

# An Agentic AI Framework for Intelligent Support Ticket Classification, Routing, and Automated Resolution

Prof. Swathi Mahindrakar<sup>1</sup>, Vijay H<sup>2</sup>

<sup>1</sup>Assistant Professor <sup>2</sup>Student, 4th Semester — Department of Computer Applications, Master of Computer Applications (MCA), G M University Davangere

**Abstract**—In modern enterprise environments, Information Technology Service Management (ITSM) and customer support desks handle large volumes of incoming unstructured service requests daily. Traditional support ticket workflows rely heavily on manual triage or rigid, keyword-based rule engines, which frequently lead to misclassification, inappropriate priority assignment, repeated ticket reassignments, prolonged Mean Time to Resolution (MTTR), and significant operational overhead. While classical machine learning algorithms and standalone Large Language Model (LLM) chatbots offer localized improvements, they lack autonomous reasoning, multi-step problem-solving capability, domain knowledge grounding, and rigorous safety verification. This paper proposes a novel, end-to-end Agentic AI Framework for Intelligent Support Ticket Classification, Routing, and Automated Resolution. The architecture decomposes enterprise ticket-lifecycle management into a collaborative multi-agent ecosystem coordinated via graph-based state orchestration, integrating eight specialized agents — Understanding, Classification, Priority, Routing, Retrieval, Resolution, Verification, and Human Escalation. The methodology establishes mathematical formulations for agent state transitions, semantic retrieval over dense vector stores, and dual-threshold confidence calibration, and a comprehensive evaluation framework is defined across classification accuracy, routing efficiency, first-contact resolution, and retrieval precision. By coupling autonomous multi-agent reasoning with deterministic verification and human oversight, the proposed framework provides an auditable, scalable, and secure foundation for next-generation autonomous enterprise IT operations.

**Index Terms**—Agentic AI, Multi-Agent Systems, Support Ticket Management, Intelligent Routing, Retrieval-Augmented Generation, Natural Language Processing, Human-in-the-Loop, Enterprise IT Automation.

## NOMENCLATURE

The following symbols and abbreviations are used consistently throughout Sections VII, VIII, and X.

Table — Symbols

| Symbol                            | Definition                                                              |
|-----------------------------------|-------------------------------------------------------------------------|
| $T_{raw}$                         | Raw, unstructured ticket input string                                   |
| $E(\mathcal{E})$                  | Extracted entity dictionary (software, hardware IDs, error codes, IPs)  |
| $I$                               | Parsed user intent string                                               |
| $\bar{T}$ (T-bar)                 | Cleaned, normalized problem summary                                     |
| $C_k, P(C_k   T)$                 | Assigned category label and its softmax confidence                      |
| $\mathcal{P}(P)$                  | Evaluated priority tier {Low, Medium, High, Critical}                   |
| $Q_{dest}$                        | Destination engineering resolver queue                                  |
| $\mathcal{K}_{retrieved}(K)$      | Set of top-k retrieved knowledge chunks                                 |
| $R_{candidate}$                   | Candidate resolution text generated by the Resolution Agent             |
| $V_{score}$                       | Verification grounding & safety confidence score [0, 1]                 |
| $\theta_{class}, \theta_{verify}$ | Decision-gate thresholds for classification and verification confidence |
| $\alpha, \beta$                   | Weighting coefficients in the grounding-score formulation               |
| $d$                               | Dense embedding dimensionality (384 or 768)                             |
| $k$                               | Number of top-scoring retrieved chunks per query                        |

Table — Abbreviations

| Abbrev. | Expansion                                 |
|---------|-------------------------------------------|
| ITSM    | Information Technology Service Management |
| LLM     | Large Language Model                      |
| RAG     | Retrieval-Augmented Generation            |
| SLA     | Service Level Agreement                   |
| MTTR    | Mean Time to Resolution                   |
| SOP     | Standard Operating Procedure              |
| DAG     | Directed Acyclic Graph                    |
| ANN     | Approximate Nearest Neighbor              |
| HNSW    | Hierarchical Navigable Small World        |
| NER     | Named Entity Recognition                  |
| PII     | Personally Identifiable Information       |
| BPE     | Byte-Pair Encoding                        |
| FCR     | First Contact Resolution                  |
| ARR     | Autonomous Resolution Rate                |

| Abbrev. | Expansion                    |
|---------|------------------------------|
| HER     | Human Escalation Rate        |
| HGR     | Hallucination Gating Rate    |
| MRR     | Mean Reciprocal Rank         |
| IAM     | Identity & Access Management |
| HITL    | Human-in-the-Loop            |
| SVM     | Support Vector Machine       |

## I. INTRODUCTION

The fast growth of digital enterprise infrastructure, cloud computing systems, and software-as-a-service (SaaS) solutions led to the exponential increase in IT support tickets and customer assistance requests. Helpdesk staffs handle a great number of tickets per day in regards to hardware issues, network problems, complicated software bugs, access management, and administrative requests. Properly handling such tickets is essential for providing the necessary continuity of operations, compliance with SLA requirements, and ensuring productivity of organizations.

Even though the considerable amount of resources was invested in new ITSM software, the process of handling customer support tickets in modern helpdesk systems is still greatly limited by human involvement and primitive rules of classification. Upon receiving a ticket written in unstructured form and natural language, support coordinator needs to manually analyze the request, understand the meaning behind the request, recognize the domain of the problem, define the priority level of the ticket, and direct it to relevant engineering team. The process of manual ticket classification is accompanied by a number of inefficiencies: subjective classification due to human dispatcher involvement; misinterpretation of the technical context that makes tickets "bounce" between departments (Application Support, Database Administration, and Infrastructure Engineering); and spending much time by Tier-1 engineers on answering simple requests (password resets, VPN configuration, access permissions).

Initial approaches made use of keyword matching heuristics and regular expressions, which failed to address issues such as lexical ambiguity, spelling variations, polysemies, and conversational language. The following introduction of classical ML classifiers like Naive Bayes, SVM, and Random Forests with TF-IDF increased the

accuracy of categorical classification, but these models are pure static, discriminative classifiers: they cannot reason about complex problem statements, infer contextual variables, retrieve troubleshooting runbooks on the fly, or generate a sequence of resolution actions.

