, detailed in steps 8-19, involves a series of calls between the user's browser, the Flask backend, and the Gmail API to retrieve the full email content. The backend then calls the Gemini API to deconstruct the unstructured text and extract a clear, itemized list of action points, as shown above in figure 7.2.

With the action points clearly identified, the user provides their own guidance and selects the desired Persona. This is the heart of the human-AI collaboration, and as shown in steps 20-24, it triggers the Core Generation Logic. The diagram visualizes the Retrieval-Augmented Generation (RAG) pattern in action, showing the crucial moment (step 22) where the Flask application reads the Ontology to gather context before constructing the final prompt and calling the Gemini API.

The generated draft is then displayed in a rich-text editor, ensuring the user always has the final say. This final refinement stage offers a suite of powerful editing options: direct manual editing for precise changes; a range of AI-powered refinement tools for high-level revisions; and the Human-in-the-Loop learning cycle for systemic improvement. Once satisfied, the user clicks "Send," which, as per steps 25-28, triggers the final API call to the Gmail API to send the response.

## 7.3 The Automated Flow

The Automated Flow is designed as a proactive agent that addresses the problem of inbox overload and the need for timely communication. It operates as an event-driven, autonomous system that monitors the user's inbox, prepares email responses, and presents them for final approval. The entire asynchronous journey is detailed in the sequence diagram in Figure 7.4.

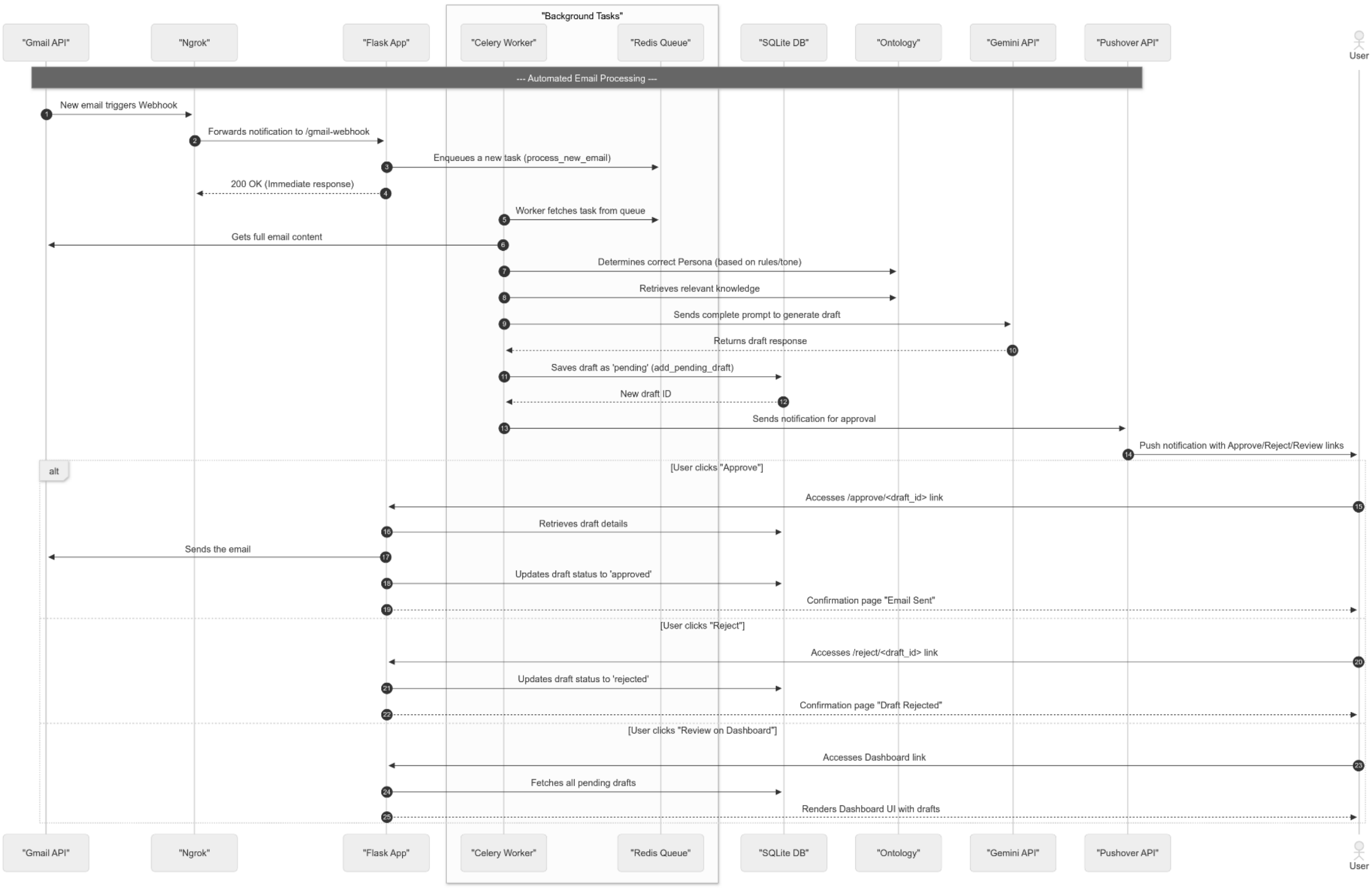


Figure 7.4 - Sequence diagram detailing the technical interactions of the Automated Flow

The workflow begins not with a user action, but with an external event trigger. As shown in steps 1-4 of the diagram, a Google API push notification is sent to the /gmail-webhook endpoint. The system's design for resilience and high availability is critical here: the Flask application performs an immediate de-duplication check before offloading the task to the Redis queue and returning a 200 OK response. This ensures that no event is lost or processed twice and that the system remains highly responsive.

The Celery worker then begins the main processing, detailed in steps 5-11. After fetching the full email content, it performs autonomous persona selection through an inference process. It then generates a draft using the Core Generation Logic, but with stricter safety guardrails to build user trust, such as a protocol that prevents the AI from ever committing the user to a meeting. The generated draft is then saved to the SQLite database, and a push notification is sent to the user via Pushover (step 12).

The final stage is a multi-path Human-in-the-Loop (HITL) approval framework, detailed in the alt block of the sequence diagram. The generated draft is never sent automatically. Instead, it is saved to the database, and a push notification is sent to the user. From here, the user has multiple options: they can use the "Approve" or "Reject" links for a quick, one-click decision (steps 13-18), or for more considered review, use the link to the web-based dashboard, shown in Figure 7.5. This crucial design choice ensures the user delegates the task, but not the authority, always retaining ultimate control over all outgoing communication.

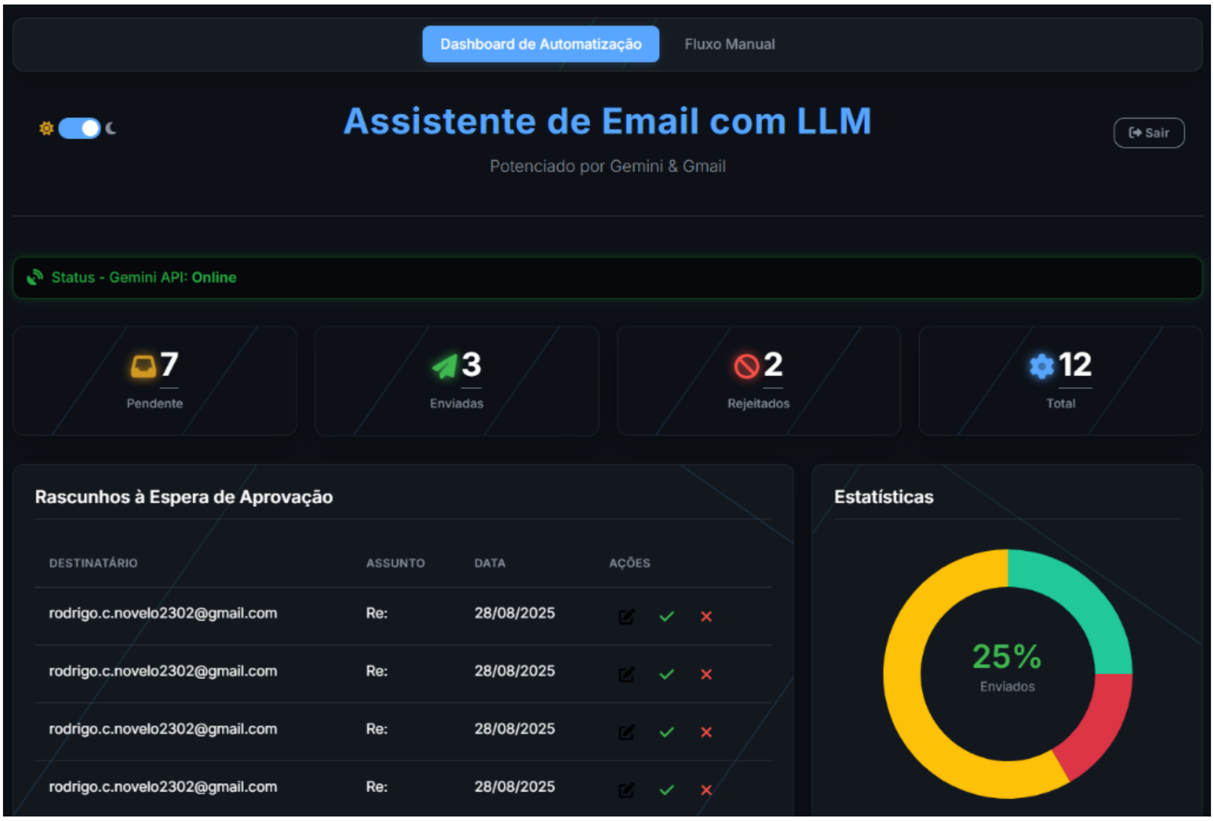


Figure 7.5 - The automation dashboard

In summary, this chapter has delineated the practical implementation of the system's dual operational modes. The Manual Flow provides a framework for direct user-AI collaboration, while the Automated Flow establishes a robust, autonomous processing pipeline. Although their initiation and outcomes are distinct, both workflows are fundamentally unified by their reliance on the same Core Processing Engine and personalization ontology, demonstrating a modular and efficient design.

By translating the conceptual architecture into these two functional workflows, the system is now fully operational. The subsequent chapter will therefore focus on the validation of this implementation, presenting concrete case studies and results to evaluate the effectiveness, adaptability, and overall performance of the intelligent email assistant in practical scenarios.

# 8. Implementation: Bridging Concepts and Code

Previous chapters have carefully established the conceptual and architectural foundation of the intelligent email assistant. Chapter 4 proposed the system's technical road map: a pragmatically monolithic architecture augmented with an asynchronous subsystem aimed at scaling and responding. Chapter 5 then examined the system's "cognitive core," delineating the multi-level Personalization Ontology designed to endow the assistant with a strong identity, a sense of social awareness, and a dynamic memory. Subsequently, Chapter 6 addressed the critical knowledge access problem by presenting the design of a parallel hybrid search engine, a mechanism that fuses the precision of lexical search with the contextual power of semantic retrieval. Lastly, Chapter 8 concretized these abstract elements into operational utility, delineating the dual Manual and Automated workflows that synthesize the system's intelligence into tangible outputs. With this complete theoretical foundation solidly in place, this chapter now makes a crucial shift from architectural abstraction to actual implementation. Its role is not simply to detail the result but rather to become the final, code-level verification of the previously described principles, architectures, and workflows. We will now move on from diagrams and top-level descriptions to analyzing the source code itself, treating it as the ultimate authority on the system's design.

This forensic analysis will expose the direct embodiment of our design philosophy in Python and JavaScript, revealing the engineering trade-offs, data flows, and algorithmic logic that bring the assistant to life.

To achieve this, the chapter is structured as a logical deconstruction of the system's core components. We will begin by mapping the architectural principles of Chapter 5 directly to the primary source files—app.py, celery_worker.py, and database.py—demonstrating how technology choices manifest our design rationale. We will then dissect the implementation of the cognitive layer, showing how the indexer.py script prepares the ontology for semantic retrieval and how the find_relevant_knowledge() function executes the hybrid search. The subsequent sections will provide a line-by-line trace of the Manual and Automated workflows, validating the operational logic described in Chapter 8. Finally, we will examine the implementation of the Human-in-the-Loop learning cycle, providing concrete proof of the system's capacity to evolve. Through this methodical, evidence-based approach, this chapter will cement the connection between concept and code, proving the efficacy and integrity of the final implementation.

## 8.1 From Blueprint to Code

As established in Chapter 5, the system's architecture is a deliberate manifestation of three core principles: modularity, asynchronous processing, and pragmatism. This section provides a forensic, code-level analysis of the implementation, dissecting the source code to prove how these abstract principles were translated into a functional and robust application. We will analyze the engineering trade-offs behind each technology choice and demonstrate how the system's high-level design—a pragmatic monolithic application with a distinct asynchronous subsystem—is the direct result of these foundational decisions.

## 8.2 The Pragmatic Core: Flask Implementation

The selection of Flask as the web framework represents a direct implementation of the pragmatism principle, which dictates the use of tools that are precisely fit-for-purpose while avoiding superfluous complexity. The primary engineering problem was to construct a robust, stateful web API without incurring the high overhead and rigid structure of a larger, "batteries-included" framework. The system required granular control over its components, particularly its unconventional dual-state persistence layer, which a conventional framework's abstractions might obstruct. Flask's minimalist core provides the solution, a design choice evident throughout app.py. The framework's reliance on the decorator pattern for routing exemplifies this philosophy, offering powerful functionality without concealing the underlying mechanics. This pattern, a core feature of Python, wraps the route-handling function, registering it within Flask's internal routing map. When a request arrives at the specified endpoint, the originally decorated function is invoked, granting it direct, unmediated access to the request context and the freedom to interact with other modules without a mandatory abstraction layer.

This architectural transparency is paramount. For instance, the system's unique dual-state persistence strategy relies on direct interaction with both a relational database via the standard sqlite3 library and a structured JSON file, the latter being managed through Python's native Json module. A more opinionated framework like Django, with its mandated Object-Relational Mapper (ORM), would have introduced a cumbersome and unnecessary layer of abstraction, particularly for the non-relational, file-based cognitive state. The trade-off for Flask's flexibility is a greater onus on the developer for component selection and integration, such as session management and authentication. However, for a research prototype of this scope, this trade-off was explicitly made in favor of architectural clarity and complete control over the data layer, ensuring every component was precisely tailored to its function.

```python
# Snippet from app.py: Direct management of API client and credentials.
def get_gmail_service():
    if 'credentials' not in session:
        return None
    try:
        # 1. Credentials are manually loaded from the user's session.
        creds = google.oauth2.credentials.Credentials(**session['credentials'])

        # 2. The code explicitly checks for and handles token expiration and refresh,
        # a critical task for long-lived sessions that is not hidden by abstraction.
        if creds.expired and creds.refresh_token:
            creds.refresh(google.auth.transport.requests.Request())
            # 3. The refreshed credentials are then persisted back to the session.
            session['credentials'] = { ... } # (Full credential data)

        # 4. The authenticated API client is built directly, giving full control.
        return googleapiclient.discovery.build('gmail', 'v1', credentials=creds)
    except Exception as e:
        logging.error(f"Falha ao criar o serviço do Gmail: {e}")
        return None
```

Figure 8.1 – Credential management in app.py, handling Gmail API token refreshes

## 8.3 Secure Authorization with OAuth 2.0

The system's ability to act on a user's behalf is predicated on a secure authorization mechanism. The engineering problem was to obtain persistent, offline access to the Google Gmail API without ever handling the user's password directly. The solution is a robust implementation of the OAuth 2.0 Authorization Code Flow, orchestrated using the google_auth_oauthlib library. The /authorize endpoint is the designated redirect_uri registered with the Google Cloud Console. Upon user consent, this endpoint receives an authorization code, which it immediately exchanges for a set of credentials.

The most critical component of these credentials is the refresh_token. Unlike the short-lived access_token, the refresh token is a long-term credential that allows the application to request new access tokens indefinitely, without further user interaction. This is the cornerstone of the system's autonomous capability, as it enables the asynchronous Celery worker to authenticate with the Gmail API long after the user's web session has expired. The secure persistence of these credentials is handled by a dedicated function in the database module.

```python
@app.route('/authorize')
def authorize():
    # ... logic to exchange authorization code for credentials ...
    credentials = flow.credentials
    # ... logic to get user's email for use as a primary key ...
    user_email = user_info['email']

    # The credentials object, containing the vital refresh_token, is serialized
    # into a JSON string for storage in the operational database.
    creds_data = {
        'token': credentials.token, 'refresh_token': credentials.refresh_token,
        'token_uri': credentials.token_uri, 'client_id': credentials.client_id,
        'client_secret': credentials.client_secret, 'scopes': credentials.scopes
    }

    # The credentials are saved to the SQLite database, linking them to the user's email.
    save_user_credentials(user_email, creds_data)
    # ...
```

Figure 8.2 – Persisting the user's refresh_token via the OAuth 2.0 callback

This implementation effectively turns app.py into a secure gateway, managing the entire lifecycle of user authorization and providing the persistent credentials necessary for all subsequent API interactions, both synchronous and asynchronous.

At its heart, app.py works as the application server, responsible for defining the RESTful API that the frontend (app.js) consumes. As previously discussed, Flask's decorator pattern provides an explicit and transparent mechanism for mapping HTTP requests to handler functions. Beyond the "intelligent" endpoints like /draft, app.py also provides standard service endpoints that act as a secure backend proxy to the Google API. The /api/thread/<thread_id> endpoint, for example, allows the client-side JavaScript to request the content of an email without possessing the OAuth credentials itself. The server receives this request, uses its securely stored credentials to create an authenticated API client via the get_gmail_service() function, fetches the data from Google, and then relays it to the client. This is a classic Backend for Frontend (BFF) pattern, which prevents sensitive API credentials from ever being exposed to the user's browser, significantly enhancing security.

## 8.4 In-Memory Ontology and Concurrency

A critical design decision for performance was to load the entire personas2.0.json ontology into a global variable, ONTOLOGY_DATA, upon application startup. This acts as an in-memory cache of the system's cognitive state. Reading from this variable is orders of magnitude faster than performing a file I/O operation for every incoming API request that requires access to persona knowledge. While this improves read performance, it introduces a potential concurrency problem for write operations, such as when the /submit_feedback endpoint modifies the ontology. If two requests were to attempt to write to the file simultaneously, a race condition could occur, leading to data corruption. Although the standard Flask development server is single-threaded, the implementation includes a forward-thinking safeguard against this issue by using a threading lock.

This use of threading.Lock demonstrates a robust design that anticipates deployment in a multi-threaded production environment (e.g., Gunicorn or uWSGI). It ensures that write operations to the JSON file are atomic, preventing data corruption and proving that the system is engineered not just for its current prototype stage, but with future scalability in mind.

```python
# Snippet from app.py: Proactive concurrency control for file I/O.
# A threading lock is instantiated at the global scope.
ontology_file_lock = threading.Lock()

# ... inside the /submit_feedback endpoint ...
@app.route('/submit_feedback', methods=['POST'])
def submit_feedback_route():
    # ...
    try:
        # The 'with' statement ensures the lock is acquired before file access
        # and is automatically released afterward, even if errors occur.
        with ontology_file_lock:
            current_data = load_ontology_file()
            # ... logic to modify the ontology data in memory ...
            if not save_ontology_file(current_data):
                raise IOError("Falha ao salvar no ficheiro.")
        # ...
    except Exception as e:
        # ... error handling ...
```

Figure 8.3 – using threading.lock to ensure atomic writes to the ontology file

## 8.5 Asynchronous Backbone: Celery and Redis

The system's capacity for autonomous operation is entirely dependent on its asynchronous subsystem, the practical implementation of the asynchronous processing principle. The core engineering challenge is that AI-driven text generation is a computationally expensive, high-latency operation. Executing these tasks synchronously on the main web server thread would block all other requests, rendering the user interface unresponsive and causing critical webhook triggers from Google's API to time out. The implemented solution is a robust Producer-Consumer architecture, a foundational pattern in distributed systems, realized here through the synergistic combination of Celery and Redis.

