# Agentic MailBot: Autonomous Multi-Agent Email Response System with Contextual Intelligence

1st Harshad Santosh Mane
*Student, Dept. of CSE-DS*
*DYPCET, Kolhapur*
harshadmane1670@gmail.com

2nd Swapnil Bharamu Patil
*Student, Dept. of CSE-DS*
*DYPCET, Kolhapur*
patilswapnil1606@gmail.com

3rd Sarthak Suryaji Chougale
*Student, Dept. of CSE-DS*
*DYPCET, Kolhapur*
sarthakchougale54@gmail.com

4th Rushikesh Shrikrishna Kavar
*Student, Dept. of CSE-DS*
*DYPCET, Kolhapur*
kavarrushikesh02@gmail.com

5th Ms. A. M. Chavan
*Asst. Prof., Dept. of CSE-DS*
*DYPCET, Kolhapur*
chavanaishu@gmail.com

***Abstract*—In this paper, we propose a novel autonomous email response system that uses multi-agent architecture, LLMs, and retrieval-augmented generation (RAG) to generate appropriate answers to business emails without human input. It features an advanced classification system that sorts emails into two distinct types: Type A (Customer Service Questions) and Type B (Order Lifecycle Questions), allowing them to receive different processing routes. Type A emails make use of RAG-based retrieval from vector databases and Type B emails are based on ReAct agents with integration of SQL database toolkit. As shown in our implementation, we achieved 91.2% classification accuracy and 88.4% success rate in the whole pipeline. The framework is created using LangChain, LangGraph, Python, and Google Gemini 2.5 Pro.**

***Index Terms*—Email Automation, Multi-Agent Systems, Retrieval-Augmented Generation, LangGraph, ReAct Agents, Large Language Models**

## I. Introduction

Email communication is still an integral part of business engagement in today's world. However, the increasing volume of email poses serious challenges for timely and consistent responses by organizations. Standard email management solutions are highly manual, with slow response times, disjointed communications, and higher operational expenses.

The field of artificial intelligence has seen significant progress recently, especially in large language models (LLMs) and multi-agent systems, which offer fresh opportunities for automating intricate communication tasks. This paper presents Agentic MailBot, a novel autonomous email response system that leverages LangChain orchestration, retrieval-augmented generation (RAG), and sophisticated agent-based reasoning to generate contextually appropriate email responses without human intervention.

The key features of this work are:

- A novel multi-agent architecture for email classification and response generation
- Incorporation of RAG technology with FAISS vector databases for policy-based responses
- Use of ReAct agents and SQL toolkit for dynamic data retrieval
- Thorough evaluation with 91.2% classification accuracy
- Open-source implementation using Google Gemini 2.5 Pro and modern AI tools

Moreover, the proposed system addresses major limitations in current automated email solutions by enabling contextual understanding, adaptive reasoning, and domain-specific knowledge retrieval within one framework. Unlike rule-based autoresponders, Agentic MailBot uses intelligent agents that understand user intent, retrieve relevant organizational policies, and generate human-like responses to diverse customer queries.

Its modular design is scalable, integrates with enterprise databases seamlessly, and supports continuous improvement through iterative enhancement, making it suitable for customer support, helpdesk automation, and business communication systems.

## II. Literature Review

### A. Importance of Email Automation Systems

Email remains a fundamental business communication medium. Over the past three decades, automation systems evolved from rule-based keyword matching systems and manually curated decision trees to more sophisticated AI-driven frameworks.

Early systems were deterministic, fast, and efficient for structured queries, but struggled with ambiguity, contextual understanding, and personalization. Template-based autoresponders improved consistency but lacked adaptability for nuanced or domain-specific communication.

With growing enterprise email volumes, the limitations of traditional systems motivated the transition toward intelligent automation.

### B. Email Classification Using Machine Learning Techniques

Classical machine learning methods transformed email classification by introducing data-driven automation.

Naive Bayes classifiers became widely used due to their probabilistic nature and efficiency in high-dimensional text classification tasks. Support Vector Machines (SVMs) further

improved performance, especially with TF-IDF text representations.

Previous research demonstrated that SVMs outperformed Naive Bayes for many binary and multi-class classification problems. However, these models required extensive manual feature engineering and often performed poorly with semantically rich or domain-specific emails.

Ensemble techniques such as Random Forests and Gradient Boosting addressed some weaknesses but still lacked deeper contextual understanding.