The latest advent of LLMs and transformer architectures [1] in NLP marked a new era in the field, but deploying such monolithic LLMs in the support process within enterprises creates several risks, namely: (i) hallucinations and factuality drift, that is the generation of plausible yet invalid troubleshooting actions that may lead to security or operational hazards — a failure mode quantified directly by hallucination-corpus benchmarks such as RAGTruth [7]; (ii) the lack of enterprise context: general-purpose LLMs have no access to enterprise-specific knowledge bases, historical resolutions, network graphs, or enterprise-access policies, motivating retrieval-grounded designs [4], [5]; (iii) the inability to orchestrate a sequence of tasks: single-prompt conversations are incapable of performing multi-stage diagnostics, validation, and invocation of enterprise-specific tools, a gap that multi-agent orchestration frameworks [8], [9], [10] are designed to close.

As a solution to the above-discussed problems, this paper presents an Agentic AI Framework for Intelligent Support Ticket Classification, Routing, and Automated Resolution. The transition from reactive text production to agentic systems allows one to transform the process of dealing with support tickets into a highly structured one. The decomposition of the entire life cycle of the support ticket into a coherent system of agents allows one to provide semantic comprehension of unstructured complains, contextual classification [2], [16] and multivariate prioritization, optimized routing [14], enterprise knowledge extraction using Retrieval-Augmented Generation (RAG) [4], [5], automated generation and self-reflective validation of solutions [6], [7], and human intervention for exceptional cases [11].

## II. PROBLEM STATEMENT

Support-desk functions within modern enterprises are hindered by a number of systematic inefficiencies that harm both the quality of service and drive up costs:

- High ingestion rate and human processing — support agents spend thousands of man-hours deciphering and analyzing tickets, resulting in cognitive overload and SLA violations.
- Category/priority errors — human dispatchers frequently classify tickets in the wrong category or overprioritize them due to lack of full information, which results in delays for critical issues and excessive prioritization of non-critical problems.
- Incorrect routing and ticket "bouncing" — mistakes in initial routing require additional routing to another team, increasing MTTR and losing context in the troubleshooting process.
- Fragmentation of knowledge base and inefficient search — documentation, SOPs, and previous ticket resolutions are spread across different systems, making it hard for engineers to find relevant solutions in times of emergency.
- Repetitive tier-one issues — a large proportion of daily tickets are actually routine inquiries which do not require any advanced troubleshooting, consistent with the ticket-category distributions reported by AutoML-based classification studies [12].
- Fragility of current automation tools — rule engines fail when a new situation arises outside of hardcoded logic; classic ML ticket classifiers cannot summarize information and provide any resolution and, finally, unconstrained LLM chatbots may hallucinate and produce dangerous instructions, a risk formally characterized by recent hallucination-mitigation research [6], [7].

There is therefore the obvious need to have an integrated multi-agent AI system, which can integrate the natural language reasoning capabilities of LLMs, the grounding of facts using RAG, mathematical

logic of the classification algorithms, and safe verification by humans in the loop.

## III. LITERATURE REVIEW AND RELATED WORK

IT support ticketing systems' automation has gone through five different broad stages: lexical matching stage, supervised machine learning stage, contextual transformer embeddings stage, retrieval-augmented generation stage, and finally, the autonomous multi-agent stage.

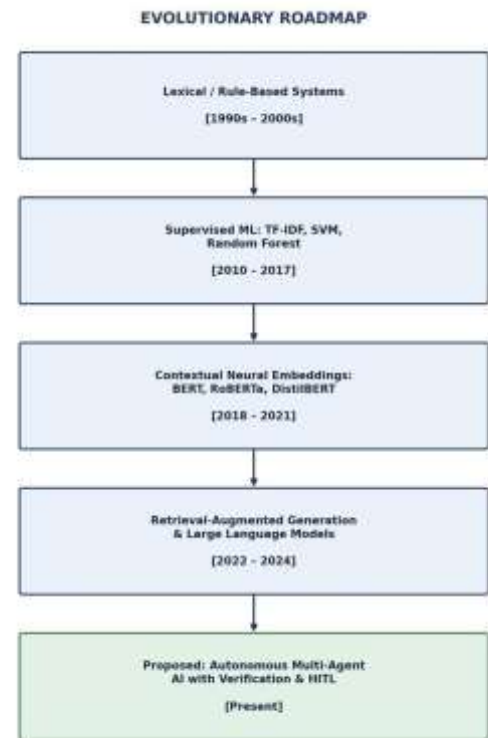


Fig. 1. Evolutionary roadmap of enterprise ticket-automation paradigms.

### A. Prior Work Summary

Table I. Comparative Mapping of Prior Work vs. Proposed Framework

| Ref.                            | Approach                                                                      | Strength                                                                        | Limitation                                                                                         |
|---------------------------------|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Truss & Böhm [12]               | AutoML (Vertex AI) for IT/support ticket category & escalation classification | Democratizes ML dev for non-specialist teams; strong F1 on small, balanced sets | Purely discriminative; label ambiguity causes misclassification; no reasoning/retrieval/resolution |
| Devlin et al. [2]               | Bidirectional Transformer pre-training (MLM+NSP), fine-tuned heads            | Deep bidirectional context; strong polysemy handling                            | Purely discriminative; high compute overhead                                                       |
| Reimers & Gurevych [3]          | Siamese/triplet SBERT sentence embeddings                                     | Fast, accurate sentence-level similarity at scale                               | Embeddings only — needs external vector-index infra                                                |
| Gao et al. [4]                  | Survey taxonomy of RAG architectures (naive, advanced, modular)               | Systematizes retrieval/augmentation/generation enhancements                     | Survey-level; no verified, agent-orchestrated pipeline                                             |
| Asai et al. [6]                 | Self-RAG: on-demand retrieval with self-reflection/critique tokens            | Critiques own grounding/relevance during generation                             | Self-critique internal to one model; no independent verifier or escalation gate                    |
| Wu et al./Chase et al. [8],[10] | Multi-agent conversational graphs (AutoGen/LangGraph)                         | Decomposes complex tasks; branching & self-correction                           | Risk of infinite loops/cascading errors w/o state validation                                       |
| Ahuja et al. (VIGIL) [13]       | Edge agentic AI, on-device diagnose-retrieve-remediate loop                   | 10-wk enterprise pilot; policy-gated remediation, trust gains                   | Endpoint remediation only, not full ticket lifecycle                                               |
| Okamoto et al. (Topaz) [14]     | Explainable, auditable model routing for agentic workflows                    | Interpretable, steerable routing via NL rationale                               | General LLM-to-LLM routing; not ticket/SLA/team-domain specific                                    |

| Ref.                    | Approach                                                   | Strength                                                  | Limitation                                                   |
|-------------------------|------------------------------------------------------------|-----------------------------------------------------------|--------------------------------------------------------------|
| Monarch & Munnally [11] | Uncertainty sampling & confidence-gated human intervention | Safety bounds and accountability for critical deployments | Often a post-hoc filter, not an integrated state-aware agent |