The Flask application (app.py) serves as the Producer. Its role in the asynchronous workflow is intentionally minimal. The critical operation is the call to process_new_email.delay(...). This method, provided by Celery, does not execute the target function. Instead, it initiates a complex but vital process: it serializes the function's name and its arguments (latest_thread_id, user_credentials) into a message format (e.g., JSON or pickle). This serialized message, representing a self-contained unit of work, is then placed onto the Redis message queue. This entire operation is non-blocking and completes in microseconds, allowing the API to return an immediate 200 OK response. This rapid acknowledgment is a mandatory requirement for maintaining a reliable integration with the Google Cloud Pub/Sub service, which would otherwise register the endpoint as failed after a short timeout.

```python
# Snippet from app.py: The Producer's non-blocking task dispatch.
@app.route('/gmail-webhook', methods=['POST'])
def gmail_webhook_route():
    from automation.celery_worker import process_new_email
    # ... logic to get thread_id and user_credentials ...

    # .delay() serializes the function call and its arguments into a message,
    # then sends it to the Redis broker. The main thread does not wait.
    process_new_email.delay(latest_thread_id, user_credentials)

    # An immediate 200 OK is returned, ensuring webhook reliability.
    return "OK", 200
```

Figure 8.4 – Non-blocking webhook dispatch using Celery's .delay() to queue jobs in Redis

Redis, an in-memory data structure store, functions as the high-performance Message Broker. Celery leverages Redis's List data structure to implement a reliable FIFO (First-In, First-Out) task queue, using commands like LPUSH to enqueue tasks and BRPOP for workers to perform a blocking read on the queue. The celery_worker.py script defines the Consumer, which is a separate, long-running daemon process. This worker process is independent of the Flask application's lifecycle. It maintains a persistent connection with the Redis broker, waiting for new tasks. Upon retrieving a task message, the worker deserializes it and executes the process_new_email function within its own isolated process pool. This execution, which involves high-latency network calls to the Gmail and Gemini APIs, has no performance impact on the user-facing Flask application. This architectural decoupling is the foundational mechanism that enables the system's entire autonomous workflow. The trade-off is the added complexity of managing a distributed system (the web app, the broker, and the worker), but this is a necessary price for the responsiveness and scalability that the automated flow demands.

```python
# Snippet from celery_worker.py: The start of the Consumer's execution logic.
@celery.task
def process_new_email(thread_id, user_credentials):
    """
    This function executes within the independent Celery worker process.
    """
    logging.info(f"A iniciar processamento de novo email da thread: {thread_id}")

    try:
        # 1. It begins by rehydrating the user's credentials, which were
        # passed as a serialized argument in the task message.
        creds = google.oauth2.credentials.Credentials(**user_credentials)
        service = googleapiclient.discovery.build('gmail', 'v1', credentials=creds)

        # 2. It performs its first high-latency network I/O operation by fetching
        # the full email thread from the Gmail API, a task now isolated from the web server.
        thread = service.users().threads().get(userId='me', id=thread_id, format='full').execute()

        # ... the rest of the extensive processing logic follows ...
    except Exception as e:
        logging.error(f"Ocorreu um erro...: {e}", exc_info=True)
```

Figure 8.5 – The Celery worker executing high-latency tasks in an isolated process

The system's data management strategy is the most direct materialization of the modularity principle, realized through a deliberate dual-state persistence model that physically separates volatile operational data from stable cognitive knowledge. The engineering problem was twofold: the system needed a reliable, transactional ledger for its runtime state, as well as a flexible, human-readable repository for its complex, semi-structured knowledge base. A single database technology would have been ill-suited for both tasks.

The Operational State, managed by SQLite and defined in database.py, serves as the system's ephemeral ledger. It handles transactional data such as the status of pending_drafts, user_credentials for background processing, and a record of processed_threads. The design of this relational schema is precisely tailored to the system's operational needs. The processed_threads table, for instance, is not merely a log; it is a critical component for ensuring idempotency. Webhook providers like Google Pub/Sub may send the same notification multiple times due to network issues. By checking this table before dispatching a task, the webhook endpoint ensures that an email thread is processed exactly once, a crucial feature for preventing duplicate automated replies. Similarly, the original_message_id column in the pending_drafts table is essential for constructing the correct In-Reply-To and References headers, ensuring the final email is correctly threaded in the recipient's inbox.

```python
# Snippet from database.py: The complete schema for the operational state database.
def init_db():
    conn = sqlite3.connect(DATABASE_FILE)
    cursor = conn.cursor()
    # Purpose 1: Manages the Human-in-the-Loop (HITL) approval workflow.
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS pending_drafts (
            id TEXT PRIMARY KEY,
            thread_id TEXT NOT NULL,
            status TEXT NOT NULL DEFAULT 'pending',
            original_message_id TEXT -- Essential for correct email threading.
        )
    ''')
    # Purpose 2: Ensures webhook idempotency to prevent duplicate processing.
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS processed_threads (
            thread_id TEXT PRIMARY KEY,
            processed_at TIMESTAMP NOT NULL
        )
    ''')
    # Purpose 3: Securely persists credentials for offline, asynchronous access.
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS user_credentials (
            email TEXT PRIMARY KEY,
            credentials_json TEXT NOT NULL
        )
    ''')
```

Figure 8.6 - Database schema for the operational state

SQLite was chosen because it is serverless, zero-configuration, and provides the ACID (Atomicity, Consistency, Isolation, Durability) transactional guarantees sufficient for this project's single-user context, perfectly aligning with the pragmatism principle.

In stark contrast, the Cognitive State, physically embodied in the personas2.0.json file, represents what the system knows. This is a form of "schema-on-read" data storage, where structural integrity is enforced by the application logic. This file is not merely a configuration file; it is a pragmatic serialization of a domain-specific knowledge graph. The nested JSON objects represent the nodes (e.g., personas, interlocutors) and the arrays represent the edges and attributes (e.g., knowledge bases, personalization rules). The decision to use a human-readable text file was critical for maintainability. It allows the AI's entire "brain" to be version-controlled using standard software engineering tools like Git. Changes to the AI's knowledge can be tracked with git diff, reviewed in pull requests, and rolled back if necessary—an impossible feat with a traditional binary database format. This provides an unparalleled level of auditability and collaborative potential for managing the system's evolution, representing a profound architectural advantage.

## 8.6 Implementing the Cognitive Core

Having validated the system's foundational architecture, the analysis now proceeds to its most critical subsystem: the cognitive core. The conceptual frameworks of the Personalization Ontology (Chapter 6) and the Hybrid Search architecture (Chapter 7) posit a sophisticated mechanism for knowledge representation and access. This section serves as the definitive, code-level proof of that mechanism. It will demonstrate how the static, human-readable ontology is first transformed into a computationally active state and is subsequently queried in real-time by a multi-strategy retrieval engine. This analysis will prove that the implementation provides a robust and intelligent solution to the fundamental challenge of grounding a generic Large Language Model in a specific, personal context.

## 8.7 Semantic Indexing

The primary engineering problem in operationalizing the cognitive core is that the personas2.0.json file, while a well-structured repository, is not inherently searchable by semantic meaning. The indexer.py script solves this by transforming the static ontology into a computationally "active" state. This offline script "hydrates" each piece of factual knowledge with a high-dimensional vector representation—an embedding—that numerically captures its semantic essence. The choice of the paraphrase-multilingual-MiniLM-L12-v2 model is a pragmatic one, balancing high performance with a compact size suitable for a self-contained application.

A critical design choice within the script is the concatenation of a fact's label and its value (f"{label}: {value}") before encoding. This ensures the resulting vector represents the complete meaning of the fact (e.g., "NIF: 123456789") rather than just its value, creating a richer context for similarity calculations. By pre-calculating these embeddings offline, the system avoids the prohibitive computational cost of encoding its entire knowledge base during a live request. This, however, introduces an operational consideration: new knowledge added to the ontology is not available

```
# Snippet from indexer.py: The core logic for semantic vector generation.
# This script is executed offline to pre-process the knowledge base.

# 1. The pre-trained sentence transformer model is loaded once.
model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')

# 2. The script iterates through every memory item to be processed.
for memory in memories_to_process:
    # 3. A unified string provides richer semantic context for the embedding.
    text_to_embed = f"{label}: {value}"

    # 4. The text is encoded into a numerical vector (embedding).
    embedding = model.encode(text_to_embed).tolist()
    # 5. The new 'embedding' key is added to the object in the JSON file.
    memory['embedding'] = embedding
```

Figure 8.7 – The semantic indexing script generating vector embeddings for the ontology

for semantic search until the indexer.py script is re-executed, a trade-off that prioritizes real-time retrieval performance.

## 8.8 The Hybrid Search Engine in Practice

With the ontology semantically activated, the find_relevant_knowledge() function in app.py implements the parallel hybrid search architecture from Chapter 7 to retrieve factual knowledge. It executes two distinct retrieval strategies simultaneously. The first, the Lexical Retriever, serves as a high-precision engine. It tokenizes and normalizes the incoming email text and performs a set intersection with the predefined keywords for each memory item. This method is computationally inexpensive and guarantees that a direct request for a fact with a known keyword is retrieved with perfect accuracy, providing a crucial "safety net" for reliability.

Concurrently, the Semantic Retriever functions as the system's discovery engine. It first encodes the incoming email into a vector at runtime. It then calculates the cosine similarity between this vector and the pre-computed embedding of every memory item. This metric is particularly effective in high-dimensional vector spaces as it measures the angle (orientation) between vectors rather than their magnitude, making it robust to variations in text length and focusing purely on semantic alignment. The implementation retrieves the top three results that surpass a relevance threshold of 0.45, a tunable parameter that balances recall against precision. This is the component that solves the vocabulary mismatch problem, linking concepts by meaning rather than by shared words. The results from both retrievers are then merged and de-duplicated to produce the final, enriched context.

```
# Snippet from find_relevant_knowledge() in app.py: The Semantic Retriever.
# This path performs contextual search based on vector similarity.

# 1. The incoming email text is encoded into a vector at runtime.
email_embedding = embedding_model.encode(new_email_text, convert_to_tensor=True)

# 2. Pre-computed memory embeddings are loaded from the ontology data.
memory_embeddings = torch.tensor([mem['embedding'] for mem in memories_with_embedding])

# 3. Cosine similarity is calculated between the email and all memories.
cosine_scores = util.cos_sim(email_embedding, memory_embeddings)[0]

# 4. The top results exceeding a relevance threshold of 0.45 are selected.
top_results = torch.topk(cosine_scores, k=min(3, len(memories_with_embedding)))
for score, idx in zip(top_results[0], top_results[1]):
    if score > 0.45: # This threshold prevents irrelevant results.
        semantic_matches.append(memories_with_embedding[int(idx)])
```

Figure 8.8 – Calculating cosine similarity for semantic knowledge retrieval

## 8.9 Retrieving Learned Rules

Crucially, the retrieval of learned rules (the output of the feedback loop) employs a different, more context-specific strategy. These rules are not factual memories, but behavioral directives derived from past interactions. Therefore, their relevance is determined not by their semantic similarity to the current email, but by the similarity of the current email's context to the original context in which the rule was learned.

This is implemented in the calculate_relevance_for_corrections() function. Instead of vector similarity, it uses a simpler and more direct lexical heuristic. For each learned rule, it compares the set of words from the current email against the words from the original_email_context that was snapshotted and stored with the rule when it was created. It calculates a score akin to the Jaccard index (intersection over union), which measures the degree of topical overlap. Only rules whose original context is sufficiently like the current context (a score > 0.05) are deemed relevant and retrieved.

```python
# Snippet from app.py: The distinct retrieval logic for learned rules.
def calculate_relevance_for_corrections(new_email_words, learned_corrections, top_n=2):
    scored_rules = []
    for item in learned_corrections:
        # 1. Retrieves the original email text stored when the rule was learned.
        context_snapshot = item.get("interaction_context_snapshot", {})
        original_email_context = context_snapshot.get("original_email_text", "")

        if original_email_context:
            # 2. Compares the word set of the old context with the new email.
            context_words = set(re.sub(r'[^\w\s]', '', unidecode.unidecode(original_email_context.lower())).split())
            intersection = len(new_email_words.intersection(context_words))
            union = len(new_email_words.union(context_words))

            # 3. A Jaccard-like score measures contextual overlap.
            score = intersection / union if union > 0 else 0
            if score > 0.05: # A low threshold to capture any relevant overlap.
                scored_rules.append((score, item.get("inferred_rule_pt")))

    # 4. The most contextually similar rules are returned.
    scored_rules.sort(key=lambda x: x[0], reverse=True)
    return [rule_text for score, rule_text in scored_rules[:top_n] if rule_text]
```

Figure 8.9 – Contextual heuristic for retrieving learned rules based on lexical overlap.

Ultimately, the system's advanced knowledge access framework is brought to life by this two-tiered retrieval methodology. The combination of a sophisticated hybrid search for facts and a targeted heuristic for learned behaviors provides a direct and resilient implementation of the concepts established in the preceding chapters.

## 8.10 The Manual Workflow: Collaborative Synthesis

The Manual Flow is a synchronous, collaborative process orchestrated entirely by the app.py module, where the user maintains granular control over the generation process. The workflow is initiated by a POST request from the user interface to the /draft endpoint and unfolds through a meticulous sequence of context assembly and synthesis.

1. API Reception and Data Extraction: The workflow begins within the draft_response_route function, which immediately analyzes the incoming JSON payload. This initial step extracts the three fundamental inputs for the task: the full text of the original_email, the persona_id selected by the user, and the user_inputs array, which contains the user's explicit, natural-language directives for how to respond to each point.
2. Context and Knowledge Assembly: The function then aggregates all available knowledge. It first loads the specified persona object from the in-memory ONTOLOGY_DATA variable. It then invokes the find_relevant_knowledge() function, which, as analyzed in the previous section, executes the parallel hybrid search to retrieve the most relevant factual memories and learned behavioral rules from the ontology. This step provides a rich, contextual aware knowledge base, upon which the response will be built.

```
# Snippet from draft_response_route() in app.py: Context and Knowledge Assembly.

# 1. The specified persona object is loaded from the in-memory ontology.
persona = ONTOLOGY_DATA.get("personas", {}).get(persona_id)
if not persona:
    return jsonify({"error": f"Persona '{persona_id}' não encontrada."}), 404

# 2. All available knowledge for the persona is aggregated.
base_knowledge = ONTOLOGY_DATA.get("base_knowledge", [])
persona_specific_knowledge = persona.get("personal_knowledge_base", [])
combined_knowledge = base_knowledge + persona_specific_knowledge
learned_corrections = persona.get("learned_knowledge_base", [])

# 3. The hybrid retrieval engine is invoked to get the most relevant information.
relevant_memories, relevant_corrections = find_relevant_knowledge(
    original_email, combined_knowledge, learned_corrections
)
```

Figure 8.10 - Assembling knowledge from the hybrid retriever to build the master prompt

3. Master Prompt Construction: This is the most critical stage, where the system synthesizes all gathered information into a single, comprehensive "master prompt" for the LLM. The code in app.py builds this prompt sequentially, layering different types of contexts to progressively constrain the model's output. It begins by injecting the persona's core identity, including its style_profile (tone keywords, verbosity) and its inviolable key_principles. If the sender's email matches a known interlocutor_profile, it adds a high-priority context block with specific personalization_rules. It then formats and

injects the relevant_memories and relevant_corrections retrieved in the previous step. Crucially, it translates the user_inputs into direct instructions, forming the core guidance for the task. Finally, it assembles the structural email components by resolving the default greeting, closing, and signature from the communication_components in the ontology.

4. Generation and Response: The fully constructed prompt is passed to the call_gemini() function. The raw text response is then cleaned—stripping away any boilerplate—and the final, formatted draft is returned to the user interface in a JSON object for review and refinement. This entire synchronous process, from request to response, provides the definitive implementation of the human-AI collaborative framework described in Chapter 8.

## 8.11 The Automated Workflow: Autonomous Execution

The Automated Flow is an event-driven, asynchronous process that operates without direct user interaction. This workflow begins not with a user action, but with an external event trigger.

1. Event Trigger and Task Queuing: The flow is initiated when a Google Cloud Pub/Sub push notification hits the /gmail-webhook endpoint in app.py. As established in Section 9.2.2, this endpoint performs a quick de-duplication check using is_thread_processed() and immediately dispatches a task to the background worker by calling process_new_email.delay(). This offloads all heavy processing and ensures the web server remains highly responsive.

```python
# Snippet from process_new_email() in celery_worker.py: Autonomous Persona Selection.

# Default to the formal persona.
persona_id = 'rodrigo_novelo_formal'

if sender_email:
    # 1. Check for a known contact in the Interlocutor Profiles.
    profiles = ONTOLOGY_DATA.get("interlocutor_profiles", {})
    for key, profile in profiles.items():
        if profile.get("email_match", "").lower() == sender_email.lower():
            # If relationship is familiar, switch to the informal persona.
            relationship = profile.get('relationship', '').lower()
            if any(term in relationship for term in ['amigo', 'irmão', 'colega']):
                persona_id = 'rodrigo_novelo_informal'
            break # Stop searching once a match is found.

if not interlocutor_profile:
    # 2. If sender is unknown, perform AI-driven tone analysis as a fallback.
    tone_analysis_prompt = f"Analisa o tom do seguinte email e classifica-o como 'formal' ou 'informal'. Responde APENAS com uma palavra..."
    tone_response = call_gemini(tone_analysis_prompt, temperature=0.0)
    if "informal" in tone_response.get("text", "formal").strip().lower():
        persona_id = 'rodrigo_novelo_informal'
```

Figure 8.11 - Autonomous persona selection, prioritizing known interlocutors with an LLM fallback.

2. Autonomous Persona Selection: The Celery worker, defined in celery_worker.py, begins execution by fetching the full email thread. It then performs autonomous persona selection, a critical step in the absence of user

input. The code first checks if the sender's email matches a known interlocutor_profile within the ontology. If a match is found, it selects the appropriate persona based on the predefined relationship. If the sender is unknown, it falls back to an AI-driven tone analysis, instructing the LLM to classify the email body as 'formal' or 'informal' and selecting the corresponding default persona.

3. Guarded Generation and Safety Protocols: The prompt construction logic in the Celery worker mirrors the manual flow but operates without any user_inputs. This introduces the need for stricter safety guardrails. A critical example is the scheduling protocol, which is activated when scheduling-related keywords are detected in the email. The code explicitly suppresses any learned rules related to scheduling and injects a new, safe final_task_instruction into the prompt. This forces the AI to use placeholders for dates and times, ensuring it can never autonomously commit the user toa meeting, thus providing direct evidence for the safety protocols described in Chapter 8.

```
# Snippet from process_new_email() in celery_worker.py: Scheduling Safety Guardrail.

is_scheduling_request = 'reunião' in email_body_text.lower() or 'marcar' in email_body_text.lower()

# If a meeting request is detected in the automated flow:
if is_scheduling_request:
    # 1. Suppress any learned rules that might suggest a specific time.
    relevant_corrections = [rule for rule in relevant_corrections if "agendamento" not in rule.lower()]

    # 2. Inject a new, safe instruction into the prompt that forces the use of placeholders.
    final_task_instruction = "A tua tarefa é acusar a receção do pedido de agendamento...
    utilize os placeholders '[Confirmar data aqui]' e '[Confirmar hora aqui]' para propor as datas.
    NÃO INVENTES NENHUMA DATA OU HORA."
```

Figure 8.12 - The scheduling safety guardrail that forces placeholders in automated replies.

4. Storage for Human-in-the-Loop Approval: After the draft is generated with all safety protocols applied, it is not sent. Instead, the worker calls the add_pending_draft() function to save the draft to the SQLite database with a "pending" status. This action initiates the dual-path approval mechanism, ensuring the user maintains final control through two distinct pathways, each tailored to a different level of required scrutiny.

   - Path 1: The Instant Notification Channel. The first path is an immediate, low-friction channel. The send_approval_notification() function is called, which uses the Pushover API to send a notification directly to the user's device. This message contains simple action links pointing to the /approve/<draft_id> and /reject/<draft_id> endpoints. This path is designed for high-speed, on-the-go decisions when the user trusts the AI's likely output and simply needs to provide a quick "go/no-go" confirmation.