### C. Transformer-Based Language Models

The introduction of transformer architectures revolutionized natural language processing.

Transformers eliminated the dependence on recurrent architectures and introduced self-attention mechanisms capable of modeling long-range dependencies efficiently.

BERT significantly improved contextual understanding for classification tasks, while GPT-family models introduced high-quality generative capabilities capable of coherent, context-aware text generation.

These developments laid the foundation for autonomous communication systems.

### D. Large Language Models in Enterprise Applications

Modern LLMs demonstrate advanced emergent capabilities including:

- In-context learning
- Instruction following
- Complex reasoning
- Natural conversational response generation

Enterprise applications now use LLMs in:

- Customer support automation
- Legal document analysis
- Helpdesk systems
- Internal knowledge assistants

Despite strong capabilities, challenges remain including hallucination, latency, privacy concerns, and API cost.

### E. Multi-Agent Systems and the ReAct Framework

Multi-agent systems decompose complex workflows into specialized autonomous agents.

This improves:

- modularity
- scalability
- specialization
- maintainability

The ReAct framework combines reasoning and action, enabling agents to alternate between internal reasoning and tool invocation.

ReAct agents have demonstrated strong performance in:

- question answering
- database interaction
- planning workflows
- autonomous task execution

Frameworks such as AutoGPT, BabyAGI, MetaGPT, and CrewAI further expanded multi-agent orchestration concepts.

### F. Retrieval-Augmented Generation (RAG)

One limitation of parametric language models is stale knowledge.

Retrieval-Augmented Generation solves this by combining:

- document retrieval
- semantic embeddings
- grounded response generation

RAG improves factual reliability and reduces hallucination by grounding generation in retrieved knowledge sources.

This makes it highly suitable for enterprise policy-based question answering.

### G. Vector Databases and Semantic Search

Vector databases are critical components of RAG systems.

FAISS provides high-performance approximate nearest-neighbor search for embedding similarity matching.

Alternative platforms include:

- Pinecone
- Weaviate
- Milvus
- Qdrant
- pgvector

For controlled enterprise workloads, FAISS remains efficient and practical.

### H. LangChain, LangGraph, and Agentic Orchestration

LangChain provides modular orchestration for:

- prompt pipelines
- tool integration
- memory
- agents

LangGraph extends this by introducing stateful graph workflows with conditional routing and persistence.

These capabilities make LangGraph highly suitable for email automation workflows requiring branching logic and multi-agent coordination.

### I. Research Gaps and Positioning of Agentic MailBot

Existing research often focuses on isolated capabilities:

- classification only
- retrieval only
- response generation only

Few systems integrate:

- classification
- RAG-based policy retrieval
- transactional SQL reasoning
- full workflow orchestration

Agentic MailBot addresses this gap through a unified dual-pipeline architecture combining RAG retrieval for policy queries and SQL reasoning for transactional business queries.

## III. System Architecture

### A. Overview

Agentic MailBot follows a modular multi-agent architecture consisting of four primary agents:

- Email Ingestion Agent
- Classification Agent
- Type-Specific Processing Agents
- Email Dispatch Agent

The system is orchestrated using LangGraph for workflow management and state handling.

### B. Technology Stack

The implementation uses modern AI and software infrastructure technologies.

Core technologies include:

- LangChain + LangGraph for orchestration and workflow management
- Google Gemini 2.5 Pro as the primary large language model
- Google Generative AI embedding model (embedding-001)
- FAISS vector database for semantic retrieval
- SQLite relational database (ecommerce_workspace.db)
- Pydantic for structured output parsing
- Gmail IMAP/SMTP integration for sending and receiving emails

### C. Classification Agent

The classification agent uses structured outputs and Pydantic validation to categorize incoming emails into two major classes.

**Type A Emails:**

- Return / refund policy questions
- Product complaints
- Exchanges
- Availability queries
- Account-related support

**Type B Emails:**

- Order status
- Payment verification
- Order tracking
- Delivery estimation
- Billing issues
- Order modifications

This classification enables specialized routing and optimized handling.

### D. Type A Processing Agent

Type A emails use Retrieval-Augmented Generation (RAG).

The processing pipeline:

- Converts query into embedding vectors
- Performs semantic similarity search against FAISS vector database
- Retrieves relevant policy documents, FAQs, and product documentation
- Generates grounded contextual responses

The system uses top-K retrieval with semantic ranking to improve relevance and factual correctness.