This paper adds to the existing literature that covers all five generations of ticket-automation research, and it extends that literature with three 2024–2026 works that sit closest to the present proposal. Truss and Böhm [12] confirm that AutoML lowers the barrier to ticket classification but remains confined to categorical labelling. Ahuja et al. [13] (VIGIL) demonstrate, in a real 10-week enterprise pilot, that policy-governed agentic AI can autonomously diagnose and remediate a majority of endpoint IT issues — but their architecture is edge-resident and scoped to device remediation rather than the full ticket lifecycle (classification, prioritization, routing, retrieval, resolution, and verification) addressed here. Okamoto et al. [14] (Topaz) show that agentic routing decisions can be made auditable and explainable, a property the proposed Intelligent Routing Agent adopts but extends with workload- and SLA-aware, ticket-domain-specific criteria. Two persistent weaknesses remain visible across this expanded set: (i) lexically or even fine-tuned transformer classifiers, and AutoML pipelines built on top of them, do nothing beyond categorical labeling; and (ii) while response-generation systems like RAG and Self-RAG [4], [6] base their output on retrieved text or internal self-critique, none of the surveyed systems combine an independently-scored, dual-threshold verification gate with a calibrated human-escalation protocol across the entire ticket lifecycle.

### B. Research Gaps Identified

Gap 1 – Lack of End-to-end Lifecycle Automation. The majority of research articles — including recent agentic systems such as VIGIL [13] — address classification, routing, knowledge search, and response formulation (or, in VIGIL's case, on-device diagnosis and remediation) as separate concerns without a unifying strategy for moving from ticket intake through to validated resolution.

Gap 2 – Lack of Self-correction & Verification. Modern LLM and RAG systems, even self-reflective variants such as Self-RAG [6], create responses without an additional, independent process that verifies their correctness, accuracy, and compliance with policy before delivery; hallucination benchmarks such as RAGTruth [7] confirm that grounding failures persist even in retrieval-augmented pipelines.

Gap 3 – Lack of Contextual, Multi-dimensional Priority Modelling. Traditional priority management is conducted via static user-selected dropdown menus (possibly exaggerated) and simple keyword triggers without consideration of business impact, infrastructure impact, and risk of SLA breach.

Gap 4 – Static Routing with Ignored Real-time Operation Status. Existing routing models, even ML-based and explainable routing frameworks such as Topaz [14], map tasks to models or categories to predefined static team queues without regard to current skillsets, backlog, and available capacity in an enterprise ticketing context.

Gap 5 – Weak Human-in-the-loop Mechanism. Automation processes either demand full manual confirmation (defeating the point of automation), or act independently (creating operational risks), while a calibrated confidence threshold escalation method is yet to be discovered.

## IV. OBJECTIVES

The overarching goal of the research project is to develop, design, and structurally validate an autonomous multi-agent AI architecture to manage enterprise support tickets. The specific goals are to:

1. design an Agentic AI Framework consisting of multiple specialized autonomous agents for comprehensive management of support ticket lifecycles;

2. provide context-aware understanding of tickets through identification of intents, entities, system parameters, and sentiment within unstructured ticket text;
3. perform multi-class classification of tickets across distinct operational domains along with confidence scores;
4. incorporate a multi-dimensional assessment of priority based on user urgency, business impact, scope of issue, and SLA parameters;
5. design an intelligent routing system where classified tickets can be assigned to appropriate teams based on the domain of expertise, past performances, and capacity;
6. build a RAG-based Knowledge Retrieval Agent to retrieve SOPs, documentation, and resolved ticket histories from vector stores via dense embeddings;
7. develop an Automated Resolution Agent to generate safe troubleshooting procedures for recurring tickets;
8. develop a reflective Verification Agent to verify the proposed solution based on the obtained ground truth document to prevent hallucinations;
9. develop a Human-in-the-Loop Escalation Protocol which will be used to safely escalate tickets that require human intervention, providing all necessary diagnostics; and
10. develop an evaluation method including metrics for classification, routing efficiency, information retrieval, and agent safety.

## V. PROPOSED AGENTIC AI FRAMEWORK

In order to rectify the shortcomings and deficiencies of the previous literature, the current paper proposes an eight-agent autonomous system. Every agent acts as a unique entity, with its own scope, constraints on prompts, input-output interface, and external toolset (see Fig. 2).

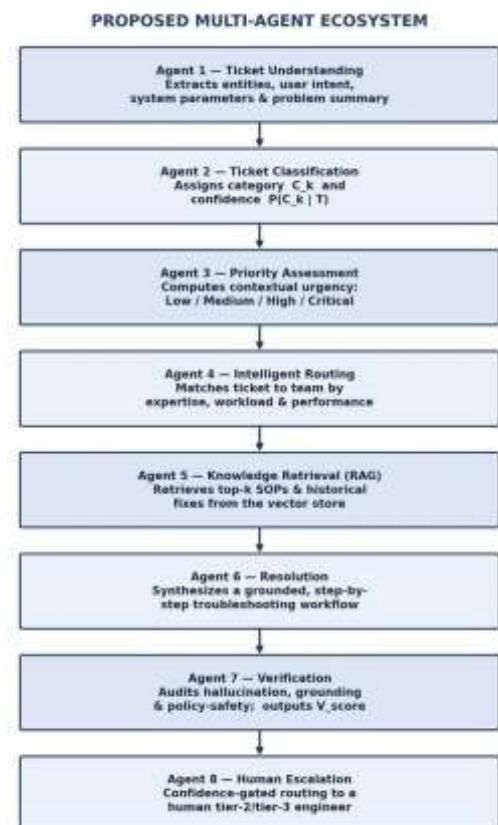


Fig. 2. Proposed multi-agent ecosystem for the ticket lifecycle.

**Agent 1 – Ticket Understanding Agent.** This agent parses unstructured and often noisy information contained within ticket

descriptions and transforms it into structured JSON metadata, using contextual transformer encoders for Named Entity Recognition [2]: it removes email headers, signatures, and boilerplate, extracts named entities (names of software, names of hardware, error codes like HTTP 500 or ORA-01017, IP addresses, names of employees); it identifies user's main intention (Access Request, System Incident, Configuration, Bug Report); it creates a normalized description of a technical problem.