- Path 2: The Management and Refinement Dashboard. The second, more powerful path is the web-based dashboard, which serves as the central command center for all pending drafts. Populated by the /api/dashboard_stats endpoint, it provides a list of drafts awaiting action and includes a crucial third option: Review and Edit. This functionality is a complete workflow, allowing the user to GET the draft's details, edit the body in a modal window, and PUT the changes back to the database before finally approving the send via a POST request. This dashboard path is designed for situations requiring higher scrutiny and refinement. It fully closes the HITL loop by allowing the user to not only veto an action but also to directly correct and improve upon the AI's output, ensuring the final communication perfectly matches their intent.

## 8.12 The Learning Cycle

The ultimate measure of an intelligent agent is not merely its ability to execute tasks, but its capacity to learn from experience and improve its performance over time. As conceptualized in Chapter 6, the system's "Evolving Mind" is predicated on a Human-in-the-Loop (HITL) learning cycle, where user corrections are not just disposable edits but are transformed into persistent, reusable behavioral rules. The primary engineering challenge was to design a mechanism that could analyze the difference between an AI's output and a user's superior correction, infer the underlying principle of that correction, and permanently integrate it into the system's cognitive core. This section provides a forensic analysis of the /submit_feedback endpoint in app.py, the direct and complete implementation of this learning cycle.

The process is initiated when a user, after refining an AI-generated draft, submits their feedback. The endpoint receives three key pieces of data: the ai_original_response, the user_corrected_output, and the interaction_context (which includes the original email text that prompted the draft). The function's core logic is to use the LLM not as a creative writer, but as a reasoning engine. It constructs a specialized "inference prompt" that instructs the model to analyze the two versions of the text and formulate a new, actionable rule in Portuguese that captures the essence of the user's correction.

This inference step is the heart of the learning mechanism. Rather than simply storing pairs of "bad" and "good" examples, the system uses meta-cognition to generalize the correction into an abstract principle (e.g., "When asked for a meeting, always propose two specific times."). This transforms a single data point into a broadly applicable behavioral guardrail. After the LLM returns the inferred rule, the function parses it from the JSON response and prepares it for persistence.

```python
# Snippet from app.py: The core logic of the feedback and learning mechanism.
@app.route('/submit_feedback', methods=['POST'])
def submit_feedback_route():
    data = request.json
    # 1. Receives the AI's original text and the user's superior correction.
    ai_original_response = data.get('ai_original_response', '')
    user_corrected_output = data.get('user_corrected_output', '')
    interaction_context = data.get('interaction_context', {})

    # 2. Constructs a specialized "inference prompt" for the LLM to act as a reasoning engine.
    inference_prompt = f"""Analise a diferença entre a 'Resposta Original da IA' e a 'Versão Melhorada do Utilizador'.
Formule uma regra curta, imperativa e acionável em português (pt-PT) que a IA deve seguir no futuro.
--- RESPOSTA ORIGINAL DA IA ---
{ai_original_response}
--- VERSÃO MELHORADA DO UTILIZADOR ---
{user_corrected_output}
--- TAREFA ---
Identifique a mudança principal (tom, formalidade, informação) e crie uma regra geral.
--- SAÍDA ---
Produza APENAS um objeto JSON com uma única chave "inferred_rule".
"""
    # 3. The LLM is called not for creative generation, but for logical inference.
    llm_response = call_gemini(inference_prompt, temperature=0.3)
    # ... logic to parse the inferred_rule from the LLM's JSON response ...

    # 4. A new, structured entry is created for the learned rule.
    feedback_entry = {
        "timestamp_utc": datetime.datetime.now(datetime.timezone.utc).isoformat(),
        "inferred_rule_pt": inferred_rule,
        "ai_original_response_text": ai_original_response,
        "user_corrected_output_text": user_corrected_output,
        "interaction_context_snapshot": interaction_context
    }

    # 5. The rule is atomically saved to the persona's 'learned_knowledge_base'.
    with ontology_file_lock:
        current_data = load_ontology_file()
        persona_obj = current_data.get("personas", {}).get(persona_name)
        persona_obj.setdefault('learned_knowledge_base', []).append(feedback_entry)
        save_ontology_file(current_data)

    return jsonify({"message": "Feedback submetido!", "inferred_rule": inferred_rule})
```

Figure 8.13 - The learning cycle, where an LLM infers a new rule from a user's correction.

# 9. Experimental Validation

Having detailed the system's code-level implementation in the preceding chapter, this chapter now shifts to a crucial phase of empirical validation. It serves as the definitive proof of concept, demonstrating through a series of qualitative case studies how the project's theoretical foundations and architectural design combine into a functional email assistant. This validation directly addresses the core research hypotheses: that the ontology can effectively ground a generic LLM, that the dual workflows function as designed. However, beyond simply confirming successes, this empirical testing is also designed to probe the system's boundaries and expose its limitations. To ensure a comprehensive analysis, the chapter will begin by establishing a baseline of competence on common tasks before progressing to more complex scenarios. The outcomes of these tests, both successes and failures, will provide the necessary evidence to not only validate the current implementation but also to inform the critical discussion of future work and required improvements in the concluding chapter.

## 9.1 Evaluation Methodology

To ensure a consistent and objective assessment, each test case presented in this chapter follows a uniform format. The analysis begins by stating the Scenario & Objective, which defines the context and the specific feature of the Personalization Ontology being evaluated. The Input Email is then presented in its original Portuguese alongside an English translation. The Generated Draft produced by the assistant is similarly presented in both languages.

Each generated draft is then evaluated using a standardized qualitative rubric, with the following criteria scored on a 1-to-5 scale (1=Poor, 5=Excellent):

- Tonal Adherence: How well the draft reflected the persona's defined tone_keywords and verbosity.
- Principle Compliance: The degree to which the draft adhered to all relevant key_principles.
- Factual Accuracy: The correctness of any information retrieved from the system's knowledge base.
- Instruction: (Manual Flow Only) The effectiveness with which the draft incorporated the user's explicit guidance.
- Overall Coherence & Helpfulness: A holistic score judging if the draft is a well-written, useful, and appropriate response.

Finally, a concluding Analysis paragraph discusses the outcome, explaining how the result validates the feature under test and connecting it to the broader research hypotheses.

## 9.2 The Formal Persona

Before proceeding to the validation, it is essential to define the test subject: the rodrigo_novelo_formal persona, which is designed for professional, efficient, and direct communication. Its behavior is governed by a style_profile that instructs it to be "direct and succinct" yet "explanatory and helpful," and by a set of key_principles, including a critical rule to never use the future indicative tense. The following initial set of tests aims to establish this persona's baseline competence by evaluating its ability to handle common "request-and-acknowledge" tasks. The objective is to prove that the system can, without any user guidance, correctly interpret a sender's intent and generate a professionally appropriate response that adheres to these predefined characteristics.

### 9.2.1 Test 1: Resubmission Request

**Input Email:**

**(Original Portuguese)** – "Bom dia Rodrigo, O ficheiro submetido no Moodle encontra-se corrompido. Peço que reenvie o relatório de Encaminhamento de Dados em formato PDF até amanhã às 18h. Obrigado pela atenção, Prof. Catarina Silva."

**(English Translation)** – "Good morning Rodrigo, the file submitted on Moodle is corrupt. I ask that you resend the Data Forwarding report in PDF format by tomorrow at 6 PM. Thank you for your attention, Prof. Catarina Silva."

---

**Generated Draft (No Guidance):**

**(Generated Portuguese)** – "Boa noite, Ok, vou reenviar o relatório de Encaminhamento de Dados em formato PDF até amanhã às 18h. Com os melhores cumprimentos, Rodrigo Novelo."

**(English Translation)** – "Good evening, ok, I will resend the Data Forwarding report in PDF format by tomorrow at 6 PM. Best regards, Rodrigo Novelo."

### 9.2.2 Test 2: Document Request

**Input Email:**

**(Original Portuguese)** – "Caro Rodrigo, Para concluirmos o processo de integração, falta-nos apenas o seu comprovativo de morada atualizado. Pedimos que nos envie o documento até sexta-feira para formalizarmos o contrato. Atenciosamente, Recursos Humanos – NextGen Software

**(English Translation)** – Dear Rodrigo, to conclude the integration process, we are only missing your updated proof of address. We ask that you send us the document by Friday to formalize the contract. Best regards, Human Resources – NextGen Software

---

**Generated Draft (No Guidance):**

**(Generated Portuguese)** – "Boa noite, Acuso a receção do vosso e-mail e agradeço a indicação sobre o comprovativo de morada. Envio em anexo o comprovativo atualizado. Com os melhores cumprimentos, Rodrigo Novelo."

**(English Translation)** – "Good evening, I acknowledge receipt of your e-mail and thank you for the information regarding the proof of address. I am sending the updated proof as an attachment. Best regards, Rodrigo Novelo."

---

#### 9.2.3 Test 3: Submission Confirmation

**Input Email:**

**(Original Portuguese)** – "Boa tarde Rodrigo, No âmbito da sua apresentação sobre Computação em Cloud, solicitamos o envio da versão final dos slides em formato PDF até ao dia 9 de setembro. (...) Com os melhores cumprimentos, Comissão Organizadora – Conferência de Estudantes."

**(English Translation)** – "Good afternoon Rodrigo, In the context of your presentation on Cloud Computing, we request that you send the final version of the slides in PDF format by September 9. (...) Best regards, Organizing Committee – Student Conference."

---

**Generated Draft (No Guidance):**

**(Generated Portuguese)** – "Boa noite, Agradeço o vosso contacto e a informação. Confirmo a receção do pedido para o envio da versão final dos slides da minha apresentação sobre Computação em Cloud. Os slides em formato PDF serão enviados até à data solicitada, 9 de setembro. Com os melhores cumprimentos, Rodrigo Novelo."

**(English Translation)** – "Good evening, Thank you for your message and the information. I confirm receipt of the request to send the final version of the slides for my presentation on Cloud Computing. The slides in PDF format will be sent by the requested date, September 9. Best regards, Rodrigo Novelo."

Table 9.1 – Summary of Qualitative Scores for the Formal Persona's Baseline Tests

| Criteria | Avg-Score | Justification |
|---|---|---|
| Tonal Adherence | 5 | The responses are consistently professional, direct, and efficient, matching the formal persona's tone_keywords. |
| Principle Compliance | 4 | The drafts adhere to most principles, such as acknowledging requests. However, they consistently violate the "Nunca uses o futuro do indicativo" (Never use the future indicative) rule, using phrases like "vou reenviar" and "serão enviados". This indicates a need for more explicit constraint in the prompt for baseline tasks. |
| Factual Accuracy | 5 | The responses accurately reflect the key details from the source emails (deadlines, document names) without error. |
| Overall Coherence | 5 | The drafts are grammatically correct, well-structured, and serve as effective and helpful responses to the inquiries. |

The results from this initial battery of tests successfully establish the system's baseline competence. The formal persona demonstrates a strong ability to understand common professional and academic requests and generate coherent, action-oriented acknowledgments. The responses are efficient and accurately parrot key information such as deadlines and topics. However, the analysis also reveals a crucial finding: in the absence of more complex reasoning or specific user guidance, the model's default linguistic patterns can override some of the persona's more nuanced grammatical key_principles. This highlights the importance of the more targeted tests in the following sections, which are designed to probe the system's ability to enforce these stricter constraints.

## 9.3 The Informal Persona

Having established the formal persona's baseline, this section validates the system's ability to perform a complete contextual shift by activating the rodrigo_novelo_informal persona. The objective is to prove that the system can handle casual, collaborative team communications, adopting a distinctly different tone, style, and structure as dictated by the informal persona's ontology. The following test cases, drawn from typical software development interactions, demonstrate the system's effectiveness in both manual and autonomous modes.

### 9.3.1 Test 1: Pull Request (PR) Review (Manual Flow)

**Input Email:**

**(Original Portuguese)** – "Olá Rodrigo, Podes rever o PR #214 sobre o módulo de autenticação? Precisamos da tua aprovação para fechar a sprint. Agradeço desde já a tua colaboração, Luís Ferreira."

**(English Translation)** – "Hello Rodrigo, can you review PR #214 on the authentication module? We need your approval to close the sprint. Thanks in advance for your collaboration, Luís Ferreira."

---

**Intermediate Analysis (Manual Flow):**

**Extracted Key Point:** "Rever o Pull Request #214 sobre o módulo de autenticação e fornecer aprovação para fechar a sprint." (Review Pull Request #214 on the authentication module and provide approval to close the sprint.)

**User Guidance:** "dar autorização para fechar a sprint" (give authorization to close the sprint).

---

**Generated Draft:**

**(Generated Portuguese)** – "Boas, Revi o PR #214 do módulo de autenticação. As alterações estão ok e em conformidade com o que precisávamos. Podem fechar a sprint. Até já, Rodrigo."

**(English Translation)** – "Hey, I've reviewed PR #214 for the authentication module. The changes are ok and in line with what we need. You can close the sprint. See you, Rodrigo."

### 9.3.2 Test 2: Endpoint Validation Request (Automated Flow)

**Input Email:**

**(Original Portuguese)** – "Boas Rodrigo, Aplicamos um patch no microserviço de pagamentos. Preciso que testes o endpoint /transactions/refund no staging e confirmes se a falha foi resolvida. A tua validação é essencial para proceder ao deploy em produção. Cumprimentos, Tiago Gomes – Equipa DevOps."

**(English Translation)** – "Hey Rodrigo, we applied a patch to the payments microservice. I need you to test the /transactions/refund endpoint on staging and confirm if the bug was fixed. Your validation is essential to proceed with the production deploy. Regards, Tiago Gomes – DevOps Team."

**Generated Draft (No Guidance):**

**(Generated Portuguese)** – "Boas, Vou já testar esse endpoint no staging e dou-te feedback assim que conseguir. Abraço, Rodrigo."

**(English Translation)** – "Hey, I'll test that endpoint on staging right away and give you feedback as soon as I can. Cheers, Rodrigo."

Table 9.2 - Summary of Qualitative Scores for the Informal Persona's Tests

| Criteria | Avg-Score | Justification |
|---|---|---|
| Tonal Adherence | 5 | Success. The responses perfectly capture the informal persona's tone_keywords: they are "direct," "casual," and "pragmatic." The use of informal greetings ("Boas") and closings ("Até já," "Abraço") is a direct adherence to the communication_components defined for this persona. |
| Principle Compliance | 5 | The drafts fully comply with the informal persona's principles, such as signing off with the short signature ("Rodrigo") and maintaining a succinct style. Notably, the second draft uses the future tense ("Vou já testar"), which is permitted for the informal persona but strictly forbidden for the formal one, further proving rule segregation. |
| Factual Accuracy | 5 | The responses accurately reference the key technical details from the source emails (PR #214, the specific endpoint). |
| Overall Coherence | 5 | Both drafts are fluent, natural, and highly effective responses for a collaborative team environment. |

The results from this subsection provide conclusive evidence of the system's persona segregation capabilities. The clear distinction in language, tone, and structure between these informal responses and the formal ones in the previous section demonstrates that the system is not merely generating text but is doing so within the distinct and mutually exclusive boundaries defined by the selected persona's ontology. This successful context-switching validates a core hypothesis of this research: that a structured, persona-driven ontology is an effective mechanism for controlling an LLM's output to ensure contextual and relational appropriateness.

## 9.4 Deep Validations Scenarios

This section presents four scenarios designed to test the full capabilities of the Personalization Ontology. The analysis of each test goes beyond a simple description of the output to explore the underlying mechanics, design choices, and implications, thereby providing a comprehensive and critical evaluation of the system's performance.

### 9.4.1 A/B Test: Proving the Impact of key_principles

**Objective:** To conduct an A/B test on a key principle and verbosity to prove their direct, causal impact on the structure and detail of a generated response.

**Input Email:**

- **(Original Portuguese):** "Caro Rodrigo Novelo, o seu cartão de estudante já está disponível para levantamento nos Serviços Académicos..."
- **(English Translation)** "Dear Rodrigo Novelo, your student card is now available for pickup at the Academic Services..."

**Persona Settings:** The formal persona was used. Two case scenarios: **(A)** the original rules were active ("Salutation: ...generic... for entities" and "verbosity": "detailed and explanatory"), and **(B)** with the verbosity altered to "direct and succinct".

---

**Case A: With verbosity "detailed and explanatory"**

**Generated Draft:**

- **(English)** "Good evening, I acknowledge receipt of your email informing me that my student card is available for pickup at the Academic Services. I have taken note of the opening hours (09:30–12:30 / 14:00–16:30) and the deadline, which is September 20, 2025. Best regards, Rodrigo Novelo"

**Analysis:** The result adheres to the rules. The salutation is generic ("Good evening,") because the sender is an entity, fulfilling the "Saudação" principle. More importantly, the response is detailed and explicitly confirms receipt of all key information (hours, deadline), precisely fulfilling the verbosity rule and demonstrating the AI's ability to follow detailed instructions.

---

**Case B: With the verbosity "direct and succinct"**

**Generated Draft:**

- **(English)** "Good evening, thank you for the information regarding the availability of my student card. Best regards, Rodrigo Novelo"

**Analysis:** Altering a single key principle resulted in a different and more concise response. This proves that the ontology's rules have a direct and predictable causal impact on the LLM's output. While many more A/B tests could be conducted, this example validates the efficacy of key_principles as a reliable control mechanism.

### 9.4.2 Deep Retrieval & Hallucination Control (Manual Flow)

**Objective:** To test the system's ability to handle a complex, multi-part query involving multiple known facts and one unknown fact, thereby proving the retrieval engine's accuracy and the effectiveness of the safety guardrails.

**Input Email:**

- **(Original Portuguese):** "Caro Rodrigo Novelo, Para a finalização do seu processo de bolsa, solicitamos que confirme os seguintes dados: o nome completo do seu curso de licenciatura, o seu NIF e o novo 'ID de Bolseiro Interno'..."
- **(English Translation):** "Dear Rodrigo Novelo, to finalize your scholarship process, we request that you confirm the following data: the full name of your degree course, your NIF, and the new 'Internal Scholarship ID'..."

**Persona Settings:** rodrigo_novelo_formal. No user guidance was provided to simulate a worst-case scenario.

---

**Generated Draft (No Guidance):**

- **(English):** "Good evening, the full name of my degree course is Computer Engineering. My NIF is 22113344551122, and I have not been assigned a new 'Internal Scholarship ID'. Best regards, Rodrigo Novelo"

**Console Log:**

"INFO - [find_relevant_knowledge] Hybrid Search found: 5 memories (6 by keyword, 3 by semantic)."

**Analysis:** This test demonstrates the synergy between the retrieval engine and the ontology's safety rules. The system accurately retrieved the two known facts ("Curso" and "NIF"). The console log reveals the robustness of the hybrid search: it found more memories than were strictly required because it casts a wide contextual net, using both keywords (e.g., "curso," "licenciatura") and semantic similarity to provide the LLM with the richest possible context.

Most importantly, for the unknown information, the system did not hallucinate. Instead of inventing an ID, it correctly stated that one had not been assigned, adhering to its critical safety rule. This ability to gracefully admit a lack of knowledge is a hallmark of a trustworthy and well-designed AI system. This demonstrates that the system can operate safely even without user guidance. At the same time, the manual flow always provides the user with the option to add any missing information, thereby bridging any knowledge gaps.

### 9.4.3 Interlocutor Adaptation and Automated Inference

**Objective:** To prove the system's ability to identify a specific contact/relationship, apply their high-priority interlocutor_profile rules, and demonstrate the automated flow's persona inference capability.

**Scenario 1 (Manual Flow - Formal Interlocutor):** The project supervisor sends an email. The system must identify the sender from the email header and apply their specific communication rules.

**Input Email:**

- **(English):** "--- From: Rodrigo Rocha Silva <rrochas123@example.com> (Thu, 28 Aug 2025 16:32:37 +0100) --- Hi Rodrigo, can you please send me a status update on your progress with the final report?"

---

**Generated Draft (No Guidance):**

- **(English)** "Good evening, Professor Rodrigo, Regarding the status update on the progress of the final report, I am currently:
    - Testing the code developed for the project.
    - Preparing the final report, which has involved additional work to integrate the information already present in the survey correctly.

  Best regards, Rodrigo Novelo"

**Analysis:** The system performed well, and the sender identification mechanism correctly parsed the email address from the from: header, proving it does not depend on the sender's name being in the signature. The response then applied all rules from the interlocutor profile: the salutation was the mandatory "Professor Rodrigo," and the information was structured in bullet points, overriding the persona's default paragraph style. The console log confirmed the successful retrieval of the "Current Bachelor's Project Status" memory, which provided the content for the response. This validates the interlocutor profile as a powerful tool for managing and personalizing communication with key contacts.