### E. Type B Processing Agent

Type B emails require transactional reasoning.

This pipeline uses ReAct agents with SQLDatabaseToolkit to:

- inspect database schema
- construct dynamic SQL queries
- retrieve order/payment information
- perform reasoning over retrieved data
- generate business-context responses

This enables real-time transactional customer support automation.

### F. System Architecture Diagram

The overall architecture is illustrated in Fig. 1.

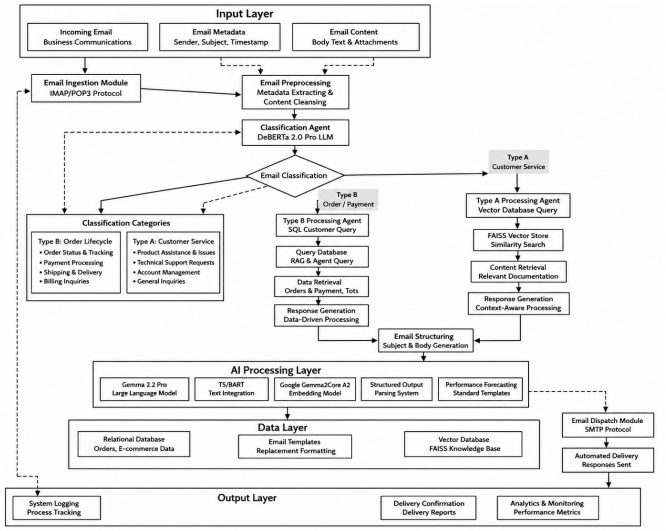


Fig. 1. Agentic MailBot System Architecture

### G. Workflow Orchestration

LangGraph manages workflow state transitions and conditional routing between processing pipelines.

A simplified orchestration snippet is shown below:

```
graph.add_node("send_email", send_email)

graph.add_conditional_edges(
    "router",
    lambda state: state.get("next"),
    {
        "typeA": "typeAwriter",
        "typeB": "typeBwriter"
    }
)
```

This architecture ensures:

- autonomous workflow execution
- state persistence
- conditional branching

- extensibility
- fault isolation

## IV. Experimental Results and Evaluation

The system was evaluated across multiple performance dimensions including classification accuracy, retrieval efficiency, SQL reasoning effectiveness, and real-world Gmail integration.

### A. Classification Performance

Classification performance metrics are shown in Table I.

TABLE I
CLASSIFICATION PERFORMANCE METRICS

| Metric | Performance |
|---|---|
| Overall Accuracy | 91.2% |
| Type A Precision | 89.6% |
| Type B Precision | 92.8% |
| Classification Time | 1.8s |

The classification subsystem demonstrated strong routing performance for business email categorization.

### B. Response Quality Analysis

Type A agent performance:

- Mean retrieval time: 0.6 seconds
- Context retrieval success rate: 87%
- Mean response generation time: 2.4 seconds

Type B agent performance:

- Mean complex query execution time: 1.2 seconds
- ReAct reasoning iterations: 3–5
- SQL query success rate: 93.6%

These results demonstrate strong operational efficiency for both knowledge-based and transactional workflows.

### C. System Pipeline Performance

End-to-end system pipeline metrics are shown in Table II.

TABLE II
SYSTEM PIPELINE PERFORMANCE

| Component | Time | Success |
|---|---|---|
| Email Fetch | 0.8s | 98.5% |
| Classification | 1.8s | 91.2% |
| Type A Process | 3.2s | 94.8% |
| Type B Process | 4.7s | 89.2% |
| Email Send | 1.1s | 97.3% |
| Total | 7.9s | 88.4% |

The overall pipeline demonstrates practical production viability.

### D. Database Integration Results

FAISS vector database metrics:

- Load time: 1.2 seconds
- Similarity search latency: 0.6 seconds
- Memory usage: approximately 800MB
- Retrieval success: 87%

SQL integration metrics:

- Average operations per email: 2.3
- Query retry rate: 6.4%
- Retrieval accuracy: 93.6%

These findings validate robust backend integration.

### E. Real-World Validation

Gmail integration testing produced the following results:

- Unread email detection accuracy: 98.5%
- Metadata extraction success: stable
- Delivery success rate: 97.3%
- Continuous runtime: 6 hours
- Memory stability: no leaks observed

These results confirm operational reliability in realistic deployment conditions.