**Agent 2 – Ticket Classification Agent.** This agent translates the obtained description and named entities into a standardized enterprise taxonomy (Authentication & Identity Management, Software & Application Support, Network & Connectivity, Hardware & Infrastructure, Database & Storage Services, Security & Compliance Incidents, and General IT Inquiries) using fine-tuned transformer classifiers [2], [16], building on the AutoML feasibility results reported for real IT ticket data [12], and assigns a category  $C_k$  with confidence  $P(C_k | T) \in [0, 1]$ .

**Agent 3 – Priority Assessment Agent.** This agent calculates the operational priority  $\in \{\text{Low, Medium, High, Critical}\}$  regardless of user-chosen, often biased flags based on the user-expressed urgency, business "blast radius", system criticality, and risk of SLA breach.

**Agent 4 – Intelligent Routing Agent.** Determines which of the following resolver groups the ticket should be routed to: Application Support, Network Engineering, Systems Administration, Database Administration, Cyber Security Operations, or Identity & Access Management depending on technical fit, current queue depth, real-time workloads, and past resolution efficiency — extending the auditable, explainable routing-decision principle of Topaz [14] with ticket-domain, workload-aware criteria.

**Agent 5 – Knowledge Retrieval Agent (RAG).** Embeds the question asked within the incoming ticket into a dense vector space via Sentence-Transformers [3] and does semantic Approximate Nearest Neighbor (ANN) search within the vector database (FAISS [15] / Chroma), following the retrieval-augmented generation paradigm of Lewis et al. [5] as systematized by Gao et al. [4]. It retrieves top-k chunks from the database and re-ranks them using a cross-encoder in order to generate a grounded context payload for the Resolution Agent.

**Agent 6 – Resolution Agent.** Generates accurate and actionable troubleshooting steps, grounded in the context produced by Agent 5 in a form of numbered workflows [5] without being able to perform any destructive actions independently (e.g., dropping databases, changing firewalls) without explicit authorization — a bounded, reversible-action discipline consistent with the policy-gated remediation model demonstrated by VIGIL [13].

**Agent 7 – Verification Agent.** Acts as an independent auditor of the generated resolution, extending self-reflective critique [6] and hallucination-corpus grounding methodology [7] with an explicit, independently-scored gate: hallucination auditing guarantees each statement is backed by the information from the retrieved context; policy and safety guarantees no credentials are leaked or any controls are bypassed; completeness auditing guarantees all identified symptoms are fixed. The agent generates a composite verification score  $V\_score \in [0, 1]$ .

**Agent 8 – Human Escalation Agent.** Manages interventions in which humans are brought in because of low classification confidence ( $P(C_k|T) < \theta_{\text{class}}$ ), insufficient grounding ( $V\_score < \theta_{\text{verify}}$ ), Critical priority, or high-risk/sensitive request that needs to be reviewed by administration, following the confidence-gated human-in-the-loop principle of Monarch and Munnelly [11] — the ticket is sent to a human engineer with a diagnostic summary that includes entities extracted, reasoning for the classification, potential fixes, and supporting documents.

## VI. SYSTEM ARCHITECTURE

The overall system operates as a directed, stateful multi-agent workflow graph. Information flows sequentially through analysis, retrieval, generation, verification, and decision-gating stages before terminating in either automated dispatch or human escalation (Fig. 3).

### A. Architectural Layer Description

**Ingestion and Presentation Layer.** REST API end-points and web-based UI for submission of tickets by the company's employees or external clients using multiple channels (web forms, e-mail, chat), including raw text, attachments, and metadata — mirroring the multi-channel intake patterns documented in enterprise ticket-classification studies [12].

**Agentic Reasoning and State-Graph Layer.** The central computational engine that controls eight agents, implemented using the stateful graph architecture (e.g., LangGraph [10]) that keeps a shared immutable TicketState object to store the intermediate reasoning, extracted entities, distribution of classifications, retrieved passages, and verification information.

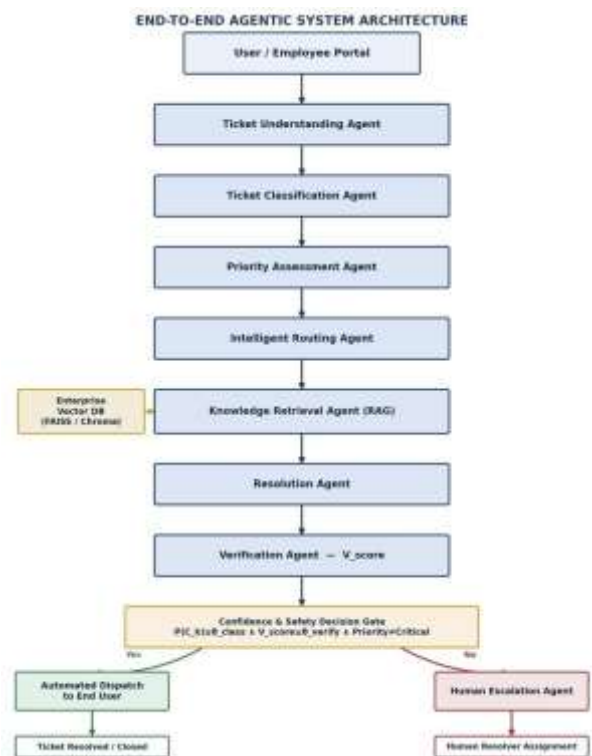


Fig. 3. End-to-end agentic system architecture with the confidence-and-safety decision gate.

**Enterprise Knowledge and Retrieval Layer.** Embeddings of chunked, dense vectors representing historical solved tickets, SOP runbooks, knowledge base articles, and security policies stored in vector indices using embedding models and vector search back-end engines (FAISS [15] / Chroma).

**Verification and Safety-Gating Layer.** Implements deterministic safety gates through computing the objective similarity and alignment scores of the outputs produced by the Resolution Agent against the reference passages provided by the Knowledge Retrieval Agent, in the spirit of the deterministic, tiered policy engine (allow / warn / deny) validated in production by VIGIL [13] and the grounding-corpus evaluation protocol of RAGTruth [7].

**Persistence and Auditing Layer.** Transaction logs in relational/document databases (PostgreSQL/MongoDB) storing traces of agent executions, token usage, confidence scores, human feedback, and final results.

## VII. METHODOLOGY

The operational process flow includes six stages starting with the dataset specification, followed by agent orchestration, and ending with the automated-resolution safety gate.