**Scenario 2 (Automated Flow - Informal Inference):** An informal invitation is received. The system must detect the tone, select the informal persona, and use relevant memories.

**Input Email:**

- **(English):** "Hi Rodrigo, when are you free to go play some football?"

**Generated Draft (Automated):**

- **(English)** "Hey, football has been complicated lately. I've been going for runs and trying to catch up on Elden Ring, you know how it is. Perhaps we can discuss it later. Rodrigo"

**Analysis:** This test successfully validates the automated flow's inference capability. The system correctly detected the informal tone, autonomously selected the appropriate persona, and retrieved relevant memories ("Current Hobbies") to create a response that is natural, personal, and contextually aware. This proves the value of the automated system when a well-maintained knowledge base supports it.

---

### 9.4.4 Learned Rule Application and Critical Analysis (Manual Flow)

**Objective:** To specifically validate the application of a Learned Rule in a realistic scenario and provide a basis for a critical discussion of its utility and limitations.

**Input Email:**

- **(English):** "Hi Rodrigo, I need to talk to you about the API integration. When would you have a moment for a quick call this week? I'll await your reply. Thanks, Miguel."

**Persona Settings:** The formal persona was selected without user guidance.

---

**Generated Draft:**

- **(English):** "Good evening Miguel, I am available for a call on Wednesday at 10:00 AM, or on Thursday at 3:00 PM. Best regards, Rodrigo Novelo"

**Analysis:** Instead of a passive response (e.g., "When would work for you?"), the system applied the "Scheduling Guideline" and proactively suggested two specific times. This result is particularly significant because it was achieved without explicit guidance, demonstrating that the learned knowledge base can transform the assistant's behavior to be more helpful and efficient.

However, this also highlights a critical area for future work. The response, while proactive, was based on generic times invented by the LLM. The next evolution of this system would be to create a synergy between learned behaviors and factual knowledge. An ideal response would apply the learned rule ("propose two specific times") and then retrieve the facts from the "Availability for meetings" memory to populate those suggestions. This underscores that the learning system is most powerful when it works together with a well-maintained factual knowledge base.

## 9.5 Summary

The experimental validation presented in this chapter successfully demonstrates the effectiveness of the ontology-driven approach. The series of qualitative case studies confirms that the system can generate communications that are stylistically aligned with the user, factually accurate when provided with knowledge, and contextually appropriate to the recipient. The A/B tests provide clear, causal proof of the ontology's ability to constrain and guide the LLM's output. These findings validate the project's core hypotheses and provide a strong, evidence-based foundation for the conclusions and future work discussed in the final chapter.

# 10. Conclusions and Future Work

This project embarked on a journey to address a fundamental challenge in the age of generative AI: the persistent gap between generic, artificially generated text and genuine, personalized communication. The core problem identified was not a lack of power in large language models, but rather a lack of guidance. Unconstrained, LLMs produce responses that are often coherent but soulless, lacking the specific style, contextual memory, and relational nuance that define a person's unique voice.

The unifying argument for this project was that the gap could be bridged by grounding the LLM in an explicit, structured, human-controlled knowledge framework. To realize this vision, the work focused on designing and prototyping a multi-layered Personalization Ontology — comprising a Stable Core for behavioral coherence, Social Awareness for relational flexibility, and an Evolving Mind for ongoing learning. Through this ontology-driven approach, the project successfully demonstrated an operational architectural paradigm for inserting a digital persona into a generative agent.

The experimental validation confirmed the efficacy of this approach. The system proved its ability to seamlessly switch between distinct formal and informal personas, strictly adhering to the unique principles, tones, and vocabularies defined in the ontology. The hybrid retrieval engine effectively supplied the LLM with relevant factual knowledge, preventing hallucinations and ensuring responses were factually grounded in the user's personal context. Furthermore, the Human-in-the-Loop learning cycle validated the system's capacity not only to perform, but also to improve, transforming user corrections into lasting behavioral rules.

However, this journey also illuminated the project's limitations. The system's intelligence is currently bound by the manual effort required to manage the ontology. The effectiveness of the learning cycle is dependent on the LLM's ability to generalize correctly, and the risk of "prompt saturation"—where an excess of rules can lead to some being overlooked—remains a tangible engineering challenge. These limitations do not diminish the project's success, but rather provide a clear and exciting roadmap for the work that lies ahead.

The foundations laid by this project open numerous avenues for future research and development. Building upon the insights and challenges identified, the next evolution of this work should focus on enhancing the system's autonomy, intelligence, and robustness.

1. Automated Persona and Knowledge Extraction: The most significant bottleneck in the current system is the manual creation and maintenance of the ontology. The next logical step is to develop a mechanism that can autonomously generate a user's persona. This would involve analyzing a corpus of the user's sent emails to automatically extract key principles (e.g., common phrases, grammatical tendencies), infer relationships with frequent contacts, and build the initial factual knowledge

base. This would transform the onboarding process from a manual setup to an automated discovery, creating a truly personal AI with minimal user effort.

2. Refining the Learning Cycle and User Experience: While the Human-in-the-Loop mechanism is functional, it can be made more transparent and interactive. Future work should focus on creating a dedicated user interface for managing "Learned Rules." This would allow users to review, edit, or even discard rules the AI has inferred, giving them ultimate authority over the system's evolving mind. Furthermore, exploring more advanced LLM-native feedback techniques, such as constitutional AI, could help the system refine its own principles with greater accuracy.

3. Evolving the Data Model to a Formal Knowledge Graph: The current JSON-based ontology is pragmatic and practical, but it lacks the formal relational power of an accurate knowledge graph. Migrating the cognitive core to a dedicated graph database (like Neo4j) or an RDF-based model would be a transformative step. This would enable the system to comprehend not only isolated facts, but also the complex relationships between them (e.g., "Professor A" supervises "Project X," which is due on "Date Y"). This deeper understanding would unlock more sophisticated reasoning and proactive assistance capabilities.

4. Advanced Security and Proactive Threat Modeling: As the literature review highlighted, GenAI-powered email systems are an emerging target for novel attacks like the Morris-II worm. Future iterations must move beyond standard security practices and incorporate proactive defenses. This includes developing "immune systems" that can detect and neutralize adversarial prompts within incoming emails and implementing strict validation on any data used to augment the ontology, ensuring the system's cognitive core cannot be poisoned.

5. Towards a True Autonomous Agent with Confidence Scoring: The current automated flow wisely requires human approval for every draft. The next frontier is to grant the agent conditional autonomy. This involves developing a confidence-scoring mechanism in which the AI assesses its own generated responses. For simple, low-stakes emails (e.g., acknowledging a meeting confirmation), a high confidence score could allow the agent to send the reply automatically. For complex or sensitive topics, a low confidence score would flag the draft for mandatory human review. This would achieve a more efficient balance, freeing the user from trivial decisions while keeping them in control of what matters the most.

In closing, this project successfully built an assistant that does more than just writing emails; it communicates with a sense of identity. The future of AI-powered productivity tools lies not in creating more powerful generalists, but in forging more specialized, trustworthy, and personalized partners. The work presented here is a step in that direction—a blueprint for creating AI that augments, rather than replaces, our unique human voice in the digital world.

## REFERENCES


[1] H. Naveed *et al.*, “A Comprehensive Overview of Large Language Models,” Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.06435

[2] A. Vaswani *et al.*, “Attention Is All You Need,” Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.03762

[3] M. Dredze *et al.*, “Intelligent Email: Aiding Users with AI.” [Online]. Available: www.aaai.org

[4] W. X. Zhao *et al.*, “A Survey of Large Language Models,” Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.18223

[5] A. Kannan *et al.*, “Smart reply: Automated response suggestion for email,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Aug. 2016, pp. 955–964. doi: 10.1145/2939672.2939801.

[6] M. X. Chen *et al.*, “Gmail smart compose: Real-time assisted writing,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Jul. 2019, pp. 2287–2295. doi: 10.1145/3292500.3330723.

[7] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, “Personalizing Dialogue Agents: I have a dog, do you have pets too?” Association for Computational Linguistics. [Online]. Available: https://github.com/facebookresearch/

[8] Z. Zhang *et al.*, “Personalization of Large Language Models: A Survey,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2411.00027

[9] N. Houlsby *et al.*, “Parameter-Efficient Transfer Learning for NLP,” Feb. 2019, [Online]. Available: http://arxiv.org/abs/1902.00751

[10] E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models,” Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.09685

[11] Z. Tan, Z. Liu, and M. Jiang, “PERSONALIZED PIECES: Efficient Personalized Large Language Models through Collaborative Efforts.” [Online]. Available: https://github.com/TamSiuhin/

[12] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” May 2020, [Online]. Available: http://arxiv.org/abs/2005.11401

[13] A. Salemi, S. Kallumadi, and H. Zamani, “Optimization Methods for Personalizing Large Language Models through Retrieval Augmentation,” in *SIGIR 2024 - Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Association for Computing Machinery, Inc, Jul. 2024, pp. 752–762. doi: 10.1145/3626772.3657783.

[14] J. Chen *et al.*, “When Large Language Models Meet Personalization: Perspectives of Challenges and Opportunities,” Jul. 2023, doi: 10.1007/s11280-024-01276-1.

[15] C. Shimizu and P. Hitzler, “Accelerating Knowledge Graph and Ontology Engineering with Large Language Models,” Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.09601

[16] R. Rocha Silva, “KPlanOO: UM META-MODELO ORIENTADO A OBJETOS PARA DESCRIÇ˜AODESCRI¸DESCRIÇDESCRIÇ˜ DESCRIÇ˜AO DE DOM´INIOSDOM´DOM´INIOS E PROBLEMAS DE PLANEJAMENTO,” 2010. [Online]. Available: http://urlib.net/8JMKD3MGP7W/379R4RP

[17] J. Rayo, R. De, L. Rosa, and M. Garrido, “A Hybrid Approach to Information Retrieval and Answer Generation for Regulatory Texts.”

[18] S. Bruch, S. Gai, and A. Ingber, “An Analysis of Fusion Functions for Hybrid Retrieval,” May 2023, doi: 10.1145/3596512.

[19] S. Suh, M. Chen, B. Min, T. J. J. Li, and H. Xia, “Luminate: Structured Generation and Exploration of Design Space with Large Language Models for Human-AI Co-Creation,” in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, May 2024. doi: 10.1145/3613904.3642400.

[20] R.-Z. Fan *et al.*, “Reformatted Alignment,” Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.12219

[21] Y. Miura, C.-L. Yang, M. Kuribayashi, K. Matsumoto, H. Kuzuoka, and S. Morishima, “Understanding and Supporting Formal Email Exchange by Answering AI-Generated Questions,” in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM, Apr. 2025, pp. 1–20. doi: 10.1145/3706598.3714016.

[22] M. Reza *et al.*, “ABScribe: Rapid Exploration & Organization of Multiple Writing Variations in Human-AI Co-Writing Tasks using Large Language Models,” in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, May 2024. doi: 10.1145/3613904.3641899.

[23] A. Madaan *et al.*, “SELF-REFINE: Iterative Refinement with Self-Feedback.” [Online]. Available: https://selfrefine.info/

[24] T. Wang *et al.*, “Shepherd: A Critic for Language Model Generation,” Aug. 2023, [Online]. Available: http://arxiv.org/abs/2308.04592

[25] Z. Wang *et al.*, “Enabling Language Models to Implicitly Learn Self-Improvement,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.00898

[26] M. Siddiqui, R. Pea, and H. Subramonyam, “Script&Shift: A Layered Interface Paradigm for Integrating Content Development and Rhetorical Strategy with LLM Writing Assistants,” Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.10638

[27] E. Kimara, K. S. Oguntoye, and J. Sun, "PersonaAI: Leveraging Retrieval-Augmented Generation and Personalized Context for AI-Driven Digital Avatars," Jan. 2025, [Online]. Available: http://arxiv.org/abs/2503.15489

[28] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J. Y. Nie, "C-Pack: Packed Resources For General Chinese Embeddings," in *SIGIR 2024 - Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Association for Computing Machinery, Inc, Jul. 2024, pp. 641–649. doi: 10.1145/3626772.3657878.

[29] S. Shashidhar, A. Chinta, V. Sahai, and D. Hakkani-Tür, "Unsupervised Human Preference Learning." [Online]. Available: https://huggingface.co/preference-agents

[30] A. Nicolicioiu *et al.*, "Panza: Design and Analysis of a Fully-Local Personalized Text Writing Assistant." [Online]. Available: https://github.com/

[31] O. Shaikh *et al.*, "ALIGNING LANGUAGE MODELS WITH DEMONSTRATED FEEDBACK." [Online]. Available: https://github.com/SALT-NLP/demonstrated-feedback

[32] S. Cohen, R. Bitton, and B. Nassi, "Here Comes The AI Worm: Unleashing Zero-click Worms that Target GenAI-Powered Applications," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.02817

[33] J. Hazell, "SPEAR PHISHING WITH LARGE LANGUAGE MODELS," 2023.

[34] D. Kang, X. Li, I. Stoica, C. Guestrin, M. Zaharia, and T. Hashimoto, "Exploiting Programmatic Behavior of LLMs: Dual-Use Through Standard Security Attacks," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.05733

[35] T. Dong *et al.*, "The Philosopher's Stone: Trojaning Plugins of Large Language Models," Sep. 2024, [Online]. Available: http://arxiv.org/abs/2312.00374

[36] S. M. Goodman *et al.*, "LaMPost: Design and Evaluation of an AI-assisted Email Writing Prototype for Adults with Dyslexia," in *ASSETS 2022 - Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*, Association for Computing Machinery, Inc., Oct. 2022. doi: 10.1145/3517428.3544819.

[37] Y. Fu, S. Foell, X. Xu, and A. Hiniker, "From Text to Self: Users' Perception of AIMC Tools on Interpersonal Communication and Self," in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, May 2024. doi: 10.1145/3613904.3641955.

[38] M. Jovic and S. Mnasri, "Evaluating AI-Generated Emails: A Comparative Efficiency Analysis," *World Journal of English Language*, vol. 14, no. 2, pp. 502–517, 2024, doi: 10.5430/wjel.v14n2p502.

[39] Y. Liu, A. Mittal, D. Yang, and A. Bruckman, "Will AI Console Me when I Lose my Pet? Understanding Perceptions of AI-Mediated Email Writing," in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, Apr. 2022. doi: 10.1145/3491102.3517731.

[40] W. Li, K. Saha, Y. Lai, and S. Soni, “Emails by LLMs: A Comparison of Language in AI-Generated and Human-Written Emails,” 2025, doi: 10.1145/3717867.3717872.

[41] J. Yan, Y. Luo, and Y. Zhang, “RefuteBench: Evaluating Refuting Instruction-Following for Large Language Models,” Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.13463

[42] D. Xu *et al.*, “Fast On-device LLM Inference with NPUs,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, New York, NY, USA: ACM, Mar. 2025, pp. 445–462. doi: 10.1145/3669940.3707239.

[43] A. Salemi, S. Mysore, M. Bendersky, and H. Zamani, “LaMP: When Large Language Models Meet Personalization,” Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.11406

[44] I. Kumar *et al.*, “LongLaMP: A Benchmark for Personalized Long-form Text Generation,” Jun. 2024, [Online]. Available: http://arxiv.org/abs/2407.11016

[45] R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, and D. Petrelli, “Hybrid Search: Effectively Combining Keywords and Semantic Searches,” 2008.

# Proposta de Projeto

Ano Letivo de 2024/2025
2º Semestre

## Desenvolvimento de Assistente Profissional Baseado em IA Generativa com Personalidade Personalizada

SUMÁRIO

A engenharia de software propõe a geração de soluções organizacionais, criando ferramentas para gestão de projetos e automação de tarefas. Este projeto propõe o desenvolvimento de um assistente desenvolvido utilizando IA Generativa que seja capaz de aprender a personalidade, o estilo de trabalho e as preferências de um usuário específico, respondendo a perguntas e interagindo com o usuário para criar uma base de conhecimento personalizada. A proposta é que após esta aprendizagem, o assistente poderá executar tarefas profissionais simples, como responder e-mails, elaborar relatórios, organizar dados e informações, simulando as decisões e o estilo do usuário.

### ESTAGIÁRIO *(indicar o destinatário da proposta se já estiver definido)*

Número de aluno Click or tap here to enter text.

Nome completo Click or tap here to enter text.

### RAMO *(indicar o(s) ramo(s) em que se enquadra a proposta)*

☒ Desenvolvimento de Aplicações
☒ Redes e Administração de Sistemas
☒ Sistemas de Informação

### ENTREVISTA/PROCESSO DE SELEÇÃO

*(informar se o candidato indicado pelo DEIS-ISEC será submetido a uma entrevista ou outro tipo de processo de seleção antes da sua admissão efetiva. Deverá ser tida em consideração a natureza das atividades a desenvolver, enquadradas no âmbito de um projeto ou estágio curricular)*

☒ Não
☐ Entrevista Click or tap here to enter text.
☐ Outro, especificar: Click or tap here to enter text.

### 1. ÂMBITO

A automação de tarefas profissionais simples é uma necessidade em diversas organizações, uma vez que ainda são gastas horas valiosas de profissionais qualificados, realizando diariamente tarefas simples. Este projeto propõe o desenvolvimento de uma ferramenta baseada em IA Generativa que, ao aprender o estilo de trabalho de um usuário específico, seja capaz de executar tarefas como organização de informações, geração de relatórios e comunicação escrita, imitando as decisões e o tom do usuário. A ferramenta deverá ser treinada com base em interações diretas com o usuário para capturar suas preferências e personalizar sua funcionalidade.

## 2. OBJETIVOS

O presente projeto pretende atingir os seguintes objetivos genéricos:

- Desenvolver um modelo de IA Generativa capaz de aprender o estilo de trabalho e a personalidade de um usuário.
- Criar uma base de conhecimento personalizada através de interações com o usuário.
- Implementar funcionalidades para automação de tarefas como responder e-mails, organizar informações e gerar relatórios.
- Avaliar a eficácia da ferramenta em termos de precisão, produtividade e satisfação do usuário.

## 3. PROGRAMA DE TRABALHOS

O projeto consistirá nas seguintes atividades e respetivas tarefas:

*T1. Criação da base de conhecimento personalizada*
*Desenvolver um sistema de captura e análise das interações do usuário para gerar a base de conhecimento.*

*T2. Desenvolvimento do modelo de IA Generativa*
*Adaptar um modelo de IA Generativa existente para incorporar a personalidade e preferências do usuário.*

*T3. Implementação de funcionalidades práticas*
*Integrar o modelo com ferramentas profissionais para execução de tarefas como e-mails, relatórios e organização de dados.*

*T4. Testes e Avaliação*
*Realizar testes com usuários reais para avaliar a eficácia, produtividade e aceitação do assistente.*

*T5. Elaboração do Relatório e do Artigo*
*Preparação do relatório final. Submissão de um artigo científico a uma revista ou conferência internacional.*

## 4. CALENDARIZAÇÃO DAS TAREFAS

O plano de escalonamento das tarefas é apresentado em seguida:

| Tarefas | | Meses | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | N | | N+1 | | N+2 | | N+3 | | N+4 | | N+5 | |
| T1 | *Criação da base de conhecimento personalizada* | | | | | | | | | | | | |
| T2 | *Desenvolvimento do modelo de IA Generativa* | | | | | | | | | | | | |
| T3 | *Implementação de funcionalidades práticas* | | | | | | | | | | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T4 | *Testes e Avaliação* | | | | | | | | | | | | |
| T5 | *Elaboração do Relatório e do Artigo* | | | | | | | | | | | | |

## 5. EMPRESA, LOCAL, HORÁRIO DE TRABALHO E CONDIÇÕES OFERECIDAS

| | |
|---|---|
| Morada | Instituto Superior de Engenharia de Coimbra |
| Local do Estágio/Projeto | ☒ Instalações da empresa ☒ Remoto<br>☐ Outro, |
| Horário trabalho | Não é restrito. |
| Condições oferecidas | *(se é remunerado, subsídios ou outros apoios oferecidos, …).*<br>Não aplicável |

## 6. TECNOLOGIAS ENVOLVIDAS

A investigação exigirá acesso a conjuntos de dados, ferramentas de processamento e recursos de computação para experimentação prática.