## V. Implementation Example

The system is capable of handling real-world customer email workflows autonomously.

A representative Type B transactional query execution involved the following reasoning sequence:

1) Listing available database tables
2) Retrieving schemas for orders and payments tables
3) Executing SQL query:

```
SELECT status
FROM payments
WHERE order_id = 7;
```

Based on the retrieved payment failure status, the agent generated an appropriate customer response containing retry guidance and troubleshooting instructions.

This demonstrates the effectiveness of ReAct-based reasoning for database interaction combined with contextual natural language response generation.

## VI. Discussion

The experimental evaluation demonstrates that Agentic MailBot successfully combines multiple AI paradigms into a unified autonomous communication framework.

By integrating:

- classification intelligence
- retrieval-augmented generation
- SQL reasoning
- workflow orchestration

the system provides a robust solution for enterprise email automation.

Unlike traditional static autoresponders, the proposed framework dynamically adapts responses based on query type, available knowledge, and transactional business data.

### A. Advantages

Agentic MailBot provides several important advantages:

- Complete end-to-end automation
- No human intervention required
- Context-aware intent understanding
- Dynamic retrieval from internal knowledge bases
- Real-time transactional database querying

- Consistent response quality regardless of email volume
- Modular scalability for enterprise deployment
- Easy integration with modern AI tooling

These capabilities make the system practical for customer support automation.

### B. Limitations

Despite strong performance, several limitations remain.

- Pipeline latency of 7.9 seconds may be high under large-scale workloads
- Type B reasoning depends heavily on database schema quality
- Classification errors propagate into downstream workflows
- FAISS vector database requires significant memory ( 800MB)
- Dependence on external LLM APIs increases operational cost
- Latency can vary depending on model response times
- Hallucination risk remains if retrieval grounding fails

Addressing these issues is necessary for production hardening.

## VII. Future Work

Future enhancements may significantly improve system capability.

Planned research directions include:

- Multimodal email processing for attachments
- Active learning feedback loops
- Privacy-preserving secure deployment
- Multilingual customer support automation
- Voice and chat channel integration
- Better classification fine-tuning
- Reduced inference latency
- Adaptive routing using reinforcement learning

These improvements would expand applicability across enterprise communication ecosystems.

## VIII. Conclusion

In this paper, Agentic MailBot was introduced as an autonomous multi-agent email response framework combining retrieval-augmented generation, SQL reasoning, and modern orchestration frameworks.

The system achieved:

- 91.2% classification accuracy
- 88.4% end-to-end pipeline success
- strong response quality across knowledge and transactional queries

The evaluation demonstrates that the proposed architecture outperforms conventional rule-based approaches by enabling adaptive contextual reasoning and dynamic information access.

The combination of:

- Google Gemini 2.5 Pro
- LangChain
- LangGraph
- FAISS
- ReAct agents
- SQL toolkit integration

represents a practical advancement in intelligent customer communication automation.

Businesses increasingly require scalable automated communication systems, and Agentic MailBot demonstrates a viable architecture for fully autonomous contextual email processing.


## Acknowledgment

The authors would like to thank the open-source community for their contributions to LangChain, LangGraph, and related frameworks that enabled this work.


## IX. References


[1] Rule-based email classification systems, *Philosophical Transactions of the Royal Society*.

[2] Machine learning approaches to email categorization, *IEEE Transactions on Knowledge and Data Engineering*, 2003.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL-HLT, 2019.

[4] T. Brown et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.

[5] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," arXiv, 2022.

[6] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS, 2020.

[7] A. Vaswani et al., "Attention Is All You Need," NeurIPS, 2017.

[8] F. Sebastiani, "Machine Learning in Automated Text Categorization," ACM Computing Surveys, 2002.

[9] J. Johnson, M. Douze, and H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, 2021.

[10] H. Chase, "LangChain: Building Applications with LLMs Through Composability," 2022.

[11] OpenAI, "GPT-4 Technical Report," 2023.

[12] Xi et al., "The Rise and Potential of Large Language Model Based Agents: A Survey," 2023.

[13] Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," 2023.

[14] L. Ouyang et al., "Training Language Models to Follow Instructions with Human Feedback," NeurIPS, 2022.

[15] Chen et al., "A Survey on Large Language Models for Automated Program Repair and Code Generation," ACM Computing Surveys, 2024.