### A. Dataset Selection and Specification

For ensuring replicability without any proprietary enterprise data, the framework is defined based on the open-sourced Customer Support on Twitter / Kaggle Helpdesk dataset and the ITIL Service Desk Ticket dataset, which consists of about 20,000+ enterprise tickets interactions with natural language summary, description, category tagging, priority tagging, and resolution history for seven enterprise domains as per Section V.

### B. Data Preprocessing and Normalization

- Cleaning — removes HTML encoding, base64 attachments, and disclaimer content using regex filtering techniques.
- PII masking — identifies and masks any personally identifiable information (emails, phone number, IPs, login credentials) using Named Entity Recognition.
- Normalization — converts all text to lowercase, un-contracts technical terminology, and standardizes abbreviations ("auth," "pwd," "db," "vpn").
- Tokenization — uses Byte-Pair Encoding subword tokenization (Hugging Face Tokenizers).

### C. Classification of Tickets Modeling

Four mutually complementary classification approaches provide baseline results: (1) TF-IDF + Logistic Regression on top of n-grams ( $n \in \{1, 2\}$ ); (2) Support Vector Machines using linear and RBF kernel on top of TF-IDF vectors with L2 penalty; (3) a Random Forest model of  $N = 300$  estimators; and (4) a pre-trained Transformer (BERT-base-uncased [2] / RoBERTa-base [16]) fine-tuned with cross-entropy loss function on the target taxonomy  $\mathcal{C}$ :

$$L_{CE} = - \sum_i y_i \log P(C_i | T), \quad i = 1..|\mathcal{C}| \quad (1)$$

### D. Orchestrator and Global State Management

The agent's behavior is represented by a Directed Acyclic Graph (DAG) with conditional branching. The global state is defined as follows:

$$\mathcal{S} = \{T_{raw}, \mathcal{E}, I, \tilde{T}, C_k, P(C_k), \mathcal{P}, Q_{dest}, \mathcal{K}_{retrieved}, R_{candidate}, V_{score}, Status\} \quad (2)$$

where  $Status \in \{AutoResolved, EscalatedToHuman, PendingReview\}$ , and the other variables are defined in the Nomenclature table.

### E. RAG-Based Knowledge Retrieval Pipeline

The retrieval subsystem operates across an offline indexing phase and an online query phase (Fig. 4), consistent with the general RAG taxonomy of Gao et al. [4] and the original DPR + Seq2Seq formulation of Lewis et al. [5].



Fig. 4. RAG subsystem architecture: offline indexing (top) and online query phase (bottom).

Phase indexing (offline): enterprise SOP documents, user manuals, and historical pairs of resolved tickets  $D = \{d_1, \dots, d_M\}$  are recursively divided by chunking characters in chunks of  $L_{chunk} = 512$  tokens, having  $L_{overlap} = 64$  tokens overlap. Each chunk  $c_j$  is transformed to a dense embedding  $e_j \in \mathbb{R}^d$  ( $d = 384$  or  $768$ ) through fine-tuned Sentence-Transformer [3] (all-MiniLM-L6-v2 or bge-base-en-v1.5):

$$e_j = \text{Embed}(c_j) \quad (3)$$

using Hierarchical Navigable Small World (HNSW) graphs in FAISS [15] for indexing. Phase query (online): the Knowledge Retrieval Agent converts the ticket description  $\tilde{T}$  into a query embedding  $q = \text{Embed}(\tilde{T})$ , and cosine similarity is calculated for all indexed vectors:

$$\text{Sim}(q, e_j) = (q \cdot e_j) / (|q| |e_j|) \quad (4)$$

The top-k most similar chunks  $\mathcal{K}_{retrieved} = \{c(1), c(2), \dots, c(k)\}$  are retrieved and used by the Resolution Agent.

### F. Automated Resolution, Verification, and Safety Gate

The Resolution Agent conditions text generation on the query  $\tilde{T}$ , extracted entities  $\mathcal{E}$ , and context  $\mathcal{K}_{retrieved}$ :

$$R_{candidate} = \text{LLM}_{gen}(\text{Prompt}(\tilde{T}, \mathcal{E}, \mathcal{K}_{retrieved})) \quad (5)$$

The Verification Agent computes the grounding score  $V_{score}$  by evaluating semantic alignment and entity overlap between  $R_{candidate}$  and  $\mathcal{K}_{retrieved}$ , extending the self-reflective critique idea of Self-RAG [6] and the grounding-corpus methodology of RAGTruth [7] with an explicit, independently-scored gate:

$$V_{score} = \alpha \text{Sim}_{semantic}(R_c, \mathcal{K}_r) + \beta \text{Overlap}_{entity}(R_c, \mathcal{K}_r) \quad (6)$$

The decision gate then executes:

$$\text{Action} = \text{DispatchAuto}(R_c) \text{ if } P(C_k) \geq \theta_{class} \wedge V_{score} \geq \theta_{verify} \wedge P \neq \text{Critical}, \text{ else } \text{EscalateToHuman}(\mathcal{S}) \quad (7)$$

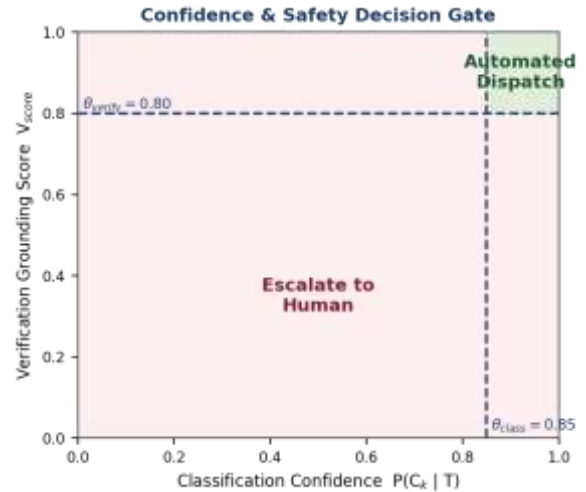


Fig. 5. Confidence-and-safety decision regions defined by the dual-threshold gate (Eq. 7).

## VIII. ALGORITHMS AND PROCEDURAL WORKFLOWS

The end-to-end agentic workflow in resolving tickets is discussed in Section VII. Algorithm 1 presents this workflow in summary.