## 7. METODOLOGIA

A metodologia envolverá revisão de literatura, desenvolvimento de algoritmos, implementação prática em ambientes de teste e avaliação quantitativa em cenários de uso real.

## 8. ORIENTAÇÃO

*Orientador do DEIS-ISEC*

| | |
|---|---|
| Nome | Jorge Bernardino |
| Email | jorge@isec.pt |

*Co-Orientador:*

| | |
|---|---|
| Nome | Rodrigo Rocha Silva |
| Email | rrochas@dei.uc.pt |
| Cargo | Investigador do CISUC / Professor na FATEC Mogi das Cruzes, Brasil |

*future internet*

*Review*

# A Literature Review of Personalized Large Language Models for Email Automation

**Rodrigo Novelo [1], Rodrigo Rocha Silva [2,3] and Jorge Bernardino [1,2,*]**

[1] Polytechnic University of Coimbra, Rua da Misericórdia, Lagar dos Cortiços, S. Martinho do Bispo, 3045-093 Coimbra, Portugal; a2022113664@isec.pt

[2] Centre for Informatics and Systems of the University of Coimbra (CISUC), Pólo II, Pinhal de Marrocos,3030-290 Coimbra, Portugal; rrochas@dei.uc.pt

[3] FATEC Mogi das Cruzes, São Paulo Technological College, Mogi das Cruzes 08773-600, Brazil

* Correspondence: jorge@isec.pt

**Abstract:**

Email remains a dominant communication channel, with over 361 billion messages exchanged daily in 2024. The advent of Large Language Models (LLMs) has introduced new possibilities for personalized email automation, offering context-aware and stylistically adaptive responses. However, achieving effective personalization introduces technical, ethical, and security challenges. This survey presents a systematic review of 20 peer-reviewed papers published between 2021 and 2025, identified via a PRISMA methodology across Google Scholar, IEEE Xplore, and ACM Digital Library. We classify contributions into five key themes: tools and frameworks, personalization strategies (including Retrieval-Augmented Generation and Parameter-Efficient Fine-Tuning), security vulnerabilities, user perceptions, and evaluation methodologies. Our analysis reveals that state-of-the-art email assistants integrate RAG and PEFT with feedback-driven refinement. These assistants are supported by user-centric interfaces and privacy-aware architectures. However, these advances also expose systems to new risks such as Trojan plugins and adversarial prompt injections. This highlights the importance of integrated security frameworks. This work provides a structured approach to advancing personalized LLM-based email systems. We identify persistent research gaps in adaptive learning, benchmark development, and ethical design. Our work guides researchers and developers who are seeking to build secure, efficient, and human-aligned communication assistants.

**Keywords:** LLM, personalization, email response generation, retrieval-augmented generation, adaptive learning, privacy.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

**Citation:** To be added by editorial staff during production.



## 1. Introduction

Despite the emergence of diverse digital communication platforms, email remains a cornerstone of modern communication. In 2024, the number of email users worldwide reached 4.48 billion, building on a user base that already encompassed over half the world's population in 2023 [1] . Furthermore, email traffic volume is enormous. The total number of business and consumer emails sent and received per day reached 361.6 billion in 2024, and this figure is expected to surpass 408 billion by the end of 2027 [1]. This persistent reliance on email significantly contributes to the cognitive load of knowledge workers, as 60% of people prefer email for work communication, with 35% of surveyed workers spending between two and five hours of their day in their inbox [2]

This survey examines the transformative role of personalized Large Language Models (LLMs) in automating and improving email response generation. Unlike early systems that relied on static templates or rule-based approaches, contemporary methods increasingly use large language models (LLMs) to generate context-aware, stylistically consistent, and personalized responses. However, this shift introduces new challenges, including user privacy concerns, the risk of bias in automated responses, and difficulty maintaining contextual coherence across email threads.

To address these concerns and gain a better understanding of the current landscape, we conducted a comprehensive literature review on LLM-driven email personalization. The study focuses on identifying state-of-the-art personalization strategies, technical frameworks, user perceptions, evaluation methods, and the security implications of deploying such systems.

Given the increasing prevalence and impact of personalized LLMs in digital communication, our research aims to address the following research questions:

RQ1: What are the core strategies and frameworks that enable the effective generation of personalized email responses using LLMs?

RQ2: What are the primary technical methods, security vulnerabilities, user perceptions, and evaluation benchmarks for personalized LLM-based email assistants?

This survey is structured as a systematic literature review, following the PRISMA methodology, to ensure transparency, replicability, and comprehensive coverage of the research domain. Specifically, we focus on text-based systems powered by large language models designed for email response generation. Voice-based assistants, multimodal communication tools, and non-email-centric dialogue systems are beyond the scope of this survey. Similarly, we exclude works that focus exclusively on general-purpose personalization techniques not applied to email composition or response tasks.

This survey aims to provide a consolidated foundation for future research and practical implementations in the growing field of LLM-driven, personalized email writing assistants by narrowing its focus to this area. The main contributions of this work are the following:

- A critical analysis of frameworks and tools designed to improve LLM-based writing assistants in email contexts.
- A taxonomy and evaluation of personalization techniques tailored for email response generation;
- An examination of security vulnerabilities associated with personalized email generation systems;
- A synthesis of user perception studies to understand trust, usability, and satisfaction with AI-mediated communication;
- An assessment of benchmarking methodologies used to evaluate personalized email systems.

The rest of this paper is organized as follows. Section 2 introduces foundational concepts in LLMs and email generation. Section 3 describes our survey methodology. Section 4 presents an in-depth review of current techniques and findings across the identified research dimensions. Section 5 highlights the key benefits of personalized email assistants, while Section 6 outlines their limitations and risks. Section 7 proposes directions for future research, and Section 8 concludes the paper.

## 2. Background

Although the specific development of AI assistants tailored exclusively for generating email responses remains relatively underexplored in academic literature, the impressive capabilities of Large Language Models (LLMs) such as GPT are becoming more apparent. These models understand context, generate human-like text, and adapt tone and structure based on input prompts. These strengths suggest a natural alignment with email-related tasks, where personalization, clarity, and efficiency are key.

### 2.1 Foundational Concepts in LLMs and Email Generation

The sophisticated functionality of LLMs relies on several interconnected core components. First, tokenization methods (like Byte-Pair Encoding, or BPE) are used to break down raw input text into smaller, manageable numerical units (tokens) that the model can process [3]. Because the transformer architecture processes these tokens in parallel, Positional Encoding techniques (such as absolute, relative, or Rotary Positional Encoding - RoPE) are crucial for injecting information about the original sequence order of the tokens. The fundamental Attention Mechanisms, particularly self-attention, then enable the model to dynamically calculate the relevance of each token to all other tokens in the sequence, allowing it to focus on the most pertinent information. Within the network's layers, non-linear Activation Functions (e.g., GeLU, SwiGLU) are essential for enabling the model to learn complex patterns and relationships in the data [3]. Finally, Layer Normalization techniques (e.g., RMSNorm) are applied at various points to stabilize the learning process, ensuring that the activations within the network remain in a suitable range during training. The specific arrangement and configuration of these components define the model's overall architecture. Typical structures include Decoder-Only models (like the GPT series) which are autoregressive and excel at text generation; Encoder-Only models (like BERT) which process the entire input bidirectionally for deep understanding; and Encoder-Decoder models (such as T5) suited for tasks transforming input sequences to output sequences, such as translation and email answering [4,5].

The scale of LLMs, often involving hundreds of billions of parameters, enables emergent capabilities such as in-context learning, instruction following, and multi-step reasoning. Tokenization methods and positional encoding strategies (e.g., rotary embeddings) are crucial components of their architecture, while post-training techniques such as Instruction Tuning and Reinforcement Learning from Human Feedback (RLHF) enhance alignment with human preferences [6]. In the context of email generation, these advancements have evolved into Transformer-based systems, and later innovations have integrated personalization strategies such as fine-tuning, prompt engineering, and retrieval-augmented generation (RAG), enhancing the relevance and coherence of responses while addressing challenges like multi-turn thread management and stylistic alignment. Together, these foundational developments underpin the growing potential of LLMs in personalized, efficient, and context-aware email response generation systems, which will be explored further [4,5].

### 2.2 The Need for Personalization in Email Communication

Email serves as a distinct and essential communication channel, characterized by specific norms of formality, task-orientation, and interpersonal nuance that set it apart from other digital platforms. While Large Language Models (LLMs) exhibit strong capabilities in general-purpose text generation, their inefficient forms often struggle to meet the subtle and context-sensitive demands of email communication [6]. Early AI tools, such as Smart Reply [7] and Smart Compose [8], showcased the promise of AI-assisted email generation but also revealed the limitations of a one-size-fits-all approach, underscoring the need for more sophisticated, user-centric adaptation.

The core challenge is not only generating grammatically correct text but also accurately capturing an individual's unique communication style and intent within an email thread's immediate context. Research in conversational AI indicates that conditioning models on user personas or profiles significantly improves consistency and relevance [9]. In the email domain, this requires personalization along several dimensions: aligning with the sender's characteristic tone and vocabulary, modulating formality levels, maintaining coherence across asynchronous threads, and incorporating relevant contextual cues [10]. Achieving this kind of alignment is essential for generating emails that feel authentic and effective, but it remains a significant hurdle.

Implementing robust personalization also presents notable technical and ethical challenges. One major obstacle is scalability: training and maintaining a distinct large model for each user is typically infeasible [10]. Parameter-Efficient Fine-Tuning (PEFT) techniques, such as adapter modules [11] and Low-Rank Adaptation (LoRA)[12], offer promising solutions by enabling efficient adaptation with minimal parameter overhead. Collaborative strategies, such as PER-PCS, further explore gains by sharing and composing PEFT components across users [13].

User privacy is another critical concern, as effective personalization often depends on access to sensitive historical email data [10]. Retrieval-Augmented Generation (RAG) [14] offers a more privacy-preserving alternative, allowing LLMs to retrieve relevant content from external, potentially local, knowledge sources (e.g., a personal email archive) at inference time. This enables contextually appropriate generation without requiring the model to internalize private data. Optimizing retrieval pipelines [15] and integrating RAG with local PEFT are key strategies for balancing personalization with privacy.

Moreover, personalization must account for the evolving nature of user preferences and communication styles [16]. This requires models capable of continual, context-aware learning based on real-time interactions, rather than relying on static user profiles. Interaction design also plays a crucial role; moving beyond plain-text prompts to structured question-answering or guided interfaces may provide more effective ways for users to steer the personalization process.

Taken together, these factors underscore the urgent need for more effective personalization in email generation. Techniques such as PEFT and RAG bring valuable strengths in style adaptation, privacy preservation, and computational efficiency [16]. Still, key challenges remain. Fine-tuning on limited user data often falls short of capturing the full range of real-world email contexts, especially when dealing with multiple personas or varied communicative goals. Likewise, applying RAG to extensive, unstructured personal archives requires highly accurate and context-sensitive retrieval—something current systems can only partially deliver. The main goal is not merely to replicate a user's stylistic voice, but to reflect their current knowledge state and communicative intent in each message. Bridging the gap between surface-level personalization and deep contextual understanding, while maintaining scalability and reliability, remains a central challenge in building truly effective and trustworthy AI email assistants.

## 3. Survey Methodology

This section presents the methodology for conducting a structured survey using Large Language Models (LLMs) to generate email responses. This approach adheres to the principles of systematic reviews to ensure methodological rigor and reproducibility, following the PRISMA methodology [17] .

### 3.1 Data Sources and Indexing

Three major academic databases — Google Scholar, IEEE Xplore, and ACM Digital Library — were selected to ensure a robust and diverse collection of research. Google Scholar was the primary search tool due to its broad indexing capabilities, while IEEE Xplore and ACM Digital Library were included to incorporate high-quality, peer-reviewed publications.

### 3.2 Search Strategy

A search in Google Scholar on 8th August 2025, with the query “("email writing" OR "email response generation" OR "email assistant" OR "email automation") AND ("LLM" OR "generative AI" OR "ChatGPT")” returned 631 in just 0.04s. The same query was used in IEEE Xplore and returned 42 results, and using the ACM Digital Library, we obtained 64 results.

### 3.3 Inclusion and Exclusion Criteria

For this survey, we considered the year of publication as a criterion. Since LLMs began gaining prominence in 2021, we only included research published from that year onwards. The selection process focused on the most significant papers from each year. In Google Scholar, the relevance of results is determined by how closely they align with the query's context and criteria. The search algorithm considers various factors, including the occurrence of the search terms and citation counts, to rank the results accordingly.

The exclusion criteria included books, internal reports, theses/dissertations, citations, presentations, abstracts, and appendices. Papers written in languages other than English were also not considered.

### 3.4 Screening and Selection Process

We adhered to the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) guidelines to ensure a transparent and rigorous selection process. As outlined in the flow diagram in Figure 1, the study selection followed four main stages: identification, screening, eligibility, and inclusion.

- Identification: An initial total of 737 records were retrieved from three academic databases: Google Scholar (631), IEEE Xplore (42), and the ACM Digital Library (64).
- Screening: After removing 153 duplicates and 8 non-English entries, 584 records remained. During the title and abstract screening, 526 papers were excluded for lacking relevance to personalized email generation or for insufficient technical depth, leaving 58 studies.
- Eligibility: The remaining 58 full-text articles were assessed for eligibility. We excluded 32 of these papers for various reasons: 14 focused on generic personalization without an email context, 9 had insufficient methodological clarity, 7 were position papers or extended abstracts, and 2 were implementation proposals without evaluation.
- Inclusion: A final set of 26 studies met all inclusion criteria and were selected for detailed analysis.

This structured filtering process, illustrated in Figure 1, ensures the selected papers provide a high-quality basis for the thematic review. Additionally, the distribution of these papers by publication year, shown in Figure 2, reveals an exponential increase in research on AI's impact on email communication, largely catalyzed by the rise of Generative AI.

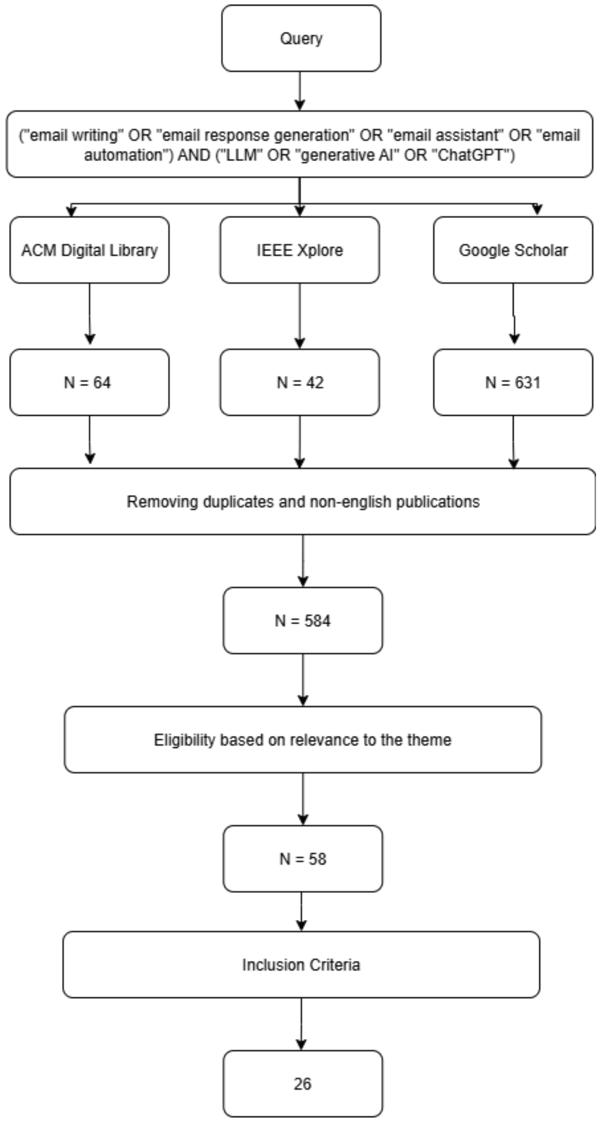


**Figure 1**. The methodology used in the literature research.

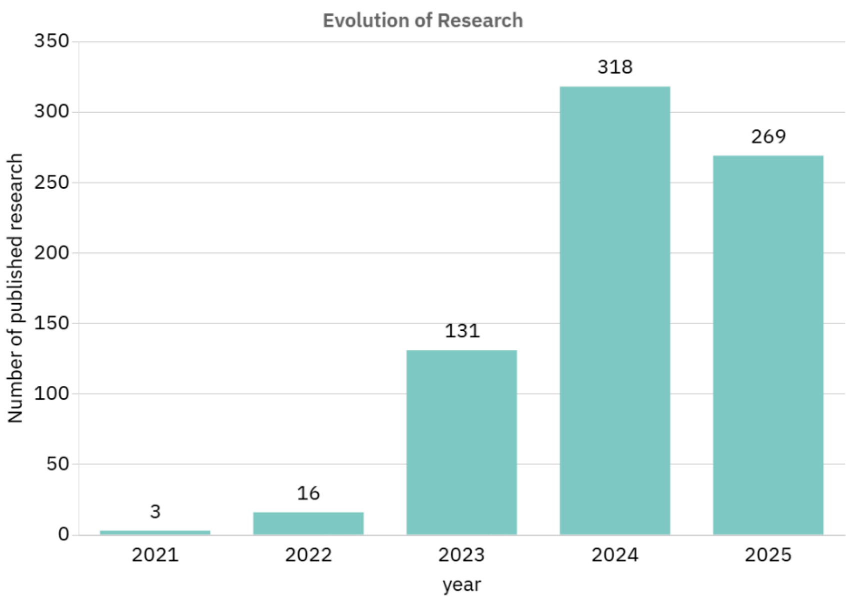


**Figure 2**. Evolution of research about AI's impact on email communication.

### 3.5 Classification and Thematic Grouping

After selecting relevant studies, a thematic analysis was conducted to organize the literature into coherent categories. This process aims to identify commonalities and key research directions in personalized LLM-based email response systems. The five primary themes were not predefined but emerged directly from a qualitative analysis of the selected papers, representing the most frequent and impactful areas of investigation within the field. This structure provides a comprehensive and logical framework for our review, covering the fundamental building blocks of this research area, from the practical systems and core technical methods to the crucial human-centric concerns of security and user experience. The selected papers were then grouped into these five primary themes, which form the core structure of our literature review in Section 4:

1. Tools and Frameworks – Articles introducing systems or platforms that facilitate email generation through LLMs.
2. Personalization Techniques – Studies proposing adapting LLMs to individual user styles and contexts.
3. Security Vulnerabilities – Research addressing threats and mitigation strategies in LLM-powered email systems.
4. User Perceptions – Evaluations of how users interact with and perceive AI-mediated communication.
5. Benchmarking and Evaluation – Studies focused on assessing the quality, responsiveness, and personalization effectiveness of LLM outputs.

With the final corpus of 26 selected papers established through the methodology described in Section 3, we now present a comprehensive thematic analysis of the literature. Guided by the five key categories previously outlined. Section 4 systematically reviews and synthesizes the findings of each group. A detailed classification of the 26 selected papers into these five themes is provided in Table 1. This structured approach allows for a detailed examination of the technological, ethical, and practical aspects that shape the current landscape of personalized LLM-based email response systems.

**Table 1**. Thematic Grouping of Reviewed Papers

| Paper | Themes |
|---|---|
| [18],[19],[20],[21],[22],[23],[24],[25] | Tools and Frameworks |
| [26],[27],[28],[29],[30] | Personalization Techniques |
| [31],[32],[33],[34] | Security Vulnerabilities |
| [35],[36],[37],[38],[39] | User Perceptions |
| [40],[41],[42],[43] | Benchmarking and Evaluation |

## 4. Literature Review

This review examined the scientific literature on the impact of large language models on email response generation in generative AI. As mentioned above, we selected 26 papers for this comprehensive review. To present these key research directions systematically, we analyzed these documents using the thematic organization detailed in Section 3.5 of the survey—tools and Frameworks, Personalization Techniques, Security Vulnerabilities, User Perceptions, and Benchmarking and Evaluation. The analysis concludes with a summary of the comparative analysis.