Algorithm 1: Agentic Ticket Resolution Workflow

- 1: Ingest  $T_{raw}$  from portal / email / chat; instantiate Ticket in DB
- 2:  $\{\tilde{T}, I, \mathcal{E}\} \leftarrow \text{UnderstandingAgent}(T_{raw})$
- 3:  $\{C_k, P(C_k)\} \leftarrow \text{ClassificationAgent}(\tilde{T}, \mathcal{E})$
- 4:  $\mathcal{P} \leftarrow \text{PriorityAgent}(\mathcal{E}, \tilde{T}, C_k)$
- 5:  $Q_{dest} \leftarrow \text{RoutingAgent}(C_k, \mathcal{P}, \text{team\_state})$
- 6:  $\mathcal{K}_{retrieved} \leftarrow \text{RetrievalAgent}(\tilde{T}) \triangleright \text{ANN search over FAISS / Chroma}$
- 7:  $R_{candidate} \leftarrow \text{ResolutionAgent}(\tilde{T}, \mathcal{E}, \mathcal{K}_{retrieved})$
- 8:  $V_{score} \leftarrow \text{VerificationAgent}(R_{candidate}, \mathcal{K}_{retrieved})$
- 9: if  $P(C_k) \geq \theta_{class}$  and  $V_{score} \geq \theta_{verify}$  and  $P \neq \text{Critical}$  then

```

10: Status ← AutoResolved; DispatchAutoResolution(R_candidate)
11: else
12: Status ← EscalatedToHuman; HumanEscalationAgent(S, Q_dest)
13: Commit S to audit log; update Continuous Feedback vector store

```

This line gets explained in Section VII.E: For each ticket summary embedding  $q$ , and for each indexed chunk embedding  $e_j$ , the cosine similarity  $\text{Sim}(q, e_j)$  is calculated, and only chunks that have a cosine similarity above an alignment threshold are kept, sorted, and truncated to the top- $k=10$  chunks that are then sent to the Resolution Agent, thereby limiting the context size for the prompt and the memory usage of the retrieval log.

## IX. PROPOSED TECHNOLOGY STACK

The architecture uses modern, production-grade open-source technologies suitable for an enterprise-ready implementation (Table II).

Table II. Proposed Technology Stack

| Layer                | Technologies & Frameworks                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------|
| Frontend UI          | React.js, TailwindCSS, Axios, Vite                                                                                 |
| Backend API          | Python 3.11+, FastAPI, Uvicorn, Pydantic                                                                           |
| Machine Learning     | Scikit-learn, PyTorch, Hugging Face Transformers                                                                   |
| Embeddings & RAG     | Sentence-Transformers (all-MiniLM-L6-v2), FAISS / Chroma                                                           |
| Agent Orchestration  | LangGraph — stateful, graph-based multi-agent management                                                           |
| Large Language Model | Open-weight LLMs (Llama-3-8B-Instruct / Mistral-7B via vLLM / Ollama) or enterprise API endpoints (Claude / GPT-4) |
| Primary Database     | PostgreSQL — structured tickets, users, logs                                                                       |
| Cache & Task Queue   | Redis + Celery — asynchronous background task processing                                                           |

LangGraph [10] was chosen ahead of more lightweight multi-agent systems like CrewAI or AutoGen [8] due to its state machine and cyclic computational graph representations for agent workflows which include loops, human approval, and deterministic branches; it also supports persistent state checkpoints, whereby a workflow may pause while waiting for human action in case of escalation, and then resume after an engineer intervenes; and it is natively compatible with LangChain tool calling, HuggingFace endpoints, and vector-search retrievers.

## X. EXPERIMENTAL SETUP

This section outlines the evaluation environment and benchmarking process that is to be followed.

### A. Hardware and Software Environment

Computing Environment: A specialized computer system equipped with an Intel Core i7/AMD Ryzen 9 processor, 32 GB DDR5 memory, and NVIDIA RTX 4080 GPU with 16 GB VRAM for local transformer inference. Runtime environment: Ubuntu 22.04 LTS/Windows 11 WSL2, Python 3.11, PyTorch 2.2, and CUDA 12.1. Vector index setup: FAISS index with Flat L2 or HNSW with Cosine metric space ( $d=384$ ).

### B. Dataset Partitioning and Validation Strategy

Table III. Dataset Partitioning and Threshold Configuration

| Parameter                                        | Value                                                    |
|--------------------------------------------------|----------------------------------------------------------|
| Total labeled tickets, $N$                       | 20,000                                                   |
| Training split (70%)                             | $N_{\text{train}} = 14,000$                              |
| Validation split (15%)                           | $N_{\text{val}} = 3,000$                                 |
| Test split (15%)                                 | $N_{\text{test}} = 3,000$                                |
| Sampling strategy                                | Stratified random sampling (preserves class proportions) |
| Cross-validation                                 | 5-fold, applied to all discriminative ML baselines       |
| Classification threshold $\theta_{\text{class}}$ | 0.85 (default)                                           |
| Verification threshold $\theta_{\text{verify}}$  | 0.80 (default)                                           |

## XI. EVALUATION METRICS

Framework performance is assessed across five functional dimensions.

### Five-Dimension Evaluation Framework

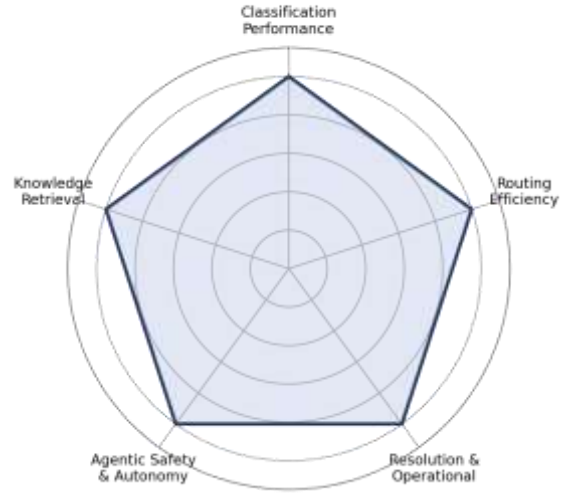


Fig. 6. Five evaluation dimensions considered by the proposed framework (scope diagram, not empirical results).