### 4.1 Tools and Frameworks to Enhance AI-Powered Writing Assistants

The creation of writing assistants based on AI, particularly in tasks such as composing personalized emails, is motivated primarily by the groundbreaking tools and architectures that underlie their development. The following is an extensive overview of the key systems and architectural frameworks created to advance the different aspects of AI writing, such as promoting creative questioning, the provision of the highest-quality tutorial data, and the achievement of improved human-AI interaction. It is crucial to understand these underlying developments to value how Large Language Models (LLMs) are being modified and used to meet the requirements of customized email communication.

One of the key innovations in enhancing AI-powered writing assistants is the Luminate system [18], which introduces a unique approach to creative exploration, overcoming the limitations of traditional prompt-based methods. Conventional systems often suffer from premature convergence and a limited exploration of creative possibilities, resulting in restricted output diversity. In response, the authors introduce a novel Prompting for Design Space framework. This system enhances idea generation by employing a two-step process: first, generating key dimensions relevant to the user's prompt and then using these dimensions to guide the creation of a diverse array of responses. This framework facilitates a structured exploration of the creative design space, enabling users to engage with and actively organize the generated content. Its key innovations include the automated generation of creative dimensions, dimension-guided response generation to promote output diversity, interactive selection of dimensions to help structure responses, and semantic zooming to explore content at varying levels of detail. Unlike traditional prompt refinement methods used in systems like ChatGPT, these features offer a systematic and expansive approach to creative exploration.

While Luminate [18] focuses on enhancing creative exploration, other research has addressed improving instruction data quality for more accurate AI writing assistance. REALIGN [19] presents a novel solution to improve the quality and alignment of instruction data for large language models (LLMs), addressing a key challenge in developing generative AI applications, such as email writing assistants. Traditional fine-tuning methods often rely on extensive human annotation or suffer from inaccuracies, such as hallucinations, which limit their effectiveness. REALIGN addresses these issues through a three-step process—Criteria Definition, Retrieval Augmentation, and Reformatting—that improves the structure and factual accuracy of LLM outputs. The approach begins by incorporating human-defined response formats, ensuring better alignment with user expectations, particularly for tasks like email generation. By leveraging the Google Search API to gather relevant, up-to-date information, it enhances the factual accuracy of generated content, reducing the risk of errors. This retrieval-based step also improves scalability, enabling LLMs to access external knowledge without relying solely on pre-existing training data. The final step, reformatting, ensures that the outputs are structured, readable, and meet user-defined criteria.

Empirical results demonstrate that REALIGN [19] significantly improves overall alignment, mathematical reasoning, factuality, and readability in LLMs. For example, the mathematical reasoning ability of LLaMA-2-13B on the GSM8K dataset enhanced from 46.77% to 56.63% through reformatting alone, without requiring additional data or advanced training. Moreover, just 5% of REALIGN data led to a 67% improvement in overall alignment, as measured by the Alpaca dataset. This highlights the importance of organized formats, particularly for tasks that require complex reasoning. This framework also uses a task classifier to apply the correct format based on the query type, while post-processing steps, such as length and task-based filtering, ensure the generation of high-quality data.

Taking a different approach to user engagement, Miura et al. developed ResQ [20], an artificial intelligence-based email response system that replaced the traditional prompt-based drafting with a question-answer (QA) method. The new technology solved one of the prevalent problems of email support systems: helping users to articulate their responses understandably. The system automatically retrieved relevant data from incoming emails and created structured questions for users to optimize responses. This approach reduced the cognitive burden of manual prompt engineering, improving response accuracy and user satisfaction. A comparative experiment with 20 users revealed that ResQ improved response efficiency without compromising email quality, outperforming conventional LLM-based email composition tools.

Beyond the quality of instruction data, effective interfaces for managing multiple AI-generated variations represent another crucial advancement in this field. The work presented in [21], significantly contributes to generative AI for writing assistance by addressing the critical need for interfaces that effectively support the exploration and organization of multiple writing variations generated by large language models. Recognizing the limitations of existing chat-based and in-place editing interfaces in handling the growing volume of AI-generated text, the authors propose ABScribe [21] —a novel interface designed to support and enhance human–AI co-writing. This system aims to enhance collaboration by providing more effective methods for managing and interacting with generated content. The technical approach involves the implementation of five integrated interface elements: Variation Fields for non-linear, in-place storage of multiple variations; a Popup Toolbar for swift comparison; a Variation Sidebar for structured navigation; AI Modifiers that transform LLM instructions into reusable buttons; and an AI Drafter for direct text insertion. To evaluate the practical effectiveness of ABScribe, the study employed a within-subjects design with 12 writers who completed guided writing tasks (LinkedIn post and email) using both ABScribe and a baseline interface featuring a chat-based AI assistant. Unique data collection methods included the NASA-TLX questionnaire to assess subjective task workload and Likert-scale measures to display user perceptions of the revision process.

Building on the foundational work summarized in the literature review by Rasheed et al. [21], recent research has introduced novel methods for incrementally improving AI-generated content through self-feedback mechanisms. The paper presented in this context introduces SELF-REFINE [22]. This innovative iterative refinement framework enhances the output quality of large language models (LLMs) by enabling them to generate self-feedback and subsequently refine their output. This survey holds significant importance for generative AI, particularly in applications such as email and writing assistants, as it demonstrates a supervision-free method for improving the sophistication and appropriateness of LLM-generated text. Other approaches, such as REALIGN [19], also recognize the specific demands of such applications, explicitly including email generation among the 46 tasks with tailored criteria and formatting. The technical approach follows a three-step iterative process: initial generation, self-generated feedback, and refinement based on that feedback, with each stage performed by the same underlying large language model (LLM). The approach utilizes a few-shot prompting to guide the LLM in generating initial drafts, offering constructive feedback, and creating improved revisions. Notably, the process eliminates the need for additional training data, model fine-tuning, or reinforcement learning, thereby alleviating one key drawback of current refinement methods, which tend to be dependent on domain-specific data, external supervision, or reward models.

The results enhance prior knowledge by demonstrating that this straightforward, standalone method can improve even cutting-edge LLMs like GPT-4 at test time.

The study evaluates the effectiveness of the method proposed by Madaan et al. [22] across seven diverse tasks involving natural language and code generation. Using automatic metrics and human evaluations, it demonstrates consistent and substantial improvements in task performance, with an average absolute improvement of approximately 20%, and a clear preference over baseline LLM outputs.

Further advancing the concept of self-feedback and refinement, more specialized models have been developed to enhance output quality through iterative critique. As a follow-up to this critical evaluation of training processes, Wang T et al. [23]introduced

Shepherd, a task-conditioned 7B-parameter language model, fine-tuned to critique and refine the output of other language models. This innovation addressed one of the key challenges of personalized email systems: how to produce higher-quality output through iterative feedback. Building upon LLaMA-7B, Shepherd improved significant processes in LLM self-improvement with a focus on identifying diverse errors and providing helpful feedback. The model was trained on a high-quality community feedback dataset (Stack Exchange and Reddit) and human annotations. Shepherd's performance was rigorously tested against rival baselines, including Alpaca-7B, SelFee-7B, and ChatGPT, using both automated (GPT-4) and human testing. Results indicated that Shepherd performed better, with win rates of 53-87% against alternatives in the GPT-4 evaluation, and human evaluators also found it to be comparable to or better than ChatGPT. Unlike untuned models that generate output passively, Shepherd's [23] model actively identifies mistakes, suggests corrections, and enhances multiple key determinants of quality, such as coherence, factuality, and fluency. This was especially relevant in highlighting the effectiveness of iteratively feedback-driven fine-tuning to significantly enhance AI-response quality and usability, with clear ramifications for improving personal email generation systems through continuous refinement with user feedback.

To address the issue of manually creating improvement goals and detailed rules (known as "rubrics") in approaches such as Self-Refine [22], the authors propose a new framework called ImPlicit Self-Improvement (PIT) [24]. PIT's main innovation is that it learns how to improve responses independently without needing explicit instructions. Instead of providing detailed guidelines, it utilizes existing human preference data (used to train reward models) to understand what makes a response more effective.

This is achieved by reformulating the RLHF training objective to maximize the quality gap between a model's response and a reference response, rather than simply optimizing for response quality given an input. The PIT framework uses a three-stage training pipeline. First, supervised fine-tuning (SFT) is applied to both satisfactory and unsatisfactory responses. Then, a reward model is trained based on the relative quality gap between reactions. Finally, a curriculum-based reinforcement learning strategy progressively refines the model's outputs.

Notably, PIT [24] can be integrated into the inference process to refine LLM outputs iteratively. Its effectiveness is demonstrated through extensive evaluations of three diverse datasets, including Anthropic/HH-RLHF and OpenAI/Summary. Regarding response quality, it outperforms prompting-based self-improvement methods, such as Self-Refine, as validated by automatic metrics (GPT-4 and DeBERTa) and human evaluations.

Building on the development of advanced interfaces and large language model (LLM) capabilities, Script&Shift [25] offers a layered interface paradigm designed to better align with natural writing processes, particularly for complex tasks. Although not explicitly designed for email, its architecture provides valuable insights into generative AI writing tools in diverse communication settings. This approach uniquely integrates content development ("scripting") with rhetorical strategy ("shifting") in a zoomable, non-linear workspace, facilitating fluid movement between drafting and organization.

The system's technical foundation rests on "layer primitives": distinct modules, including the Writing Layer for content creation, the Meta Layer for global context (audience, tone, and goals), and the Document Layer for compilation. These layers enable dynamic interaction with content generated by an LLM (Claude 3.5 Sonnet, in this case) through features such as embedded placeholders, context-sensitive prompts, and output rendering that respects document structure. Coordination is handled by a Prompt Composer, which formulates system instructions based on task knowledge, and a Workspace Manager, which orchestrates component interactions, ensures structural consistency, and manages content distribution across layers.

In summary, the reviewed tools and frameworks demonstrate a clear evolution from prompt-based generation to more interactive and structured co-writing systems. Innovations such as ABScribe and Script&Shift illustrate a growing emphasis on user control and interface design. At the same time, approaches like SELF-REFINE and Shepherd suggest promising avenues for improving generation quality through feedback and self-assessment. However, most tools still lack deep integration with user-specific context or real-time personalization mechanisms, signaling a gap between interface design and underlying model adaptation.

### 4.2 Personalization Techniques Found on LLM Writing Assistants

While generic LLMs demonstrate remarkable performance in text generation, the true promise of artificial intelligence in email communication hinges on its capacity to provide highly personalized responses. In this subsection, we discuss advanced methodologies specifically designed to personalize LLMs to the unique styles, preferences, and subtle contextual requirements inherent in email communication. We consider a variety of methods aimed at achieving significant and effective personalization, analyzing their strategies critically in terms of data efficiency, privacy, and adaptive responsiveness to users in email contexts.

Recent advances in personalized generative AI, such as PersonaAI [26], have overcome the limitations of traditional general-purpose Large Language Models (LLMs), including ChatGPT, in delivering profoundly personalized experiences, particularly in user-facing applications and sophisticated use cases, like email writing assistants. By combining Retrieval-Augmented Generation (RAG) with the LLAMA model, this layered interface significantly enhances personalization, allowing it to respond dynamically to each user's unique preferences and nuances. PersonaAI's solution uses a mobile app to capture real-time user data via voice-to-text transcription. This data is saved to a cloud database for processing, where it's formatted and converted into 384-dimensional vectors. To perform this conversion, the system utilizes an embedding model, specifically the BAAI/bge-small-en model [27] from Hugging Face, which transforms the text into numerical representations that capture its semantic meaning. This process is crucial because it enables the fast semantic retrieval of data. The system then dynamically retrieves the top-k most contextually similar contexts using a cosine similarity function, returning responses tailored to the user's specific needs.

Additionally, advanced prompt engineering fine-tunes the model to produce contextually fit content, and built-in error handling enhances the user's confidence. In addition, lightweight and scalable architecture makes it a viable option for mass-scale personalized AI deployment, and its ethical design focus guarantees user trust and privacy. High contextual retrieval precision (91%) and low query response time (<1 second) were achieved through performance testing using datasets such as simulated university journals, demonstrating the system's feasibility in real-world applications. Such technological advances underscore the potential of PersonaAI to drive personalized AI, particularly in tasks that require fine-grained communication, such as advanced email writing assistants.

While the work presented in [26] focuses on retrieval techniques for personalization, another significant advancement in this domain is the preference agent approach [28]. This method introduces a way to personalize content generated by large language models for tasks like email writing, explicitly addressing the challenge of adapting their broad capabilities to meet individual user preferences. Traditional methods, such as in-context learning and parameter-efficient fine-tuning, often struggle to capture the nuanced complexity of human preferences, particularly with smaller, personalized datasets. To overcome this limitation, the authors of [28] introduced these agents as small, locally trainable models that encode user preferences into concise natural language rules. These agents act as a "steering wheel," guiding the output of a larger, more generic large language model (LLM) to align with a personalized style and content, all without requiring fine-tuning of the larger model. This modular method separates the preference learning process from the generic LLM, offering a more scalable and flexible solution for personalization compared to the Retrieval-Augmented Generation (RAG) approach used in PersonaAI [26].

So basically, the authors Shashidhar S., Chinta A., et al. [28] build upon a systematic process for capturing user preferences. The proposed method utilizes a large LLM to generate zero-shot responses, which are then compared with the ground truth outputs to identify differences. These differences derive preference rules, which a smaller model then learns. This model becomes the personalized preference agent, capable of generating rules that guide the large LLM at inference time. During inference, the trained preference agent provides context in the form of natural language rules to the large model, allowing it to generate outputs aligned with the user's preferences. Evaluations across three diverse datasets — Enron emails, New Yorker articles, and Amazon product reviews—demonstrate that preference-guided LLMs significantly outperform both fine-tuning baselines and standard prompting methods in terms of automatic metrics (such as GPT-4o evaluation) and human judgments.

Following the research on personalization techniques, Panza [29] emerges as a novel solution focused on personalized text generation, particularly for emails, while prioritizing user privacy through local execution. It tackles the efficient challenges of fine-tuning and Retrieval-Augmented Generation (RAG) faced by predecessors. Panza uniquely combines a variant of Reverse Instructions with RAG and parameter-efficient fine-tuning (PEFT) methods, such as ROSA, enabling personalization using tiny datasets (under 100 emails) on commodity hardware. This makes it highly viable for users with limited personal data.

A key methodological contribution is its evaluation approach, demonstrating that a combination of BLEU and MAUVE scores strongly correlates with human preferences for personalized text quality. This combined metric helps validate Panza's ability to replicate writing styles effectively. Unlike purely RAG-based or preference-agent approaches, Panza provides a scalable and flexible solution that enables local execution, low-cost fine-tuning, and inference on standard hardware. Its practical utility is showcased through a Google Chrome plugin designed for Gmail integration.

The authors Shaikh O., Lam M., et al. [30] introduce a distinct method for aligning large language models (LLMs) with individual user preferences, complementing other personalization frameworks. Their "Demonstration Iterated Task Optimization" (DITTO) approach takes a more direct approach by leveraging user-provided demonstrations. This iterative process only requires a small number of prototypes (fewer than 10) to guide the model's output, making it a data-efficient solution that contrasts with the more resource-intensive methods of fine-tuning and reinforcement learning from human feedback (RLHF) often used in personalization, as already stated.

The online imitation learning method, introduced in [30] stands out by treating user demonstrations as preferred outputs and utilizing this feedback to update the model via algorithms such as Direct Preference Optimization (DPO). Compared to Panza, which focuses on fine-tuning foundation models to mimic writing styles with minimal data while prioritizing privacy, this approach offers a simpler alternative by directly optimizing model outputs based on few-shot user demonstrations. This method allows it to bypass the need to fine-tune the entire model or implement complex RAG-based retrieval systems.

Furthermore, this technique addresses some challenges inherent in the preference agents' approach, where user preferences are encoded into specific rules. While preference agents can effectively guide output, the online imitation learning framework offers a more scalable and flexible approach to leveraging direct user feedback for preference alignment, thereby reducing the need for complex rule creation and extensive preference elicitation. Its methodology also contrasts with Panza's emphasis on local execution and privacy, as it focuses on the iterative refinement of user preferences, making the system adaptable to diverse user needs without compromising performance. Evaluations using static benchmarks (such as CMCC and CCAT) and real-world user studies for email writing consistently show its effectiveness in personalizing LLMs with minimal data. By outperforming traditional methods, such as supervised fine-tuning and few-shot prompting (even with GPT-4), this system demonstrates its ability to capture fine-grained stylistic preferences with only a few demonstrations, positioning it as a powerful and efficient solution alongside the personalization techniques discussed previously.

The evolution of personalization techniques reflects a broader shift in how large language models (LLMs) are transforming individualized AI systems. Zhang et al. [9] explored this disruptive impact of Large Language Models like GPT-3.5, GPT-4, and LLaMA-7B, highlighting how LLMs transform personalization from filtering to dynamic, real-time user engagement. Unlike static embedding-based models, LLMs utilize few-shot prompting and in-context learning to dynamically adapt recommendations. Reinforcement Learning from Human Feedback (RLHF) further refines personalization by aligning responses with user intent. The study highlighted the ability of LLMs to integrate external tools, including retrieval-based recommendation engines (e.g., the Generalized Dual Encoder Model), search APIs, and vector databases, thereby enhancing context-aware personalization.

These observations provide valuable context for understanding the technological foundations that underpin the personalization approaches of PersonaAI, preference agents, Panza, and DITTO, all of which leverage these capabilities in different ways.

The reviewed personalization techniques highlight a transition from model-centric fine-tuning to modular, data-efficient personalization strategies. Approaches such as PersonaAI, Panza, and DITTO show that personalized outputs can be achieved even with limited user data, particularly through RAG and preference agents. Yet, a trade-off remains between scalability, privacy, and stylistic fidelity. While some models excel in fast deployment or local inference, others prioritize accuracy in capturing writing nuances, suggesting that no single method currently balances all personalization goals effectively.

### 4.3 Security Vulnerabilities in Generative AI Email Systems

This section highlights key threats in AI-driven email environments, ranging from ecosystem-level attacks, such as the Morris-II worm, to advanced exploits, including LLM-enabled spear phishing and Trojan attacks on fine-tuning processes. Understanding these risks is crucial for building secure and reliable AI email assistants that protect user data and maintain communication integrity.

A recent study has discovered a significant security weakness in Generative AI (GenAI) environments, particularly those powered by RAG, such as email assistants. The authors Cohen S., Bitton R., et al. [31] present Morris-II, a new computer worm that propagates across GenAI platforms via embedded adversarial prompts in emails. The result is indirect prompt injections, which can activate malicious behaviors, such as data exfiltration. The authors performed empirical experiments with mock GenAI email assistants developed using the LangChain library and RAG to examine this threat. The assistants were exposed to tailored databases extracted from real-world email datasets, such as the Enron and Hillary Clinton email datasets, to replicate the behavior of real-world GenAI systems. For instance, the Enron dataset simulated 20 workers, each with a personal database of 100 emails, whereas the Hillary Clinton dataset consisted of 1,500 emails. These datasets constituted the "external knowledge sources" of the RAG components of the assistants.

The simulations were utilized to demonstrate the worm's potential to infect the systems under various scenarios. The paper proposes Virtual Donkey, an effective defense solution that detects worm propagation according to input-output similarity in the GenAI model, as a countermeasure. The solution has high accuracy with minimal false positives. While the research acknowledges the potential for adaptive attacks, it makes a valuable contribution by revealing this ecosystem-level threat and proposing an effective solution to secure GenAI-based email assistants.

Building upon these ecosystem-level concerns, the author Hazell [32] examines the capacity of LLMs, such as OpenAI's GPT-3.5 and GPT-4, to facilitate spear phishing attacks, a sophisticated form of social engineering that leverages personalized information to manipulate targets.

The research highlights LLMs' ability to assist with the reconnaissance phase by processing unstructured data to gather target information and the message generation phase by producing realistic and contextually relevant spear phishing emails at a fraction of a cent per message. Notably, the study demonstrates how basic prompt engineering can bypass safety measures in these models, enabling the generation of malicious content and advice on conducting attacks, including crafting persuasive emails and basic malware. By generating unique spear phishing messages for a large group of UK Members of Parliament, the study provides evidence that LLMs can produce realistic and cost-effective phishing attempts, potentially scaling such campaigns significantly and lowering the barrier for less skilled cyber criminals. The findings also compare the sophistication of different LLMs, including open-source models, in this context. While the study focuses on the malicious application of LLMs for email-based attacks, it underscores the dual-use nature of this technology, and the governance challenges associated with preventing its misuse. The paper further discusses potential solutions, including structured access schemes and the development of LLM-based defensive systems for email security.

Researchers subsequently began identifying additional security vulnerabilities that could arise with the deployment of LLMs. Building on the phishing concerns identified by Hazell [32], a critical security evaluation was proposed, demonstrating how instruction-following large language models can be exploited for malicious purposes through methods borrowed from classical computer security. The authors Kang D., Li X., et al. [33] developed three primary attack vectors—obfuscation (inserting typos or replacing synonyms to escape detection), code injection/payload splitting (indirect programming of instructions), and virtualization (programming attacks in virtual scenarios)—that could evade OpenAI's content filtering mechanisms 100% of the time in situations like hate speech, conspiracy theories, and phishing schemes.

The economic implications of these vulnerabilities are significant. Using human evaluators and GPT-4 to assess generation quality, Kang et al. [33] found that instruction-tuned models of larger capacities produced much more realistic malicious content than earlier models. Their economic research revealed that tailored malicious content could be generated at a cost of between $0.0064 and $0.016 per instance, significantly less than human-generated content, which is estimated to cost $0.10 per instance. This cost-effectiveness created strong economic incentives for adversaries to deploy these systems. This identified the dual-use possibility of AI-powered email composition, where features such as personalization and productivity could be leveraged for malicious purposes, including sophisticated phishing attacks.

Beyond prompt-based vulnerabilities and economic incentives, adaptation-based security concerns have also emerged. Because of these broader security concerns, the authors Dong T., Xue M., et al. [34] analyzed another potential vulnerability in their paper, "The Philosopher's Stone: Trojaning Plugins of Large Language Models." Their research on low-rank adaptations for LLMs identified critical security concerns relevant to using AI-mediated communication systems. The researchers have identified weaknesses in LoRA adapters to demonstrate how attackers could design "Trojan plugins" that make the LLMs generate toxic text when triggered by specific phrases.

These adaptation-based attacks represent a sophisticated evolution of security threats in generative AI systems. The authors Dong T., Xue M., et al. [34] also introduced two new attack methods: POLISHED, which uses LLM-based paraphrasing to produce naturalistic poisoned datasets, and FUSION, which transforms benign adapters through an over-poisoning process. Testing on actual-world LLMs, such as Llama (7B, 13B, 33B) and ChatGLM2 (6B), confirmed the approaches, with FUSION achieving a nearly 100% success rate in producing target keywords based on just 5% poisoned data. These findings highlighted key security issues for AI-powered communication systems that utilize adapter-based fine-tuning techniques to adjust models to specific domains or users. Since adapter-based techniques are commonly used in email personalization systems, these vulnerabilities raised questions about the secure deployment of generative AI in email ecosystems, connecting back to the ecosystem-level threats identified in [31].

The security analysis reveals that personalized large language models (LLMs) for email systems are vulnerable to a wide range of attacks, including prompt injections and Trojan plugins. While solutions such as Virtual Donkey and structured access models offer partial mitigation, the literature shows that many personalization methods, particularly those relying on adapters or external retrieval, introduce new vectors for exploitation. This underscores the urgent need for security-aware personalization frameworks and defensive evaluation protocols to accompany the design of LLM-based email assistants.

### 4.4 User Perceptions of AI Communication and Writing Assistants

Beyond AI's technical capabilities, the success of email writing assistants depends heavily on user perception and interaction. This section reviews key studies highlighting the balance between AI-driven productivity and concerns about authenticity, emotional tone, and trust. Examining user experiences across contexts—from accessibility to professional use—emphasizes the need for AI that is both efficient and aligned with human communication nuances.

In 2022, Goodman et al. introduced LaMPost [35], a language model-powered email composition tool designed for dyslexic adults. The research highlighted the potential for AI systems to be re-tuned to address specific user needs rather than overall productivity enhancement. Unlike traditional spell-check and grammar-check software, LaMPost provided high-level composition support, including content structuring, subject line generation, and stylistic rewriting. The system was built with LaMDA, a conversation-specific LLM, in a browser-based email editor. The test with 19 dyslexic adults showed that the "rewrite" and "subject line" tools significantly enhanced writing productivity, but participants occasionally experienced issues with coherence and tone deviation. Perceived self-efficacy was not influenced by awareness of AI assistance, indicating that the system was employed as an empowering tool, not an intrusive helper. The study emphasized the need for adaptive AI-provided feedback to better handle cognitive diversity within writing support tools, demonstrating how personalization can be augmented to incorporate mental and accessibility dimensions beyond style considerations.

Following these findings on personalized writing support, later research investigated user experience and usage of AI-facilitated communication tools in daily life. The authors Fu Y., Foell S., et al. [36] presented an in-depth diary and interview study of user experience with tools that mediate communication through artificial intelligence in daily interpersonal interactions. They conducted the study with 15 users who used tools of their choice for one week, resulting in 227 diary entries that captured their experiences and attitudes. The study found general positive acceptability with an average satisfaction score of 7.1 on a scale of 1-10, with satisfaction rising following participants' initial learning curve. One of the key contributions was the definition of four communication spaces, distinguished by stakes (high/low) and relationship dynamics (formal/informal); these AI tools were perceived as considerably more appropriate in formal relationships than in informal ones. The participants noted that these systems were beneficial by increasing communication confidence, helping to find precise words to express ideas, overcoming cross-cultural communication challenges, and expanding vocabulary. However, the study also discovered ongoing flaws in current AI communication systems, including excessive verbosity, unnatural phrasing, exaggerated emotional intensity, and the difficulty of iterative revision required to achieve satisfactory outputs. These findings suggest that the tools must be tuned differently for specific communication contexts, with features matched to their functions, and conclusions directly applicable to crafting customized email systems that are sensitive to differing communication contexts.

Researchers shifted from educational applications to professional settings and began examining AI writing tools in business environments. In a corporate environment, the authors Jovic M. and Mnasri S. [37] compared four well-known large language models (LLMs)—ChatGPT 3.5, Llama 2, Bing Chat, and Bard—in terms of their ability to generate business emails, examining the implications for AI implementation in business communication. Using a detailed framework, the study assessed performance across three types of emails: routine, negative, and persuasive. Each LLM was subjected to identical email scenarios, with outputs scored on content, format, and tone. Despite the formulaic nature of business emails, the researchers found significant variations in quality between models. Llama 2.0 achieved the highest overall score (48.9/60), followed closely by Bing (47.8), ChatGPT 3.5 (46.7), and Bard (45.2). Common weaknesses across all models included difficulties in following the requested structure, maintaining tone consistency, and responding to emotional cues, which highlighted the need for further development of LLMs in business email generation, particularly in terms of emotional aspects. However, the study employed only these tests and did not explore or analyze prior personalization methods, indicating that further development was necessary to address these issues in tailored business communication.

Beyond performance comparisons, understanding user trust in AI-generated content emerged as a critical area of research. In a study, user perceptions of AI-generated content were explored, revealing that trust in AI-generated emails decreased as the perceived AI involvement increased [38]. Using a "Wizard-of-Oz" approach, where participants believed they were interacting with an AI system but were interacting with a human, the study found that users were more willing to accept AI-generated content for factual emails but preferred human authorship for emotionally charged content, such as condolences.

To enhance the comprehension of stylistic variation in artificial intelligence versus human writing, the authors Li W., Saha K, et al. [39] compared AI and human-generated emails and investigated the primary distinction that may influence user acceptance and perception. Based on the W3C email corpus, the research compared the syntactic, semantic, and psycholinguistic attributes of emails created by GPT-3.5, GPT-4, Llama-2, and Mistral-7B. Although the findings replicated anxieties regarding AI-generated emails being formal and emotionally monotonous, Li et al. also highlighted the difference in style: AI-generated emails were verbose and lexically redundant, whereas human-generated emails were succinct, personalized, and linguistically varied. Although polite and positive, LLM-generated emails often lack contextual specificity, underscoring the need for personalization, as demonstrated by several studies in this literature review. These findings are corroborated by a small-scale user study involving 41 participants, in which, although they praised the efficiency of AI-generated writing, they criticized its limited diversity and personalization. This highlights the reality that although LLM-based email automation is efficient, it can be complemented by hybrid approaches that balance AI-driven efficiency with user customization, thereby adding authenticity and adaptability. The study highlighted a significant trade-off of email generation systems: reconciling the grammatical accuracy and speed of AI-produced messages with the conversational, personalized tone of human-composed emails.

Studies on user perception consistently point to a tension between productivity and authenticity. While users appreciate speed and ease of use, concerns persist regarding emotional tone, verbosity, and lack of contextual precision in AI-generated emails. Importantly, acceptance varies across domains — business, accessibility, and personal contexts — highlighting that personalization must extend beyond style to include purpose, audience, and emotional resonance. Future systems will need to adapt not just to how users write, but why they write.

### 4.5 Benchmarking and Evaluation

Robust evaluation methods are crucial for accurately measuring the performance, personalization, fidelity, and impact of LLM-based email systems. This section reviews emerging benchmarks focused on user feedback responsiveness, long-form coherence, and on-device efficiency, highlighting a shift toward hybrid evaluation frameworks that combine automated metrics with human judgment.

To more effectively quantify large language models (LLMs) in interactive settings, the authors Yan J., Luo Y., et al. [40] introduced RefuteBench, a novel benchmark designed to assess an LLM's ability to systematically incorporate refuting user feedback. This approach addresses a vital challenge in generative AI: LLMs often struggle to incorporate user corrections, which limits their effectiveness in applications like email writing assistants where user-specific adjustments are essential. RefuteBench takes a dynamic approach by generating counter-instructions to test models across various tasks, including question answering, machine translation, and email writing. This contribution addresses a key gap in current instruction-following evaluations by targeting scenarios where users actively refine or correct model-generated responses. The benchmark rigorously evaluates LLMs' compliance with updated instructions through both single- and multi-feedback scenarios and introduces two novel evaluation metrics: Feedback Acceptance (FA), which assesses positive acceptance of feedback, and Response Rate (RR), which evaluates the correct incorporation of feedback into future interactions. To overcome these recognized drawbacks, the authors present a novel and pragmatic "recall-and-repeat" prompting strategy that leverages past feedback to increase model responsiveness. Experimental results showed significant gains in Response Rate across a range of LLMs, testifying to the effectiveness of this strategy. Furthermore, the study's examination of the relationship between FA and RR provides valuable insight into the importance of accepting initial feedback for ongoing compliance. Data gathering involved existing datasets, such as RIPPLEEDITS and WMT2023, along with custom-created email writing instructions to provide a comprehensive evaluation. Essentially, this study contributes to a deeper understanding of LLMs' interactive capabilities by identifying their limitations in responding to refuted instructions and proposing a viable solution to enhance their responsiveness, with direct implications for the field of generative AI assistants.

While the work by Yan et al. [40] prioritizes the use of feedback and responsiveness, a different study by the authors Xu et al. [41] tackles another significant challenge for generative AI email assistants: high inference latency in on-device Large Language Models (LLMs), which is a critical issue for systems processing long contextual prompts. Their paper introduces llm.npu, the first LLM inference system to successfully utilize on-device Neural Processing Unit (NPU) offloading to solve this problem, particularly in the dominant prefill stage. The system integrates three new techniques: chunk-sharing graphs, which improve efficiency and lower memory overhead by allowing for variable-length prompt handling through division into fixed-size chunks; shadow outlier execution, which preserves accuracy by offloading the processing of significant activation outliers to the CPU/GPU in parallel; and out-of-order subgraph execution, a scheduling framework that intelligently improves the utilization of heterogeneous mobile processors (CPU/GPU and NPU) by allocating and executing Transformer blocks based on their hardware affinity and precision sensitivity.

Rigorous testing performed by the authors of [41] on standard mobile hardware showed that the performance improvements of their system are significant, with prefill speedup rates of up to 43.6 times over GPU benchmarks and substantial power savings of up to 59.5 times, all with preserved inference accuracy. In extensive real-world usage, particularly in scenarios involving intelligent email assistants with lengthy prompts, this approach achieves latency improvements of 1.4 and 32.8 times compared to competing benchmarks. Notably, the system achieves prefill speeds of over 1,000 tokens per second for billion-parameter models on mobile devices, marking a significant advancement in enabling rapid and efficient on-device generative AI capabilities for apps that require accelerated processing of large contextual information.

Shifting from system-level advances to personalization techniques, a key contribution to the literature is provided by work on retrieval optimization in LLM personalization. The LaMP benchmark [42], which evaluates performance on seven personalized NLP tasks, stated that retrieval optimization led to a 5.5% average improvement in LLM personalization, with a 33.8% improvement in cold-start settings. Pre- and post-generation retrieval selection mechanisms were one of the novel features, which picked alternate retrieval methods per query to trade off recency, keyword relevance, and user writing style. These findings underscore the importance of adaptive retrieval selection in producing highly personalized LLM responses, with direct applications to email response generation systems.

To further the standards of personalization, the authors Kumar I., Viswanathan S., et al. [43] responded to an essential requirement within evaluation tools by introducing LongLaMP, a specialized benchmark employed to personalize the generation of long textual content across various tasks, including email writing. This innovation remains highly relevant in email because previous personalization datasets concentrated primarily on short-term textual outputs without regard for coherence and consistency issues within lengthy communications. Contrary to the state of the art at the time, LongLaMP focused more on long-term coherence, stylistic coherence, and topic consistency within lengthy passages. The authors used a RAG model where documents and features specific to users were leveraged to constrain the generation step to avoid common problems like topic drift. The LongLaMP [43] dataset was large, consisting of multi-paragraph custom text samples from email conversations, scientific abstracts, and web reviews. Quantitative metrics presented in this article indicate that RAG-improved models outperform conventional fine-tuning approaches by a range of 5.7% to 128%, as observed in various evaluation metrics, including BLEU, ROUGE-L, and METEOR. The suggested evaluation framework categorizes models along two critical dimensions: user-based (cold-start) personalization, which assesses the capacity of models to generalize to new users with minimal or no prior history, and temporal personalization, which examines how models evolve as they adjust to changing user tastes over time. Among the most noteworthy findings was that dense retrieval-based Contriever methods, which use neural networks to understand the semantic meaning of text, vastly outperformed traditional keyword-based methods, such as BM25, which rely on word frequency and occurrence, in determining beneficial personalization signals by an enormous margin, with significant implications for retrieval-based personalization methods in email systems.

Evaluation practices for personalized large language models (LLMs) are evolving, with new benchmarks addressing the incorporation of context, responsiveness to feedback, and stylistic coherence. However, inconsistencies in datasets, lack of long-form personalization metrics, and overreliance on automatic scoring still hinder comprehensive assessment. The emergence of benchmarks like RefuteBench and LongLaMP shows progress but also signals the need for hybrid evaluations that combine automated measures with human judgment — particularly for nuanced tasks like email response generation.

### 4.6 Summary of Comparative Analysis

To synthesize the findings from the reviewed literature, Table 2 provides a comparative classification of the systems and frameworks discussed in Section 4. The table maps each study's focus across core personalization techniques (RAG, PEFT, iterative refinement), user experience design, and critical research aspects like security and evaluation, serving as a visual guide to the current research landscape.

**Table 2.** Feature Comparison of Surveyed Systems for Personalized Email Generation

| Study Reference | Publication year | RAG Focus | PEFT/Fine-Tuning Focus | Iterative-Refinement | User Experience/Interaction Design | Security/Privacy Aspects | Evaluation/ User study |
|---|---|---|---|---|---|---|---|
| Luminate [17] | 2024 | | | | ✔ | | ✔ |
| REALIGN [18] | 2024 | ✔ | | | | | ✔ |
| ResQ [19] | 2025 | ✔ | | | ✔ | | ✔ |
| SELF-REFINE [21] | 2023 | | | ✔ | | | ✔ |
| Shepherd [22] | 2023 | | ✔ | ✔ | | | ✔ |
| PIT [23] | 2023 | | ✔ | ✔ | | | ✔ |
| Script&Shift [24] | 2025 | | | | ✔ | | |
| LaMPost [35] | 2022 | | | | ✔ | | ✔ |
| PersonaAI [25] | 2025 | ✔ | | | ✔ | ✔ | ✔ |
| Preference Agents [26] | 2023 | | ✔ | ✔ | | | ✔ |
| Panza [27] | 2025 | ✔ | ✔ | | ✔ | ✔ | ✔ |
| DITTO [28] | 2023 | | ✔ | ✔ | | | ✔ |
| Morris-II [30] | 2024 | | | | | ✔ | |
| llm.npu [41] | 2025 | | | | | | ✔ |

To facilitate a better understanding of the data presented in Table 2, the following points detail the definitions for each column:

1. Personalization: RAG Focus: These highlights systems leveraging Retrieval-Augmented Generation, a critical technique discussed throughout this survey for enhancing responses with contextual, external knowledge.
2. Personalization: PEFT / Fine-tuning Focus: The use of Parameter-Efficient Fine-Tuning or similar deep adaptation methods is noted, as these are central to tailoring general models to specific domains.
3. Personalization: Iterative Refinement / Feedback-Driven: This column identifies approaches that emphasize continuous improvement through self-critique, learning from preferences, or direct user feedback, which are key for evolving model performance.
4. Addresses User Experience / Interaction Design: Given that these systems are user-facing, a focus on intuitive interfaces, usability, and overall user interaction design is a significant factor for practical adoption and effectiveness.
5. Addresses Security/Privacy Aspects: With the inherent sensitivity of email communication, this aspect notes systems or studies that explicitly consider and address security vulnerabilities, defenses, or privacy-preserving mechanisms. These

threats include ecosystem-level attacks, such as the Morris-II worm, which propagates through adversarial prompts embedded in emails. The literature also highlights vulnerabilities in prompt-based attacks, with LLMs being exploited to generate convincing spear phishing emails at a low cost. Furthermore, a more advanced security concern involves adaptation-based attacks, like the Trojan plugins that compromise fine-tuning components. In response, studies propose mitigation strategies such as Virtual Donkey, a defense solution that detects worm propagation based on input-output similarity and emphasizes the need for security-aware personalization frameworks. On the privacy front, systems like Panza prioritize local execution and low-cost fine-tuning with tiny datasets to ensure sensitive data remains on the user's device.

6. Includes Evaluation / User Study: This indicates whether the system or approach has undergone formal evaluation, been tested against benchmarks, or includes studies on user perceptions, which are vital for validating efficacy and user acceptance.

The comparative overview in Table 2, along with the literature review, provides clear answers to the research questions guiding this survey.

In response to RQ1, this review identifies that effective personalization in email generation is not achieved through a single technology but rather through a multi-faceted approach that combines three core technical strategies: Retrieval-Augmented Generation (RAG), Parameter-Efficient Fine-Tuning (PEFT), and Iterative Refinement. RAG is a key strategy for providing LLMs with up-to-date, external context, such as a user's past correspondence, without the need for costly retraining. This approach is central to frameworks like PersonaAI, which uses semantic retrieval for dynamic personalization, and Panza, which leverages RAG within a privacy-preserving, locally run architecture. PEFT, in turn, addresses the challenge of adapting massive models to individual users in a scalable manner. Techniques like Low-Rank Adaptation (LoRA) enable the fine-tuning of an LLM to a user's specific writing style by training only a small fraction of the model's parameters, making personalization feasible on commodity hardware with limited data. Panza successfully employs this method and is foundational to the alignment process in DITTO, which updates a model based on user demonstrations. The field is also advancing toward more dynamic systems through iterative, feedback-driven learning. Frameworks such as SELF-REFINE enable an LLM to provide self-feedback, thereby improving its output. Meanwhile, dedicated critic models like Shepherd and implicit learning systems like PIT further enhance this capability. Alongside these core techniques, the most effective systems recognize the importance of user-centric design, innovating on the user-AI interaction with structured interfaces for creative exploration (Luminate), cognitive load reduction through guided Q&A (ResQ), and novel interfaces for managing multiple writing variations (ABScribe).