### A. Classification Performance

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (8)$$

$$\text{Precision} = TP / (TP + FP), \text{ Recall} = TP / (TP + FN) \quad (9)$$

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (10)$$

### B. Routing Efficiency

Routing Accuracy (RA) is the ratio of tickets correctly assigned to the appropriate queue on the first attempt:

$$RA = N_{\text{correct}} / N_{\text{total}}, \text{ MR} = 1 - RA \quad (11)$$

and the average number of team transfers before final resolution is:

$$\bar{\tau} = (1/N) \sum_i \tau_i \quad (12)$$

### C. Resolution and Operational Metrics

First Contact Resolution (FCR) is the percentage of tickets resolved and closed on initial automated contact without escalation:

$$FCR = (N_{\text{auto\_resolved}} / N_{\text{total}}) \times 100\% \quad (13)$$

and Mean Time to Resolution (MTTR) is the average elapsed time from submission to confirmed resolution:

$$MTTR = (1/N) \sum_i (t_{\text{resolved},i} - t_{\text{submitted},i}) \quad (14)$$

### D. Agentic Safety and Autonomy

Autonomous Resolution Rate (ARR) and Human Escalation Rate (HER) quantify the balance between automation and safety:

$$ARR = N_{\text{auto\_success}} / N_{\text{total}}, \text{ HER} = N_{\text{escalated}} / N_{\text{total}} \quad (15)$$

Hallucination Gating Rate (HGR) is the percentage of potentially inaccurate candidate resolutions correctly intercepted by the Verification Agent — analogous in spirit to the hallucination-detection benchmarks established by RAGTruth [7].

### E. Knowledge Retrieval

Precision@K and Recall@K quantify the relevance and coverage of the top-k retrieved documents, and Mean Reciprocal Rank evaluates the rank position of the first relevant document:

$$MRR = (1/|Q|) \sum_i (1/\text{rank}_i) \quad (16)$$

## XII. RESULTS AND DISCUSSION (BENCHMARKING TEMPLATES)

Note: consistent with rigorous academic practice, this section presents standardized benchmarking templates reflecting the planned experimental protocol; empirical values are marked "To be evaluated" pending execution against the dataset in Section X.

### A. Ticket Classification Performance (Benchmarking Template)

Table IV. Classification Benchmarking Template

| Model                           | Acc. (%)        | Prec. (%)       | Rec. (%)        | F1 (%)          |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|
| TF-IDF + Logistic Regression    | To be evaluated | To be evaluated | To be evaluated | To be evaluated |
| TF-IDF + SVM                    | To be evaluated | To be evaluated | To be evaluated | To be evaluated |
| Random Forest (Ensemble)        | To be evaluated | To be evaluated | To be evaluated | To be evaluated |
| Fine-Tuned BERT-base-uncased    | To be evaluated | To be evaluated | To be evaluated | To be evaluated |
| Fine-Tuned RoBERTa-base         | To be evaluated | To be evaluated | To be evaluated | To be evaluated |
| Proposed Multi-Agent Classifier | To be evaluated | To be evaluated | To be evaluated | To be evaluated |

Table V. Routing Benchmarking Template

| Routing Approach                  | Acc. (%)        | Mis-r. (%)      | Avg. Xfers      | Latency   |
|-----------------------------------|-----------------|-----------------|-----------------|-----------|
| Manual Human Dispatcher           | To be evaluated | To be evaluated | To be evaluated | 15–45 min |
| Static Keyword / Rule Engine      | To be evaluated | To be evaluated | To be evaluated | < 1 s     |
| Standard Supervised ML Classifier | To be evaluated | To be evaluated | To be evaluated | < 2 s     |
| Proposed Agentic Routing Agent    | To be evaluated | To be evaluated | To be evaluated | 2–5 s     |

Table VI. Retrieval Benchmarking Template

| Retrieval                    | Embedding                | P@3 (%)         | R@3 (%)         | MRR             |
|------------------------------|--------------------------|-----------------|-----------------|-----------------|
| BM25 Lexical Keyword Search  | N/A (sparse)             | To be evaluated | To be evaluated | To be evaluated |
| Dense Semantic Vector Search | all-MiniLM-L6-v2         | To be evaluated | To be evaluated | To be evaluated |
| Dense Semantic Vector Search | bge-base-en-v1.5         | To be evaluated | To be evaluated | To be evaluated |
| Hybrid (BM25 + Dense Rerank) | Cross-Encoder (ms-marco) | To be evaluated | To be evaluated | To be evaluated |

### D. Discussion & Guidelines for Result Interpretation

- Classification and contextualization – contextual transformers (RoBERTa [16]) and the agentic understanding module are supposed to beat TF-IDF sparsity baseline for tickets with colloquial language, misspellings, and multiple issues.
- Routing optimization – the Intelligent Routing Agent will work towards making  $\bar{\tau}$  equal 1.0, overcoming the usual bottleneck of tickets going back-and-forth between departments, building on the auditable-routing principle demonstrated by Topaz [14].
- Verification and safety – the double-threshold decision gate ( $\theta_{\text{class}}$ ,  $\theta_{\text{verify}}$ ) can serve as a tunable system that regulates the trade-off between the Autonomous Resolution Rate and the probability of hallucinating by increasing  $\theta_{\text{verify}}$  to make human interventions more frequent and hallucination-free.

## XIII. NOVELTY AND RESEARCH CONTRIBUTIONS

Table VII. Novelty Matrix — Proposed Contributions

| # | Contribution                                                                                   |
|---|------------------------------------------------------------------------------------------------|
| 1 | End-to-End Multi-Agent Lifecycle — replaces fragmented point tools with a unified graph state. |
| 2 | Decoupled Contextual Understanding — separates intent parsing from categorical classification. |

| # | Contribution                                                                                 |
|---|----------------------------------------------------------------------------------------------|
| 3 | Multi-Factor Priority Synthesis — contextually weights business impact and SLA-breach risk.  |
| 4 | Workload-Aware Intelligent Routing — considers team specialization, capacity, and backlog.   |
| 5 | Dual-Grounded RAG Retrieval — grounds resolution synthesis in verified enterprise knowledge. |
| 6 | Self-Reflective Verification Agent — audits hallucinations before responses reach end users. |
| 7 | Calibrated Human-in-the-Loop Protocol — enforces safe escalation with pre-filled AI context. |

**End-to-End Multi-Agent Orchestration.** While previous work has focused on solving isolated problems like classification, routing, or chatbot-based tickets — or, most recently, on-device remediation as in VIGIL [13] — this work offers a cohesive architecture for agents handling the end-to-end workflow of tickets, starting from ingestion all the way to validation of resolutions.

**Decoupling Intent and Semantic Extraction.** By decoupling the Ticket Understanding Agent from the Classification Agent, we can achieve higher accuracy in downstream tasks by structuring text into clean JSON entities before classification.

**Multi-Criteria Priority Modeling.** Presents priority modeling based on a dynamic balance between user-asserted urgency, business impact, and system impact, thereby avoiding queue blowup.

**Grounded Generation with Anti-Hallucination Verification.** Uses a combination of dense RAG retrieval [4], [5] along with a Verification Agent that validates generated responses using retrieved information, extending self-reflective-grounding techniques such as Self-RAG [6] and hallucination-corpus evaluation methodology such as RAGTruth [7] with an explicit dual-threshold decision gate.

**Human-in-the-Loop Safety Gate with Confidence Calibration.** Offers a mathematically robust solution where two thresholds govern automatic routing of high-confidence tickets to humans in the loop along with full context diagnostics.

## XIV. ADVANTAGES

- Major decrease in manual triaging – removes laborious reading and tagging of requests by an engineer, allows him or her to focus on more complicated tasks, consistent with the workload-reduction case made for AutoML-assisted triage [12].
- Reduced Mean Time to Resolution – immediate, automatic replies to tier one requests help solve problems within a few seconds, not hours; VIGIL's enterprise pilot reports at least a 4× diagnosis speed-up and a 39% reduction in interaction rounds under a comparable agentic design [13].
- Limited ticket bounces – efficient routing eliminates need to bounce tickets between departments which saves time and context, aided by auditable routing rationale such as that produced by Topaz [14].
- Accuracy grounded in context – thanks to the RAG system [4], [5], the information on how to troubleshoot the problem is correct, aligned with internal documentation.
- Increased safety and reliability – independent Verification Agent, extending self-reflective and hallucination-corpus grounding techniques [6], [7], makes sure that there are no hallucinations and false statements, no unauthorized actions.
- Human supervision preserved – critical, sensitive, and problematic tickets get automatically escalated with pre-analyzed diagnostic summary, following confidence-gated HITL practice [11].
- System evolution through continuous improvement – human verified answers go into vector knowledge base.

## XV. LIMITATIONS AND FUTURE SCOPE

### A. Limitations

- Dependence on knowledge base accuracy – the RAG module relies on the completeness, accuracy, and up-to-date nature of internal documentation; inaccurate or conflicting SOPs will affect resolution synthesis, a limitation also noted for retrieval-grounded generation generally [4], [5].
- Latency due to inference and computation overhead – orchestrating several steps from the LLM agents is more costly computationally than using single ML classifiers [2], [16].
- Complex multi-ticket issues – tickets which combine unrelated problems (hardware requests plus a database error) pose problems for sequential classification.
- Data privacy – processing sensitive enterprise tickets involves redacting personal information and, where required, running models locally to comply with GDPR or HIPAA.
- Cold start problem – initially low retrieval accuracy for new categories and applications with little historical data causes high levels of escalation, mirroring the class-imbalance and small-dataset findings reported for AutoML ticket classification [12].

### B. Future Scope

- Dynamic sub-ticket decomposition – improving the Understanding Agent to perform recursive sub-tasking of multi-issue tickets into separate child tasks for parallel processing.
- Reinforcement Learning from Human Feedback – using technician rating data and reopening rates as feedback for continuous improvement of decision parameters of the agents.
- Multilingual and multimodal processing – developing multimodal vision-language models to extend the system to be able to process screenshots, diagnostic logs and multilingual requests.
- Predictive incident management – detecting imminent infrastructure outages using the temporal pattern analysis on tickets prior to users' report about them.
- Autonomous tool execution and auto-remediation – secure and sandboxed environment for executing pre-approved administrative tasks (sending password reset emails, restarting frozen containers) by agents according to strict policies of authorization, following the deterministic, tiered policy-gating model demonstrated by VIGIL [13].

## XVI. CONCLUSION

In this paper, we have described the Agentic AI Framework for Intelligent Support Ticket Classification, Routing and Automation. We have focused on solving the ongoing issues of inefficient triaging, mis-routing, prolonged resolution times and poorly grounded automation of enterprise IT service desks. By splitting the support ticket life cycle into a system of eight specialized state-managed AI agents, our framework integrates deep semantic understanding, multi-factor priority assessment, workload-based ticket routing and RAG-based knowledge base retrieval.

The framework overcomes several key problems of the previous approaches by adding a verification step performed by the Verification Agent in conjunction with an independent Human Escalation Agent routing critical and uncertain incidents to human technicians with the pre-assembled diagnostic context. With this design, we achieve maximum automation without compromising safety and auditability of the process. By proposing a methodology, mathematical formulation and benchmarking standards — and by situating the design against the most recent 2024–2026 agentic-AI, retrieval-verification, and explainable-routing literature [4], [6], [7], [12]–[14] — we lay the basis for the future deployment of autonomous agentic AI in enterprises.

## REFERENCES

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.
- [2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. 2019 Conf. North American Chapter of the ACL (NAACL-HLT)*, 2019, pp. 4171–4186.
- [3] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. 2019 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 3982–3992. doi: 10.18653/v1/D19-1410.
- [4] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint arXiv:2312.10997*, 2023.
- [5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 9459–9474.
- [6] A. Asai, S. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2024.
- [7] C. Niu, Y. Wu, J. Zhu, S. Xu, K. Shum, R. Zhong, J. Song, and T. Zhang, "RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models," in *Proc. 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024, pp. 10862–10878.
- [8] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, N. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023.
- [9] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," in *Proc. 33rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2024.
- [10] H. Chase et al., "LangChain / LangGraph: Building stateful, multi-actor applications with LLMs," *Software Documentation and Framework*, 2024. [Online]. Available: <https://github.com/langchain-ai/langgraph>
- [11] R. Monarch and C. Munnely, *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-Centered AI*. Shelter Island, NY: Manning Publications, 2021.
- [12] M. Truss and S. Böhm, "AI-based classification of customer support tickets: State of the art and implementation with AutoML," *arXiv preprint arXiv:2406.01789*, 2024.
- [13] S. Ahuja, N. Kordjazi, E. Yortucboyu, V. Kapoor, M. Dundua, Y. Li, D. Ho, V. Padala, J. Whitted, and R. Steinert, "VIGIL: Towards edge-extended agentic AI for enterprise IT support," *arXiv preprint arXiv:2603.16110*, 2026.
- [14] M. Okamoto, A. K. Erol, and M. Riedl, "Explainable model routing for agentic workflows," in *ACM CHI 2026 Human-Centered Explainable AI (HCXAI) Workshop*, *arXiv preprint arXiv:2604.03527*, 2026.
- [15] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.
- [16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [17] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
- [18] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975. doi: 10.1145/361219.361220.