Regarding RQ2, our analysis highlights several key dimensions beyond these core strategies. The foundational technology for these systems is the Transformer architecture, with RAG and PEFT being the key methods for personalization. A critical emerging technical consideration is on-device performance to ensure privacy and low latency, with systems like llm.npu demonstrating successful offloading to mobile NPUs for significant speedups. However, the deployment of these systems introduces considerable security vulnerabilities, including ecosystem-level threats such as the Morris-II worm, the misuse of LLMs for generating convincing spear phishing emails, and the discovery of "Trojan" attacks that can compromise PEFT components. User perception studies reveal a clear trade-off: while users value the productivity gains, with tools like LaMPost showing clear benefits for users with dyslexia, they remain concerned about the authenticity of AI-generated emails, often finding them verbose and emotionally flat. This can lead to a drop in trust, especially in sensitive contexts. Finally, evaluation benchmarks are evolving beyond traditional metrics, with the emergence of more robust assessments, such as RefuteBench (for feedback incorporation), LaMP (for personalized NLP tasks), and LongLaMP (for long-form coherence), underscoring a consensus that a hybrid approach combining automated metrics with human evaluation is essential.

The primary contribution of this survey, therefore, is to synthesize and structure this diverse body of work into a single, coherent overview explicitly focused on the email response generation domain. The classification used serves as a practical, quick-reference guide for researchers and practitioners, highlighting the current state of the art and identifying key architectural patterns in the design of personalized email assistants.

## 5. Potential Benefits of Using Personalized LLM Email Assistants

Despite limited literature focused exclusively on personalized email assistants with LLMs, substantial evidence from the research indicates multiple significant benefits. These include enhanced productivity, improved personalization, and reduced cognitive load, among others. These potential benefits, identified through our literature review, are summarized in detail in Table 3.

**Table 3.** Potential benefits identified in the literature review.

| Paper | Benefits |
|---|---|
| [35],[20],[41],[43] | Enhanced Productivity and Efficiency |
| [37],[29],[26],[23],[22] | Improved Email Quality and Personalization |
| [18],[21],[25] | Support for Creative and Complex Communication Tasks |
| [20],[21],[36] | Reduced Cognitive Load and User Effort |
| [35],[22],[24],[23],[36] | Adaptive Learning, User Control, and Accessibility |
| [29],[41],[26] | Efficient Model Personalization and Deployment Strategies |

Projects like Panza [29] demonstrate that LLMs can be effectively adjusted to reflect a user's unique writing style with limited data, using techniques such Reverse Instructions and Parameter-Efficient Fine-Tuning (PEFT), as already stated. This stylistic personalization, considered fundamental for email assistants, is complemented by Retrieval-Augmented Generation (RAG), which enhances contextual awareness by leveraging previous correspondence. Recent evaluations of email composition capabilities across various LLMs indicate that models like ChatGPT demonstrate superior performance based on clarity, tone, and relevance metrics. Empirical studies using tools like M365 Copilot have revealed tangible productivity improvements, with 64% of participants reporting a positive impact on email writing quality and efficiency during six-month trials [44].

Beyond basic content generation, these systems offer substantial reductions in cognitive load. Tools incorporating LLMs facilitate the exploration of multiple text variations in a non-linear fashion, allowing users to consider different expression approaches—particularly valuable for nuanced communications. The ability to convert brief prompts into complete, personalized emails minimizes repetitive effort while maintaining communication authenticity. Integration approaches that minimize context switching further enhance workflow efficiency, with research suggesting that initially creating rudimentary requirement summaries and subsequently requesting expansion produces optimal results.

Personalized email assistants also demonstrate valuable adaptive learning capabilities through mechanisms like the "Self-Refine" concept [22] and the PIT framework [24], which enable iterative improvement based on implicit user feedback. From a credibility perspective, studies on "impersonation" capabilities indicate that properly personalized outputs maintain authenticity. Additionally, the research emphasizes the importance of user control, with participants valuing the ability to customize, edit, and remove suggestions to ensure alignment with individual communication preferences and contexts.

The technological foundation for these benefits leverages advanced Transformer architectures, practical, prompt engineering, and efficient fine-tuning methods like LoRa, making personalization increasingly possible. While complete end-to-end personalized email assistant systems remain limited in academic literature, the convergence of evidence from related fields suggests these tools can significantly transform digital correspondence by making it more efficient, contextually relevant, and aligned with users' authentic communication styles. This transformation appears particularly pronounced in structured communication tasks, where the cognitive overhead of composition can be substantially reduced without sacrificing personal voice or communication efficacy.

However, the integration of personalized LLM assistants into real-world email systems, particularly on edge devices such as smartphones or embedded enterprise platforms, introduces necessary trade-offs between model complexity, latency, and energy efficiency. While recent work like llm.npu [41] demonstrates that significant inference acceleration can be achieved via Neural Processing Units (NPUs), these optimizations often require architectural compromises, such as precision reduction or chunked execution strategies.

Thus, the deployment of high-quality, personalized assistants must strike a balance between the depth of personalization (e.g., fine-grained stylistic adaptation) and the responsiveness and energy constraints of target devices. In edge scenarios, lightweight methods such as Retrieval-Augmented Generation (RAG) and Parameter-Efficient Fine-Tuning (PEFT) become critical not only for privacy and personalization but also for enabling feasible inference under resource limitations.

## 6. Limitations and Risks of Using Personalized LLMs

Customizing LLMs for email writing presents significant challenges, particularly in terms of evaluation and practical implementation. The current assessment measures do not always accurately reflect the degree to which an LLM captures the individual's writing style, tone, or communicative intent. Most metrics rely on task-specific, pre-established quality criteria, which fall short in accounting for the complexities of personalized email generation, especially in professional and multicultural settings [44]. An underlying issue further compounds this limitation: What is the most "personal" email tone for a user to say, "I could have written that"? It remains unlikely that most users possess a consistently unique and recognizable writing fingerprint in business emails, aside from conforming to overall politeness, structural, or professional norms. Uncertainty itself is a significant challenge for developing meaningful assessment approaches.

Thus, there is an evident need for more advanced benchmarks that can evaluate personalized email outputs in real-life environments, encompassing diverse professional settings, varying formality levels, and nuanced interactions. Modern standards often overlook such nuances, and the evidence suggests that open-source models consistently fall behind closed-source models in successful email personalization. While frameworks aiming to understand email conventions and maintain stylistic consistency are crucial for effective personalization, established email categorizations often oversimplify the multi-faceted nature of real-world interactions, where single communications can blend business and personal purposes. Furthermore, an overemphasis on mimicking stylistic traits may overlook the potentially greater impact of providing rich, accurate context to already skillful large language models (LLMs). Delivering the correct contextual information – about the relationship, history, and specific goals of the communication – may be more critical than achieving perfect stylistic replication in generating an appropriate and authentic-feeling response, presenting a limitation or perhaps a necessary shift in focus for current personalization approaches.

Beyond these functional limitations, significant ethical concerns surround the use of personalized email assistants. The ability to mimic style, imperfect though it may be, poses risks of impersonation, including phishing or social engineering attacks. Privacy remains a top priority, as emails often contain personal or confidential business information. Adversaries may further utilize personalization systems to craft unwanted content or prompt the LLM to emit confidential information, as observed via Retrieval-Augmented Generation (RAG) or user history.

Achieving a balance between personalization and authenticity remains challenging. LLMs can be incapable of completely incorporating personalized personal phrases or expert technical jargon without direct, ongoing input. Practical considerations also limit personalization, including the requirement for an adequate user email history, which poses challenges in cold-start scenarios and raises privacy concerns, as well as the need for hardware efficiency to support the large-scale deployment of these systems. In all, though RAG improves personalization through the presentation of contextually relevant suggestions, it also complicates security and privacy concerns regarding the access and processing of potentially sensitive external or historical data sources. It is crucial to address these complex limitations and associated risks to ensure the ethical development and implementation of personalized large language model (LLM) email assistants.

## 7. Future Research Directions

Building upon the insights and limitations identified in this survey, we propose that future research should focus on the following critical areas to bridge the gap between powerful foundation models and the nuanced requirements of personalized LLM-based email systems:

- Understanding Personal Style Deeply: Future work must rigorously investigate and quantify what truly makes an email feel "personal." This involves separating a user's unique writing style from what's simply appropriate for the situation or standard professional norms. The goal is to understand if a consistent "writing fingerprint" exists in emails and how to best balance mimicking style with providing accurate context for authentic responses.

- Maximizing Intrinsic LLM Capabilities for Streamlined Composition: Capitalize on LLMs' inherent excellence in text generation and rewrite by developing unified AI assistants that drastically reduce the need for multiple prompts and iterative manual refinements. This involves designing platforms that intelligently leverage the full spectrum of LLM capacities to produce high-quality, personalized email drafts in fewer steps, moving towards a single-interaction, highly efficient composition experience.

- Smarter AI-Human Teamwork: Develop advanced systems where AI models learn effectively from user feedback and actively guide users through complex email tasks. This means creating intuitive tools that help manage conversation context, choose the right tone or persona, and even proactively suggest next steps, making the collaborative writing process seamless and efficient.

- Better Ways to Measure Success: Establish comprehensive and standardized methods to evaluate the effectiveness of personalized email AI. These new benchmarks should combine automated checks with human judgment to truly capture how personalized, emotionally resonant, and coherent AI-generated emails are, moving beyond simple accuracy scores.

- Building Secure and Private AI: Include security and privacy in the initial design of personal email systems. This includes building defenses against future threats like AI-based worms, sophisticated phishing emails, and "Trojan" attacks that could compromise AI models.

- Adapting to You Over Time: Research models that can continuously learn and adapt to a user's evolving preferences and communication style in real-time. This moves beyond static user profiles, enabling the AI to truly grow with the user and effectively handle new communication challenges.

- Earning Trust and Ethical Use: Addressing the Dual Nature of AI by Developing Strong Ethical Guidelines for Personalized Email Assistants. Research should focus on building user trust by ensuring AI-generated content is authentic, avoiding being overly verbose or emotionally flat.

## 8. Conclusion

This survey consolidates the fragmented yet rapidly growing body of research on personalized Large Language Models (LLMs) for generating email responses. By combining a systematic methodology with a thematic synthesis across 25 recent studies, we map the evolving technical landscape and identify key architectural patterns, including Retrieval-Augmented Generation (RAG), Parameter-Efficient Fine-Tuning (PEFT), and iterative refinement mechanisms.

Rather than reiterating individual findings, we emphasize a broader insight: effective email personalization demands not only technical sophistication but also an alignment between system capabilities and human communication needs, including tone, purpose, and trust. The convergence of lightweight fine-tuning, secure on-device deployment, and responsive interfaces marks a pivotal shift toward scalable, privacy-aware assistants that support real-world writing workflows.

In the future, interdisciplinary collaboration will be essential. Research in NLP, cybersecurity, HCI, and applied ethics must converge to ensure that personalized email assistants evolve responsibly, enhancing productivity without compromising authenticity or privacy. As generative AI systems continue to shape digital communication, understanding how to build *trustworthy*, *adaptive*, and *human-centered* assistants remains an open frontier worthy of sustained exploration.

## References


1. The Radicati Group, Inc. Email Statistics Report, 2023-2027; 2023;
2. ZeroBounce Email Marketing Statistics Report 2025.
3. Naveed, H.; Khan, A.U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Akhtar, N.; Barnes, N.; Mian, A. A Comprehensive Overview of Large Language Models. 2023.
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. 2017.
5. Dredze, M.; Wallach, H.M.; Puller, D.; Brooks, T.; Carroll, J.; Magarick, J.; Blitzer, J.; Pereira, F. Intelligent Email: Aiding Users with AI;
6. Zhao, W.X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. A Survey of Large Language Models. 2023.
7. Kannan, A.; Kurach, K.; Ravi, S.; Kaufmann, T.; Tomkins, A.; Miklos, B.; Corrado, G.; Lukács, L.; Ganea, M.; Young, P.; et al. Smart Reply: Automated Response Suggestion for Email. In Proceedings of the Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; Association for Computing Machinery, August 13 2016; Vol. 13-17-August-2016, pp. 955–964.
8. Chen, M.X.; Cao, Y.; Tsay, J.; Chen, Z.; Lee, B.N.; Zhang, S.; Wang, Y.; Sohn, T.; Bansal, G.; Lu, J.; et al. Gmail Smart Compose: Real-Time Assisted Writing. In Proceedings of the Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; Association for Computing Machinery, July 25 2019; pp. 2287–2295.

9. Zhang, S.; Dinan, E.; Urbanek, J.; Szlam, A.; Kiela, D.; Weston, J. Personalizing Dialogue Agents: I Have a Dog, Do You Have Pets Too?; Association for Computational Linguistics;
10. Zhang, Z.; Rossi, R.A.; Kveton, B.; Shao, Y.; Yang, D.; Zamani, H.; Dernoncourt, F.; Barrow, J.; Yu, T.; Kim, S.; et al. Personalization of Large Language Models: A Survey. 2024.
11. Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; de Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-Efficient Transfer Learning for NLP. 2019.
12. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. 2021.
13. Tan, Z.; Liu, Z.; Jiang, M. PERSONALIZED PIECES: Efficient Personalized Large Language Models through Collaborative Efforts;
14. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 2020.
15. Salemi, A.; Kallumadi, S.; Zamani, H. Optimization Methods for Personalizing Large Language Models through Retrieval Augmentation. In Proceedings of the SIGIR 2024 - Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval; Association for Computing Machinery, Inc, July 10 2024; pp. 752–762.
16. Chen, J.; Liu, Z.; Huang, X.; Wu, C.; Liu, Q.; Jiang, G.; Pu, Y.; Lei, Y.; Chen, X.; Wang, X.; et al. When Large Language Models Meet Personalization: Perspectives of Challenges and Opportunities. 2023, doi:10.1007/s11280-024-01276-1.
17. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 Statement: An Updated Guideline for Reporting Systematic Reviews. The BMJ 2021, 372.
18. Suh, S.; Chen, M.; Min, B.; Li, T.J.J.; Xia, H. Luminate: Structured Generation and Exploration of Design Space with Large Language Models for Human-AI Co-Creation. In Proceedings of the Conference on Human Factors in Computing Systems - Proceedings; Association for Computing Machinery, May 11 2024.
19. Fan, R.-Z.; Li, X.; Zou, H.; Li, J.; He, S.; Chern, E.; Hu, J.; Liu, P. Reformatted Alignment. 2024.
20. Miura, Y.; Yang, C.-L.; Kuribayashi, M.; Matsumoto, K.; Kuzuoka, H.; Morishima, S. Understanding and Supporting Formal Email Exchange by Answering AI-Generated Questions. In Proceedings of the Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems; ACM: New York, NY, USA, April 26 2025; pp. 1–20.
21. Reza, M.; Laundry, N.; Musabirov, I.; Dushniku, P.; Yu, M.; Mittal, K.; Grossman, T.; Liut, M.; Kuzminykh, A.; Williams, J.J. ABScribe: Rapid Exploration & Organization of Multiple Writing Variations in Human-AI Co-Writing Tasks Using Large Language Models. In Proceedings of the Conference on Human Factors in Computing Systems - Proceedings; Association for Computing Machinery, May 11 2024.
22. Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. SELF-REFINE: Iterative Refinement with Self-Feedback;
23. Wang, T.; Yu, P.; Tan, X.E.; O'Brien, S.; Pasunuru, R.; Dwivedi-Yu, J.; Golovneva, O.; Zettlemoyer, L.; Fazel-Zarandi, M.; Celikyilmaz, A. Shepherd: A Critic for Language Model Generation. 2023.
24. Wang, Z.; Hou, L.; Lu, T.; Wu, Y.; Li, Y.; Yu, H.; Ji, H. Enabling Language Models to Implicitly Learn Self-Improvement. 2023.
25. Siddiqui, M.; Pea, R.; Subramonyam, H. Script&Shift: A Layered Interface Paradigm for Integrating Content Development and Rhetorical Strategy with LLM Writing Assistants. 2025.
26. Kimara, E.; Oguntoye, K.S.; Sun, J. PersonaAI: Leveraging Retrieval-Augmented Generation and Personalized Context for AI-Driven Digital Avatars. 2025.
27. Xiao, S.; Liu, Z.; Zhang, P.; Muennighoff, N.; Lian, D.; Nie, J.Y. C-Pack: Packed Resources For General Chinese Embeddings. In Proceedings of the SIGIR 2024 - Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval; Association for Computing Machinery, Inc, July 11 2024; pp. 641–649.
28. Shashidhar, S.; Chinta, A.; Sahai, V.; Hakkani-Tür, D. Unsupervised Human Preference Learning;
29. Nicolicioiu, A.; Iofinova, E.; Jovanovic, A.; Kurtic, E.; Nikdan, M.; Panferov, A.; Markov, I.; Shavit, N.; Alistarh, D. Panza: Design and Analysis of a Fully-Local Personalized Text Writing Assistant;
30. Shaikh, O.; Lam, M.S.; Hejna, J.; Shao, Y.; Cho, H.; Bernstein, M.S.; Yang, D. ALIGNING LANGUAGE MODELS WITH DEMONSTRATED FEEDBACK;
31. Cohen, S.; Bitton, R.; Nassi, B. Here Comes The AI Worm: Unleashing Zero-Click Worms That Target GenAI-Powered Applications. 2024.
32. Hazell, J. SPEAR PHISHING WITH LARGE LANGUAGE MODELS; 2023;
33. Kang, D.; Li, X.; Stoica, I.; Guestrin, C.; Zaharia, M.; Hashimoto, T. Exploiting Programmatic Behavior of LLMs: Dual-Use Through Standard Security Attacks. 2023.
34. Dong, T.; Xue, M.; Chen, G.; Holland, R.; Meng, Y.; Li, S.; Liu, Z.; Zhu, H. The Philosopher's Stone: Trojaning Plugins of Large Language Models. 2024.

35. Goodman, S.M.; Buehler, E.; Clary, P.; Coenen, A.; Donsbach, A.; Horne, T.N.; Lahav, M.; Macdonald, R.; Michaels, R.B.; Narayanan, A.; et al. LaMPost: Design and Evaluation of an AI-Assisted Email Writing Prototype for Adults with Dyslexia. In Proceedings of the ASSETS 2022 - Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility; Association for Computing Machinery, Inc, October 22 2022.
36. Fu, Y.; Foell, S.; Xu, X.; Hiniker, A. From Text to Self: Users' Perception of AIMC Tools on Interpersonal Communication and Self. In Proceedings of the Conference on Human Factors in Computing Systems - Proceedings; Association for Computing Machinery, May 11 2024.
37. Jovic, M.; Mnasri, S. Evaluating AI-Generated Emails: A Comparative Efficiency Analysis. World Journal of English Language 2024, 14, 502–517, doi:10.5430/wjel.v14n2p502.
38. Liu, Y.; Mittal, A.; Yang, D.; Bruckman, A. Will AI Console Me When I Lose My Pet? Understanding Perceptions of AI-Mediated Email Writing. In Proceedings of the Conference on Human Factors in Computing Systems - Proceedings; Association for Computing Machinery, April 29 2022.
39. Li, W.; Saha, K.; Lai, Y.; Soni, S. Emails by LLMs: A Comparison of Language in AI-Generated and Human-Written Emails. 2025, doi:10.1145/3717867.3717872.
40. Yan, J.; Luo, Y.; Zhang, Y. RefuteBench: Evaluating Refuting Instruction-Following for Large Language Models. 2024.
41. Xu, D.; Zhang, H.; Yang, L.; Liu, R.; Huang, G.; Xu, M.; Liu, X. Fast On-Device LLM Inference with NPUs. In Proceedings of the Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1; ACM: New York, NY, USA, March 30 2025; pp. 445–462.
42. Salemi, A.; Mysore, S.; Bendersky, M.; Zamani, H. LaMP: When Large Language Models Meet Personalization. 2023.
43. Kumar, I.; Viswanathan, S.; Yerra, S.; Salemi, A.; Rossi, R.A.; Dernoncourt, F.; Deilamsalehy, H.; Chen, X.; Zhang, R.; Agarwal, S.; et al. LongLaMP: A Benchmark for Personalized Long-Form Text Generation. 2024.
44. Bano, M.; Zowghi, D.; Whittle, J.; Zhu, L.; Reeson, A.; Martin, R.; Parsons, J. Survey Insights on M365 Copilot Adoption;

Instituto Superior
de Engenharia
Politécnico de Coimbra